7.5

Administración de IBM WebSphere MQ

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información en “Avisos” en la página 165.

Esta edición se aplica a la versión 7 release 5 de IBM® WebSphere MQ y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones.

Cuando envía información a IBM, otorga a IBM un derecho no exclusivo para utilizar o distribuir la información de la forma que considere adecuada, sin incurrir por ello en ninguna obligación con el remitente.

Contenido

Administración de IBM WebSphere MQ

La administración de gestores de colas y recursos asociados incluye las tareas que se realizan con frecuencia para activar y gestionar estos recursos. Elija el método que prefiera para administrar los gestores de colas y los recursos asociados.

Puede administrar los objetos de IBM WebSphere MQ de forma local o remota; consulte "Administración local y remota" en la página 7.

Existe una serie de métodos diferentes que puede utilizar para crear y administrar los gestores de colas y sus recursos relacionados en IBM WebSphere MQ. Estos métodos incluyen las interfaces de línea de mandatos, una interfaz gráfica de usuario y una API de administración. Consulte las secciones y los enlaces de este tema si desea más información sobre cada una de estas interfaces.

Existen distintos conjuntos de mandatos que puede utilizar para administrar IBM WebSphere MQ en función de la plataforma:

- "IBM WebSphere MQ mandatos de control" en la página 5
- "IBM WebSphere MQ Mandatos de script (MQSC)" en la página 5
- "Formatos de mandato programable (PCF)" en la página 6

También hay las otras opciones siguientes para crear y gestionar los objetos de IBM WebSphere MQ:

- "IBM WebSphere MQ Explorer" en la página 6
- "La aplicación de configuración predeterminada de Windows" en la página 7
- "El servicio de clústeres de Microsoft (MSCS)" en la página 7

Puede automatizar algunas tareas de administración y supervisión para ambos gestores de colas, locales y remotos, mediante mandatos PCF. Estos mandatos también se pueden simplificar mediante el uso de la Interfaz de administración de IBM WebSphere MQ (MQAI) en algunas plataformas. Si desea más información sobre cómo automatizar las tareas de administración, consulte "Automatización de tareas de administración" en la página 8.

IBM WebSphere MQ mandatos de control

Los mandatos de control le permiten realizar tareas administrativas en los gestores de colas propiamente dichos.

IBM WebSphere MQ para Windows, los sistemas UNIX and Linux® proporcionan los *mandatos de control* que se emiten en la línea de mandatos del sistema.

Los mandatos de control se describen en la sección Creación y gestión de gestores de colas. Para la referencia de mandato de los mandatos de control, consulte Mandatos de control de IBM WebSphere MQ.

IBM WebSphere MQ Mandatos de script (MQSC)

Utilice mandatos MQSC para gestionar objetos de gestor de colas, incluido el propio gestor de colas, colas, definiciones de proceso, canales, canales de conexión de cliente, escuchas, servicios, y objetos de información de autenticación.

Para emitir mandatos MQSC a un gestor de colas se utiliza el mandato `runmqsc`. Esto puede hacerse interactivamente, entrando los mandatos en el teclado, o puede redireccionarse el dispositivo de entrada estándar (stdin) para que ejecute una secuencia de mandatos de un archivo de texto ASCII. En ambos casos, el formato de los mandatos es el mismo.

Puede ejecutar el mandato `runmqsc` en tres modalidades, según los indicadores definidos en el mandato:

- *Modalidad de verificación*, donde los mandatos MQSC se verifican en un gestor de colas loca, pero no se ejecutan

- *Modalidad directa*, en la que los mandatos MQSC se ejecutan en un gestor de colas local
- *Modalidad indirecta*, en la que los mandatos MQSC se ejecutan en un gestor de colas remoto

Los atributos de objeto especificados en los mandatos MQSC aparecen en mayúsculas en esta sección (por ejemplo RQMNAME), aunque no son sensibles a las mayúsculas y minúsculas. Los nombres de atributo de los mandatos MQSC están limitados a ocho caracteres.

Los mandatos MQSC están disponibles en todas las plataformas . Los mandatos MQSC se resumen en la sección Comparación de conjuntos de mandatos.

En Windows, UNIX o Linux, puede utilizar MQSC como mandatos únicos emitidos en la línea de mandatos del sistema. Para emitir varios mandatos o mandatos más complicados, el MQSC se puede construir en un archivo que ejecute desde la línea de mandatos del sistema Windows, UNIX o Linux. MQSC se puede enviar a un gestor de colas remoto. Para información más detallada, consulte Referencia de MQSC.

"Mandatos (MQSC) de script" en la página 75 contiene una descripción de cada mandato MQSC y su sintaxis.

Consulte el apartado "Realización de tareas de administración local utilizando mandatos MQSC" en la página 74 para obtener información sobre la utilización de los mandatos MQSC en la administración local.

Formatos de mandato programable (PCF)

Los formatos de mandato programable (PCF) definen mensajes de mandato y de respuesta que se pueden intercambiar entre un programa y cualquier gestor de colas (que admita PCF) en una red. Puede utilizar los mandatos PCF en el programa de aplicación de gestión de sistemas para la administración de objetos de IBM WebSphere MQ: los objetos de información de autenticación, los canales, los escuchas de canales, las listas de nombres, las definiciones de proceso, los gestores de colas, las colas, los servicios y las clases de almacenamiento. La aplicación puede operar desde un único punto de la red para comunicar información de mandato y de respuesta a cualquier gestor de colas, local o remoto, utilizando el gestor de colas local.

Si desea más información sobre los PCF, consulte "Introducción a los Formatos de mandato programable" en la página 9.

Para la definición de los PCF y las estructuras para los mandatos y las respuestas, consulte Referencia de formatos de mandatos programables.

IBM WebSphere MQ Explorer

Utilizando IBM WebSphere MQ Explorer, puede realizar las acciones siguientes:

- Definir y controlar diversos recursos, incluyendo gestores de colas, colas, definiciones de proceso, listas de nombres, canales, canales de conexión de cliente, escuchas, servicios y clústeres.
- Iniciar o detener un gestor de colas local y los procesos asociados al mismo.
- Ver gestores de colas y sus objetos asociados en su estación de trabajo o desde otras estaciones de trabajo.
- Comprobar el estado de gestores de colas, clústeres y canales.
- Comprobar qué aplicaciones, usuarios o canales tienen una cola determinada abierta, a partir del estado de la cola.

En sistemas Windows y Linux , puede iniciar IBM WebSphere MQ Explorer utilizando el menú del sistema, el archivo ejecutable `MQExplorer` o el mandato **`strmqcfg`** .

En Linux, para iniciar correctamente IBM WebSphere MQ Explorer, debe poder grabar un archivo en el directorio de inicio y debe existir un directorio de inicio.

Puede utilizar IBM WebSphere MQ Explorer para administrar gestores de colas remotos en otras plataformas, incluyendo z/OS; para obtener más información y para descargar el SupportPac MS0T, consulte https://www.ibm.com/support/docview.wss?uid=swg24021041.

Consulte “Administración utilizando IBM WebSphere MQ Explorer” en la página 57 para obtener más información.

La aplicación de configuración predeterminada de Windows

Puede utilizar el programa de configuración predeterminada de Windows para crear un conjunto de *iniciador* (o predeterminado) de objetos IBM WebSphere MQ . En la Tabla 1: Objetos creados por la aplicación de configuración predeterminada de Windows se muestra un resumen de los objetos predeterminados que se crean.

El servicio de clústeres de Microsoft (MSCS)

Microsoft Cluster Service (MSCS) le permite conectar servidores a un *clúster*, lo que proporciona una disponibilidad más alta de datos y aplicaciones y facilita la gestión del sistema. MSCS puede detectar y recuperarse automáticamente de los errores del servidor o de las aplicaciones.

Es importante no confundir los clústeres en el sentido de MSCS con los clústeres de IBM WebSphere MQ. La diferencia es la siguiente:

IBM WebSphere MQ clústeres
: Son grupos de dos o más gestores de colas en uno o varios sistemas que proporcionan una interconexión automática y permiten que se compartan las colas entre los mismos para fines de equilibrio de carga y redundancia.

Clústeres de MSCS
: Grupos de sistemas, conectados entre sí y configurados de tal modo que si se produce un error en uno de ellos, MSCS ejecuta una *migración tras error*, transfiere los datos de estado de las aplicaciones del sistema anómalo a otro sistema del clúster y reinicia su ejecución en el mismo.

Soporte de Microsoft Cluster Service (MSCS) proporciona información detallada sobre cómo configurar el sistema IBM WebSphere MQ for Windows para utilizar MSCS.

Conceptos relacionados

Visión general técnica de WebSphere MQ

“Administración de objetos de IBM WebSphere MQ locales” en la página 70
En esta sección se explica cómo administrar objetos de IBM WebSphere MQ locales para dar soporte a los programas de aplicación que utilizan la Interfaz de cola de mensajes (MQI). En este contexto, administración local significa crear, visualizar, cambiar, copiar y suprimir objetos de IBM WebSphere MQ.

“Administración de objetos de IBM WebSphere MQ remotos” en la página 107

Consideraciones cuando se pierde el contacto con el gestor de recursos XA

Tareas relacionadas

Planificación

Configuración

Referencia relacionada

Escenarios de soporte transaccional

Administración local y remota

Los objetos de WebSphere MQ se pueden administrar de forma local o remota.

Administración local significa llevar a cabo tareas de administración en cualquier gestor de colas que se haya definido en el sistema local. Puede acceder a otros sistemas, por ejemplo mediante el programa de emulación de terminal **telnet** de TCP/IP, y llevar a cabo la administración allí. En WebSphere MQ, esto puede considerarse como administración local porque no hay ningún canal implicado, es decir, la comunicación la gestiona el sistema operativo.

WebSphere MQ da soporte a la administración desde un único punto de contacto mediante lo que se conoce como *administración remota*. Esto permite emitir mandatos desde el sistema local que se procesarán en otro sistema y también es aplicable a WebSphere MQ Explorer. Por ejemplo, puede emitir

un mandato remoto para cambiar una definición de cola en un gestor de colas remoto. No es necesario que se conecte a ese sistema, aunque sí que necesita tener definidos los canales apropiados. El gestor de colas y el servidor de mandatos del sistema de destino deben estar ejecutándose.

Algunos mandatos no se pueden emitir de esta manera, especialmente los que crean o inician gestores de colas y los que inician servidores de mandatos. Para realizar este tipo de tarea, debe iniciar la sesión en el sistema remoto y emitir los mandatos desde allí o crear un proceso que pueda emitirlos automáticamente. Esta restricción se aplica también a WebSphere MQ Explorer.

En el apartado "Administración de objetos de IBM WebSphere MQ remotos" en la página 107 puede ver información más detallada sobre la administración remota.

Cómo utilizar los mandatos de control de IBM WebSphere MQ

Esta sección describe cómo utilizar los mandatos de control de IBM WebSphere MQ.

Si desea emitir mandatos de control, su ID de usuario debe ser miembro del grupo mqm. Para obtener más información sobre esto, consulte Autorización para administrar WebSphere MQ en sistemas UNIX, Linux y Windows . Además, tenga en cuenta la siguiente información específica del entorno:

WebSphere MQ para Windows
Todos los mandatos de control se pueden emitir desde una línea de mandatos. Los nombres de los mandatos y sus indicadores no son sensibles a las mayúsculas y minúsculas: puede entrarlos en mayúsculas, minúsculas o en una combinación de mayúsculas y minúsculas. Sin embargo, los argumentos de los mandatos de control (como los nombres de colas) sí son sensibles a las mayúsculas y minúsculas.

En las descripciones de sintaxis, el guión (-) se utiliza como indicador de distintivo. Puede utilizar una barra inclinada (/) en vez del guión.

WebSphere MQ para sistemas UNIX and Linux
Todos los mandatos de control de WebSphere MQ se pueden emitir desde un shell. Todos los mandatos distinguen entre mayúsculas y minúsculas.

Se puede emitir un subconjunto de los mandatos de control utilizando IBM WebSphere MQ Explorer.

Si desea más información, consulte Los mandatos de control de WebSphere MQ

Automatización de tareas de administración

Puede decidir que podría ser beneficioso para su instalación automatizar algunas tareas de administración y supervisión. Puede automatizar tareas de administración para los gestores de colas locales y remotos utilizando mandatos de formato de mandato programable (PCF). En esta sección se presupone que tiene experiencia en la administración de objetos WebSphere MQ.

mandatos PCF

Los mandatos PCF (formato de mandato programable) de WebSphere MQ se pueden utilizar para programar tareas de administración en un programa de administración. De este modo, desde un programa puede manipular objetos de gestores de colas (colas, definiciones de procesos, listas de nombres, canales, canales de conexión de clientes, escuchas, servicios y objetos de información de autenticación), e incluso manipular los mismos gestores de colas.

Los mandatos PCF abarcan el mismo tipo de funciones que las proporcionadas por los mandatos MQSC. Puede escribir un programa que emita mandatos PCF a cualquier gestor de colas de la red desde un solo nodo. De este modo, puede centralizar y automatizar las tareas de administración.

Cada mandato PCF es una estructura de datos que se inserta en la parte de datos de la aplicación de un mensaje WebSphere MQ. Cada mandato se envía al gestor de colas de destino utilizando la función MQPUT de MQI del mismo modo que cualquier otro mensaje. Si el servidor de mandatos se está ejecutando en el gestor de colas que recibe el mensaje, el servidor de mandatos lo interpreta como un mensaje de mandato y ejecuta el mandato. Para obtener las respuestas, la aplicación emite una llamada MQGET y los

datos de respuesta se devuelven en otra estructura de datos. La aplicación puede entonces procesar la respuesta y actuar en conformidad.

Nota: A diferencia de los mandatos MQSC, los mandatos PCF y sus respuestas no están en un formato de texto legible por el usuario.

En resumen, estas son algunas de las cosas necesarias para crear un mensaje de mandato PCF:

Descriptor de mensaje
Es un descriptor de mensaje estándar de WebSphere MQ, en el que:

- El tipo de mensaje (*MsqType*) es MQMT_REQUEST.
- El formato del mensaje (*Format*) es MQFMT_ADMIN.

Datos de aplicación
Contienen el mensaje PCF, incluida la cabecera PCF, en el que:

- El tipo de mensaje PCF (*Type*) especifica MQCFT_COMMAND.
- El identificador de mandato especifica el mandato, por ejemplo, *Change Queue* (MQCMD_CHANGE_Q).

Para obtener una descripción completa de las estructuras de datos PCF y cómo implementarlas, consulte "Introducción a los Formatos de mandato programable" en la página 9.

Atributos de objetos PCF

Los atributos de objetos en PCF no están limitados a ocho caracteres, como sí lo están para los mandatos MQSC. Se muestran en esta guía en cursiva. Por ejemplo, el equivalente PCF de RQMNAME es *RemoteQMgrName*.

PCF de escape

Los PCF de escape son mandatos PCF que contienen mandatos MQSC dentro del texto del mensaje. Los PCF se pueden utilizar para enviar mandatos a un gestor de colas remoto. Para obtener más información sobre los PCF de escape, consulte Escape.

Introducción a los Formatos de mandato programable

Los formatos de mandato programable (PCF) definen mensajes de mandato y de respuesta que se pueden intercambiar entre un programa y cualquier gestor de colas (que admita PCF) en una red. Los PCF simplifican la administración del gestor de colas y otras tareas de administración de red. Pueden utilizarse para resolver el problema de la compleja administración de redes distribuidas, especialmente cuando las redes crecen en tamaño y complejidad.

Los Formatos de mandato programable descritos en esta documentación del producto están soportados en:

- IBM WebSphere MQ para AIX
- IBM WebSphere MQ para HP-UX
- IBM WebSphere MQ para Linux
- IBM WebSphere MQ para Solaris
- IBM WebSphere MQ para Windows
- IBM WebSphere MQ for HP Integrity NonStop Server

Problema que resuelven los mandatos PCF

La administración de redes distribuidas puede ser compleja. Los problemas de administración continúan creciendo a medida que las redes presentan mayor tamaño y complejidad.

Como ejemplos de administración específicos de mensajes y colas, cabe citar:

- Gestión de recursos.

  Por ejemplo, la creación y supresión de colas.
- Supervisión del rendimiento.

  Por ejemplo, una mayor profundidad de cola o un mayor índice de mensajes.
- Control.

  Por ejemplo, ajustar los parámetros de colas tales como la profundidad máxima de cola, la longitud máxima de los mensajes, y la habilitación e inhabilitación de colas.
- Direccionamiento de mensajes.

  Definición de rutas alternativas a través de una red.

Los mandatos PCF de WebSphere MQ pueden utilizarse para simplificar la administración de colas y otras tareas de administración de la red. Los mandatos PCF permiten utilizar una única aplicación para realizar la administración de red desde un único gestor de colas en la red.

¿Qué son los PCF?

Los PCF definen mensajes de mandato y de respuesta que se pueden intercambiar entre un programa y cualquier gestor de colas (que admita PCF) en una red. Puede utilizar los mandatos PCF en un programa de aplicación de gestión de sistemas para la administración de objetos de WebSphere MQ: objetos de información de autenticación, canales, escuchas de canal, listas de nombres, definiciones de proceso, gestores de colas, colas, servicios y clases de almacenamiento. La aplicación puede operar desde un único punto de la red para comunicar información de mandato y de respuesta a cualquier gestor de colas, local o remoto, utilizando el gestor de colas local.

Cada gestor de colas tiene una cola de administración con un nombre de cola estándar y la aplicación puede enviar mensajes de mandato PCF a dicha cola. Cada gestor de colas tiene también un servidor de mandatos para prestar servicio a los mensajes de mandato desde la cola de administración. Los mensajes de mandatos PCF pueden, por consiguiente, ser procesados por cualquier gestor de colas de la red y los datos de respuesta pueden devolverse a la aplicación mediante la cola de respuesta especificada. Los mensajes de mandatos PCF y de respuesta se envían y reciben utilizando la Interfaz de colas de mensajes (MQI) habitual.

Para obtener una lista de los mandatos PCF disponibles, incluidos sus parámetros, consulte Definiciones de los formatos de mandato programable.

Utilización de formatos de mandato programables

Puede utilizar PCF en un programa de gestión de sistemas para la administración remota de WebSphere MQ.

Esta sección incluye:

- “Mensajes de mandato PCF” en la página 10
- “Respuestas” en la página 13
- Reglas para la denominación de objetos de IBM WebSphere MQ
- “Comprobación de autorización para mandatos PCF” en la página 15

Mensajes de mandato PCF

Los mensajes de mandato PCF constan de una cabecera PCF, parámetros identificados en dicha cabecera y los datos de mensaje definidos por el usuario. Los mensajes se emiten mediante llamadas de la interfaz de cola de mensajes.

Cada mandato y sus parámetros se envían como un mensaje de mandato independiente que contiene una cabecera PCF seguida de varias estructuras de parámetros; para obtener más información sobre la cabecera PCF, consulte MQCFH - Cabecera PCF y para ver un ejemplo de una estructura de parámetros, consulte MQCFST - Parámetro de serie PCF. La cabecera PCF identifica el mandato y las estructuras de

parámetro que siguen en el mismo mensaje. Cada estructura de mandato proporciona un parámetro al mandato.

La respuestas a los mandatos, generadas por el servidor de mandatos, tienen una estructura similar. Hay una cabecera PCF, seguida de varias estructuras de parámetros. Las respuestas pueden constar de más de un mensaje, pero los mandatos siempre constan de un único mensaje.

En plataformas distintas de z/OS, la cola a la que se envían los mandatos PCF siempre se denomina SYSTEM.ADMIN.COMMAND.QUEUE.

Cómo emitir mensajes de mandatos PCF

Utilice las llamadas normales de la interfaz de cola de mensajes (MQI), MQPUT, MQGET, etc., para colocar y recuperar mensajes de respuestas y de mandatos PCF en y desde sus colas.

Nota:

Asegúrese de que el servidor de mandatos se ejecuta en el gestor de colas de destino para que el mandato PCF se procese en dicho gestor de colas.

Para obtener una lista de los archivos de cabecera suministrados, consulte Archivos COPY, de cabecera, de inclusión y módulos de WebSphere MQ.

Descriptor de mensaje para un mandato PCF

El descriptor de mensaje WebSphere MQ está completamente documentado en MQMD - Descriptor de mensaje.

Un mensaje de mandato PCF contiene los siguientes campos en el descriptor de mensaje:

Report
Cualquier valor válido, según sea necesario.

MsgType
Este campo debe ser MQMT_REQUEST para indicar un mensaje que requiere una respuesta.

Expiry
Cualquier valor válido, según sea necesario.

Feedback
Se establece en MQFB_NONE.

Encoding
Sí realiza el envío a sistemas Windows, UNIX o Linux, establezca este campo en la codificación utilizada para los datos del mensaje; la conversión se lleva a cabo si es necesario.

CodedCharSetId
Si realiza el envío a sistemasSistemas Windows, UNIX o Linux, establezca este campo en el identificador del conjunto de caracteres codificado utilizado para los datos del mensaje; la conversión se lleva a cabo si es necesario.

Format
Se establece en MQFMT_ADMIN.

Priority
Cualquier valor válido, según sea necesario.

Persistence
Cualquier valor válido, según sea necesario.

MsgId
La aplicación de envío puede especificar cualquier valor, o puede especificarse MQMI_NONE para solicitar al gestor de colas que genere un identificador de mensaje exclusivo.

CorrelId
La aplicación emisora puede especificar cualquier valor, o puede especificarse MQCI_NONE para indicar que no hay identificador de correlación.

ReplyToQ
Nombre de la cola que debe recibir la respuesta.

ReplyToQMgr
Nombre del gestor de colas para la respuesta (o en blanco).

Campos de contexto de mensaje
Estos campos pueden establecerse en cualquier valor válido, según sea necesario. Normalmente, la opción MQPMO_DEFAULT_CONTEXT se utiliza para establecer los campos de contexto de mensaje en los valores predeterminados.

Si utiliza una estructura MQMD versión 2, debe establecer los siguientes campos adicionales:

GroupId
Se establece en MQGI_NONE.

MsgSeqNumber
Se establece en 1.

Offset
Se establece en 0.

MsgFlags
Se establece en MQMF_NONE.

OriginalLength
Se establece en MQOL_UNDEFINED.

Envío de datos de usuario

Las estructuras PCF también pueden utilizarse para enviar datos de mensaje definidos por el usuario. En este caso, el campo *Format* del descriptor de mensaje debe establecerse en MQFMT_PCF.

Envío y recepción de mensajes PCF en una cola especificada

Envío de mensajes PCF a una cola especificada

Para enviar un mensaje a una cola especificada, la llamada mqPutBag convierte el contenido del paquete especificado en un mensaje PCF y envía el mensaje a la cola especificada. El contenido del paquete de deja igual después de la llamada.

Como entrada para esta llamada, debe proporcionar:

- Un manejador de conexión MQI.
- Un manejador de objeto para la cola en la que va a colocarse el mensaje.
- Un descriptor de mensaje. Para obtener más información sobre el descriptor de mensaje, consulte MQMD - Descriptor de mensaje.
- Las opciones de transferencia de mensajes utilizando la estructura MQPMO. Para obtener más información acerca de la estructura MQPMO, consulte MQPMO – Opciones de transferir mensaje.
- El manejador del paquete que se convertirá en un mensaje.

  Nota: Si el paquete contiene un mensaje de administración y se utilizó la llamada mqAddInquiry para insertar valores en el paquete, el valor del elemento de datos MQIASY_COMMAND debe ser un mandato INQUIRE reconocido por la MQAI.

Para obtener una descripción completa de la llamada mqPutBag, consulte mqPutBag.

Recepción de los mensajes PCF de una cola especificada

Para recibir un mensaje de una cola especificada, la llamada mqGetBag obtiene un mensaje PCF de una cola especificada y convierte los datos del mensaje en un paquete de datos.

Como entrada para esta llamada, debe proporcionar:

- Un manejador de conexión MQI.
- Un manejador de objeto de la cola de la que se debe leer el mensaje.
- Un descriptor de mensaje. Dentro de la estructura MQMD, el parámetro `Format` debe ser MQFMT_ADMIN, MQFMT_EVENT o MQFMT_PCF.

  Nota: Si el mensaje se recibe dentro de una unidad de trabajo (es decir, con la opción MQGMO_SYNCPOINT) y el mensaje tiene un formato no soportado, la unidad de trabajo puede restituirse. El mensaje se reincorporará en la cola y podrá recuperarse utilizando la llamada MQGET en lugar de la llamada mqGetBag. Para obtener más información sobre el descriptor de mensaje, consulte MQGMO - Opciones de obtención de mensajes.
- Las opciones de obtención de mensajes utilizando la estructura MQPMO. Para obtener más información acerca de la estructura MQGMO, consulte MQMD - Descriptor de mensaje.
- El manejador del paquete para contener el mensaje convertido.

Para obtener una descripción completa de la llamada mqGetBag, consulte mqGetBag.

Respuestas

En respuesta a cada mandato, el servidor de mandatos genera uno o varios mensajes de respuesta. Un mensaje de respuesta tiene un formato parecido a un mensaje de mandato.

La cabecera PCF tiene el mismo valor de identificador de mandato que el mandato para el que es una respuesta (consulte MQCFH - Cabecera PCF para obtener más detalles). El identificador del mensaje y el identificador de correlación se establecen de acuerdo con las opciones de informe de la solicitud.

Si el tipo de cabecera PCF del mensaje de mandato es MQCFT_COMMAND, sólo se generan respuestas estándar. Los mandatos de este tipo se admiten en todas las plataformas, con la excepción de z/OS. Las aplicaciones más antiguas no admiten PCF en z/OS; WebSphere MQ Windows Explorer es una de estas aplicaciones (no obstante, la versión 6.0 o posteriores de IBM WebSphere MQ Explorer sí admite PCF en z/OS).

Si el tipo de cabecera PCF del mensaje de mandato es MQCFT_COMMAND_XR, se generan respuestas ampliadas o estándar. Los mandatos de este tipo se admiten en z/OS en otras plataformas. Los mandatos emitidos en z/OS sólo generan respuestas ampliadas. En otras plataformas, puede generarse cualquiera de los dos tipos de respuesta.

Si un mandato especifica un nombre de objeto genérico, se devuelve una respuesta independiente en su propio mensaje para cada objeto coincidente. Para la generación de respuestas, un mandato con un nombre genérico se trata como varios mandatos individuales (excepto para el campo de control MQCFC_LAST o MQCFC_NOT_LAST). En caso contrario, un mensaje de mandato genera un mensaje de respuesta.

Determinadas respuestas PCF pueden devolver una estructura aun cuando no se solicite. Esta estructura se muestra en la definición de la respuesta (Definiciones de los formatos de mandato programables) como *siempre devuelta*. Esto se explica porque para estas respuestas es necesario indicar el nombre de los objetos en la respuesta para identificar el objeto al que se aplican los datos.

Descriptor de mensaje para una respuesta

Un mensaje de respuesta tiene los siguientes campos en el descriptor de mensaje:

MsgType
Este campo es MQMT_REPLY.

MsgId
Este campo se genera en el gestor de colas.

CorrelId
Este campo se genera de acuerdo con las opciones de informe del mensaje de mandato.

Format
Este campo es MQFMT_ADMIN.

Encoding
Se establece en MQENC_NATIVE.

CodedCharSetId
Se establece en MQCCSI_Q_MGR.

Persistence
Igual que en el mensaje de mandato.

Priority
Igual que en el mensaje de mandato.

La respuesta se genera con MQPMO_PASS_IDENTITY_CONTEXT.

Respuestas estándar

Los mensajes de mandato con el tipo de cabecera MQCFT_COMMAND generan respuestas estándar. Los mandatos de este tipo se admiten en todas las plataformas, con la excepción de z/OS.

Hay tres tipos de respuesta estándar:

- respuesta OK
- Respuesta de error
- Respuesta de datos

respuesta OK

Esta respuesta consta de un mensaje que empieza con una cabecera de formato de mandato, con un campo *CompCode* de MQCC_OK o MQCC_WARNING.

Para MQCC_OK, *Reason* es MQRC_NONE.

Para MQCC_WARNING, *Reason* identifica la naturaleza del aviso. En este caso, la cabecera de formato de mandato puede ir seguida de una o más estructuras de parámetro de aviso apropiadas para este código de razón.

En cualquier caso, para un mandato de consulta es posible que sigan varias estructuras de parámetro, como se describe en las siguientes secciones.

Respuesta de error

Si el mandato tiene un error, se envía uno o varios mensajes de respuesta de error (es posible que se envíe más de uno incluso para un mandato que normalmente sólo tendría un mensaje de respuesta). Estos mensajes de error tienen el valor MQCFC_LAST o MQCFC_NOT_LAST establecido, según corresponda.

Cada mensaje de este tipo empieza con una cabecera de formato de respuesta, con un valor *CompCode* de MQCC_FAILED y un campo *Reason* que identifica el error en particular. Por lo general, cada mensaje describe un error distinto. Además, cada mensaje tiene ninguna o una estructura de parámetro de error (nunca más de una) después de la cabecera. Esta estructura de parámetros, si existe, es una estructura MQCFIN, con un campo *Parameter* que contiene uno de estos valores:

- MQIACF_PARAMETER_ID

  El campo *Value* de la estructura es el identificador del parámetro que tenía el error (por ejemplo, MQCA_Q_NAME).

- MQIACF_ERROR_ID

  Este valor se utiliza con un valor *Reason* (en la cabecera de formato de mandato) de MQRC_UNEXPECTED_ERROR. El campo *Value* de la estructura MQCFIN es el código de razón inesperado recibido por el servidor de mandatos.

- MQIACF_SELECTOR

  Este valor se produce si una estructura de lista (MQCFIL) enviada con el mandato contiene un selector duplicado o uno que no es válido. El campo *Reason* de la cabecera de formato de mandato identifica el

error y el campo *Value* de la estructura MQCFIN es el valor de parámetro de la estructura MQCFIL del mandato que ha fallado.

- MQIACF_ERROR_OFFSET

  Este valor se produce cuando se produce un error de comparación de datos en el mandato Sondear canal. El campo *Value* de la estructura es el desplazamiento del error de comparación de Ping Channel.

- MQIA_CODED_CHAR_SET_ID

  Este valor se produce cuando el identificador del conjunto de caracteres codificado del descriptor de mensaje del mensaje de mandato PCF entrante no coincide con el del gestor de colas de destino. El campo *Value* de la estructura es el identificador del conjunto de caracteres codificado del gestor de colas.

El último (o único) mensaje de respuesta de error es una respuesta de resumen, con un campo *CompCode* de MQCC_FAILED y un campo *Reason* de MQRCCF_COMMAND_FAILED. Este mensaje no tiene ninguna estructura de parámetro después de la cabecera.

Respuesta de datos

Esta respuesta consta de una respuesta OK (como se ha descrito anteriormente) para un mandato de consulta. La respuesta OK va seguida por estructuras adicionales que contienen los datos solicitados como se describe en Definiciones de los formatos de mandato programable.

Las aplicaciones no deben depender de que estas estructuras de parámetro adicionales se devuelvan en un orden determinado.

Comprobación de autorización para mandatos PCF

Cuando se procesa un mandato PCF, se utiliza el valor de *UserIdentifier* del descriptor del mensaje del mandato para las comprobaciones de autorización de objetos de WebSphere MQ. La comprobación de autorización se implementa de forma diferente en cada plataforma como se describe en este tema.

Las comprobaciones se realizan en el sistema en el que se está procesando el mandato; por lo tanto, este ID de usuario debe existir en el sistema de destino y tener las autorizaciones necesarias para procesar el mandato. Si el mensaje proviene de un sistema remoto, una forma de alcanzar el ID existente en el sistema de destino es tener un ID de usuario coincidente en los sistemas local y remoto.

IBM WebSphere MQ para sistemas Windows, UNIX and Linux

Para procesar cualquier mandato PCF, el ID de usuario debe tener autorización *dsp* para el objeto de gestor de colas en el sistema de destino. Además, las comprobaciones de autorización de objetos de WebSphere MQ se realizan para determinados mandatos PCF, como se muestra en la Tabla 1 en la página 16.

Para procesar cualquiera de los mandatos siguientes, el ID de usuario debe pertenecer al grupo *mqm*.

Nota: Para Windows **únicamente**, el ID de usuario puede pertenecer al grupo de *administradores* o al grupo *mqm*.

- Cambiar canal
- Copiar canal
- Crear canal
- Suprimir canal
- Sondear canal
- Restablecer canal
- Iniciar canal
- Detener canal
- Iniciar iniciador de canal

- Iniciar escucha de canal
- Resolver canal
- Restablecer clúster
- Renovar clúster
- Suspender gestor de colas
- Reanudar gestor de colas

WebSphere MQ para HP Integrity NonStop Server

Para procesar cualquier mandato PCF, el ID de usuario debe tener autorización *dsp* para el objeto de gestor de colas en el sistema de destino. Además, se realizan comprobaciones de autorización de objetos de IBM WebSphere MQ para determinados mandatos PCF, como se muestra en la Tabla 1 en la página 16.

Para procesar cualquiera de los siguientes mandatos, el ID de usuario debe pertenecer al grupo *mqm*:

- Cambiar canal
- Copiar canal
- Crear canal
- Suprimir canal
- Sondear canal
- Restablecer canal
- Iniciar canal
- Detener canal
- Iniciar iniciador de canal
- Iniciar escucha de canal
- Resolver canal
- Restablecer clúster
- Renovar clúster
- Suspender gestor de colas
- Reanudar gestor de colas

Autorizaciones de objetos de WebSphere MQ

Tabla 1. Windows, HP Integrity NonStop Server, UNIX and Linux -autorizaciones sobre objetos

Mandato	**Autorización de objetos de WebSphere MQ**	**Autorización de clases (para tipo de objeto)**
Modificar información de autorización	dsp y chg	n/d
Cambiar canal	dsp y chg	n/d
Modificar escucha de canal	dsp y chg	n/d
Modificar canal de conexión de cliente	dsp y chg	n/d
Modificar lista de nombres	dsp y chg	n/d
Modificar proceso	dsp y chg	n/d
Modificar cola	dsp y chg	n/d
Cambiar gestor de colas	chg *véase la nota 3 y la nota 5*	n/d

Tabla 1. Windows, HP Integrity NonStop Server, UNIX and Linux -autorizaciones sobre objetos (continuación)		
Mandato	**Autorización de objetos de WebSphere MQ**	**Autorización de clases (para tipo de objeto)**
Cambiar servicio	dsp y chg	n/d
Borrar cola	clr	n/d
Copiar información de autenticación	dsp	crt
Copiar información de autenticación (reemplazar) *ver nota 1*	*de:* dsp *a:* chg	crt
Copiar canal	dsp	crt
Copiar canal (reemplazar) *ver nota 1*	*de:* dsp *a:* chg	crt
Copiar escucha de canal	dsp	crt
Copiar escucha de canal (reemplazar) *ver nota 1*	*de:* dsp *a:* chg	crt
Copiar canal de conexión de cliente	dsp	crt
Copiar canal de conexión de cliente (reemplazar) *ver nota 1*	*de:* dsp *a:* chg	crt
Copiar lista de nombres	dsp	crt
Copiar lista de nombres (reemplazar) *ver nota 1*	*de:* dsp *a:* dsp y chg	crt
Copiar proceso	dsp	crt
Copiar proceso (reemplazar) *ver nota 1*	*de:* dsp *a:* chg	crt
Copiar cola	dsp	crt
Copiar cola (reemplazar) *ver nota 1*	*de:* dsp *a:* dsp y chg	crt
Crear información de autenticación	*(información de autenticación predeterminada del sistema)* dsp	crt
Crear información de autenticación (reemplazar) *ver nota 1*	*(información de autenticación predeterminada del sistema)* dsp *a:* chg	crt
Crear canal	*(canal predeterminado del sistema)* dsp	crt
Crear canal (reemplazar) *ver nota 1*	*(canal predeterminado del sistema)* dsp *a:* chg	crt
Crear escucha de canal	*(escucha predeterminado del sistema)* dsp	crt
Crear escucha de canal (reemplazar) *ver nota 1*	*(escucha predeterminado del sistema)* dsp *a:* chg	crt
Crear canal de conexión de cliente	*(canal predeterminado del sistema)* dsp	crt

Tabla 1. Windows, HP Integrity NonStop Server, UNIX and Linux -autorizaciones sobre objetos (continuación)

Mandato	**Autorización de objetos de WebSphere MQ**	**Autorización de clases (para tipo de objeto)**
Crear canal de conexión de cliente (reemplazar) *ver nota 1*	*(canal predeterminado del sistema)* dsp *a:* chg	crt
Crear lista de nombres	*(lista de nombres predeterminada del sistema)* dsp	crt
Crear lista de nombres (reemplazar) *ver nota 1*	*(lista de nombres predeterminada del sistema)* dsp *a:* dsp y chg	crt
Crear proceso	*(proceso predeterminado del sistema)* dsp	crt
Crear proceso (reemplazar) *ver nota 1*	*(proceso predeterminado del sistema)* dsp *a:* chg	crt
Crear cola	*(cola predeterminada del sistema)* dsp	crt
Crear cola (reemplazar) *ver nota 1*	*(cola predeterminada del sistema)* dsp *a:* dsp y chg	crt
Crear servicio	*(cola predeterminada del sistema)* dsp	crt
Crear servicio (reemplazar) *ver nota 1*	*(cola predeterminada del sistema)* dsp *a:* chg	crt
Suprimir información de autenticación	dsp y dlt	n/d
Suprimir registro de autorización	*(objeto de gestor de colas)* chg *ver nota 4*	*ver nota 4*
Suprimir canal	dsp y dlt	n/d
Suprimir escucha de canal	dsp y dlt	n/d
Suprimir canal de conexión de cliente	dsp y dlt	n/d
Suprimir lista de nombres	dsp y dlt	n/d
Suprimir proceso	dsp y dlt	n/d
Suprimir cola	dsp y dlt	n/d
Suprimir servicio	dsp y dlt	n/d
Consultar información de autenticación	dsp	n/d
Consultar registros de autorización	*ver nota 4*	*ver nota 4*
Consultar canal	dsp	n/d
Consultar escucha de canal	dsp	n/d
Consultar estado de canal (para **ChannelType** MQCHT_CLSSDR)	inq	n/d
Consultar canal de conexión de cliente	dsp	n/d
Consultar lista de nombres	dsp	n/d

Tabla 1. Windows, HP Integrity NonStop Server, UNIX and Linux -autorizaciones sobre objetos (continuación)		
Mandato	**Autorización de objetos de WebSphere MQ**	**Autorización de clases (para tipo de objeto)**
Consultar proceso	dsp	n/d
Consultar cola	dsp	n/d
Consultar gestor de colas	*ver nota 3*	n/d
Consultar estado de la cola	dsp	n/d
Consultar servicio	dsp	n/d
Sondear canal	ctrl	n/d
Sondear gestor de colas	*ver nota 3*	n/d
Renovar gestor de colas	(objeto de gestor de colas) chg	n/d
Renovar seguridad (para **SecurityType** MQSECTYPE_SSL)	(objeto de gestor de colas) chg	n/d
Restablecer canal	ctrlx	n/d
Restablecer gestor de colas	(objeto de gestor de colas) chg	n/d
Restablecer estadísticas de la cola	dsp y chg	n/d
Resolver canal	ctrlx	n/d
Establecer registro de autorización	*(objeto de gestor de colas)* chg *ver nota 4*	*ver nota 4*
Iniciar canal	ctrl	n/d
Detener canal	ctrl	n/d
Detener conexión	(objeto de gestor de colas) chg	n/d
Iniciar escucha	ctrl	n/d
Detener escucha	ctrl	n/d
Iniciar servicio	ctrl	n/d
Detener servicio	ctrl	n/d
Esc	*ver nota 2*	*ver nota 2*

Notas:

1. Este mandato se aplica si el objeto que debe reemplazarse ya existe, de lo contrario la comprobación de autorización es como para Crear o Copiar sin reemplazar.
2. La autorización necesaria se determina mediante el mandato MQSC definido por el texto de escape y es equivalente a uno de los mandatos anteriores.
3. Para procesar cualquier mandato PCF, el ID de usuario debe tener autorización dsp para el objeto de gestor de colas en el sistema de destino.
4. Este mandato PCF está autorizado a menos que el servidor de mandatos se haya iniciado con el parámetro -a. De forma predeterminada, el servidor de mandatos se inicia al iniciar el gestor de colas, y sin el parámetro -a. Consulte la guía de administración del sistema para obtener más información.

5. Otorgar a un ID de usuario autorización *chg* para un gestor de colas concede la capacidad de establecer registros de autorización para todos los grupos y usuarios. Esta autorización no debe otorgarse a usuarios ni aplicaciones ordinarios.

WebSphere MQ también proporciona algunos puntos de salida de seguridad de canal de modo que pueda suministrar sus propios programas de salida de usuario para la comprobación de la seguridad. Los detalles se proporcionan en Visualización de un canal.

Utilizar la MQAI para simplificar el uso de los PCF

La MQAI es una interfaz de administración para WebSphere MQ que está disponible en las plataformas AIX, HP-UX, IBM i, Linux, Solaris y Windows .

La MQAI realiza tareas de administración en un gestor de colas mediante el uso de *paquetes de datos*. Los paquetes de datos permiten manejar las propiedades (o parámetros) de los objetos de forma más sencilla que utilizando mandatos PCF.

Utilice la MQAI de las siguientes maneras:

Simplificar el uso de mensajes PCF
La MQAI es una forma fácil de administrar WebSphere MQ; no es necesario que escriba sus propios mensajes PCF, evitando así los problemas asociados a las estructuras de datos complejas.

Para pasar parámetros en programas escritos utilizando llamadas MQI, el mensaje PCF debe contener el mandato y detalles de los datos de tipo entero o de serie de caracteres. Para ello, se necesitan varias sentencias en el programa para cada estructura y debe asignarse espacio de memoria. Esta tarea puede resultar larga y laboriosa.

Los programas escritos utilizando la MQAI pasan parámetros en el paquete de datos adecuado y tan solo se requiere una sentencia para cada estructura. La utilización de paquetes de datos de MQAI elimina la necesidad de manejar matrices y asignar almacenamiento y proporciona cierto grado de aislamiento de los detalles del PCF.

Manejar las condiciones de error con más facilidad
Es difícil obtener códigos de retorno de los mandatos PCF, pero la MQAI permite al programa manejar más fácilmente las condiciones de error.

Después de haber creado y llenado los paquetes de datos, puede enviar un mensaje de mandato de administración al servidor de mandatos de un gestor de colas, utilizando la llamada `mqExecute`, que espera los mensajes de respuesta. La llamada `mqExecute` maneja el intercambio con el servidor de mandatos y devuelve respuestas en un *paquete de respuestas*.

Para obtener más información sobre la MQAI, consulte “Introducción a la Interfaz de administración de IBM WebSphere MQ (MQAI)” en la página 20.

Introducción a la Interfaz de administración de IBM WebSphere MQ (MQAI)

La Interfaz de administración de IBM WebSphere MQ (MQAI) es una interfaz de programación para IBM WebSphere MQ. Realiza tareas de administración en un gestor de colas de IBM WebSphere MQ mediante paquetes de datos para manejar las propiedades (o parámetros) de objetos de forma que es más fácil que utilizando los formatos de mandato programable (PCF).

Conceptos y terminología de MQAI

La MQAI es una interfaz de programación para WebSphere MQ que utiliza el lenguaje C y también Visual Basic para Windows. Está disponible en plataformas que no sean z/OS.

Realiza tareas de administración en un gestor de colas de WebSphere MQ utilizando paquetes de datos. Los paquetes de datos permiten manejar las propiedades (o los parámetros) de los objetos de una forma que es más fácil que utilizando los otros formatos de mandatos programables (PCF) de interfaz de

administración. Con la MQAI podrá manipular más fácilmente los PCF que utilizando las llamadas MQGET y MQPUT.

Para obtener más información acerca de los paquetes de datos, consulte “Paquetes de datos” en la página 47. Para obtener más información sobre los PCF, consulte “Introducción a los Formatos de mandato programable” en la página 9

Utilización de la MQAI

Puede usar la MQAI para:

- Simplificar el uso de mensajes PCF. La MQAI es una forma fácil de administrar WebSphere MQ; no es necesario que escriba sus propios mensajes PCF, evitando así los problemas asociados a las estructuras de datos complejas.
- Manejar las condiciones de error con más facilidad. Es difícil obtener códigos de retorno de los mandatos de script de WebSphere MQ (MQSC), pero la MQAI permite al programa manejar más fácilmente las condiciones de error.
- Intercambiar datos entre aplicaciones. Los datos de aplicación se envían en formato PCF y los empaqueta y desempaqueta la MQAI. Si los datos del mensaje constan de enteros y series de caracteres, puede utilizar la MQAI para sacar partido de la conversión de datos incorporada de WebSphere MQ para datos PCF. Esto evita la necesidad de grabar salidas de conversión de datos. Para obtener más información sobre la utilización de MQAI para administrar WebSphere MQ y para intercambiar datos entre aplicaciones, consulte “Utilizar la MQAI para simplificar el uso de los PCF” en la página 20.

Ejemplos de cómo utilizar la MQAI

La lista que se muestra proporciona algunos programas de ejemplo que demuestran el uso de la MQAI. Los ejemplos llevan a cabo las siguientes tareas:

1. Cree una cola local. “Programa C de ejemplo para crear una cola local (amqsaicq.c)” en la página 22
2. Visualizar sucesos en la pantalla utilizando un supervisor de sucesos simple.“Programa C de ejemplo para visualizar sucesos utilizando un supervisor de sucesos (amqsaiem.c)” en la página 25
3. Imprimir una lista de todas las colas locales y su profundidad actual.“Programa C de ejemplo para realizar consultas sobre colas e imprimir información (amqsailq.c)” en la página 38
4. Imprimir una lista de todos los canales y sus tipos.“Programa C de ejemplo para realizar consultas sobre objetos de canal (amqsaicl.c)” en la página 32

Creación de una aplicación MQAI

Para crear la aplicación utilizando la MQAI, el enlace con las mismas bibliotecas tal como lo haría para WebSphere MQ. Para obtener información sobre cómo crear las aplicaciones WebSphere MQ , consulte Creación de una aplicación WebSphere MQ .

Consejos y sugerencias para configurar WebSphere MQ utilizando MQAI

La MQAI utiliza los mensajes PCF para enviar mandatos de administración al servidor de mandatos, en lugar de tratar directamente con el propio servidor de mandatos. Encontrará consejos para configurar WebSphere MQ mediante la MQAI en “Consejos y sugerencias para configurar IBM WebSphere MQ” en la página 42.

Interfaz de administración de IBM WebSphere MQ (MQAI)

IBM WebSphere MQ para Windows, AIX, Linux, HP-UXy Solaris dan soporte a la interfaz de administración de IBM WebSphere MQ (MQAI). La MQAI es una interfaz de programación para IBM WebSphere MQ que ofrece una alternativa a la interfaz MQI, para enviar y recibir PCF.

La interfaz MQAI utiliza *paquetes de datos* que le permiten manejar las propiedades (o parámetros) de los objetos de forma más sencilla que utilizando mandatos PCF directamente por medio de la MQAI.

La MQAI proporciona un acceso más fácil de programación para mensajes PCF pasando parámetros en el paquete de datos, de modo que sólo se requiere una sentencia para cada estructura. Este acceso elimina la necesidad de que el programador maneje matrices y asigne almacenamiento, y proporciona aislamiento de los detalles del PCF.

La MQAI administra WebSphere MQ enviando mensajes PCF al servidor de mandatos y esperando una respuesta.

La MQAI se describe en la segunda sección de este manual. Consulte la documentación de Utilización de Java para obtener una descripción de una interfaz de modelo de objetos componentes para la MQAI.

Programa C de ejemplo para crear una cola local (amqsaicq.c)

El programa C de ejemplo `amqsaicq.c` crea una cola local utilizando MQAI.

```
/********************************************************************************/
/*                                                                              */
/* Program name: AMQSAICQ.C                                                     */
/*                                                                              */
/* Description:  Sample C program to create a local queue using the             */
/*               WebSphere MQ Administration Interface (MQAI).                  */
/*                                                                              */
/* Statement:    Licensed Materials - Property of IBM                           */
/*                                                                              */
/*               84H2000, 5765-B73                                              */
/*               84H2001, 5639-B42                                              */
/*               84H2002, 5765-B74                                              */
/*               84H2003, 5765-B75                                              */
/*               84H2004, 5639-B43                                              */
/*                                                                              */
/*               (C) Copyright IBM Corp. 1999, 2024.                            */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
/* Function:                                                                    */
/*    AMQSAICQ is a sample C program that creates a local queue and is an       */
/*    example of the use of the mqExecute call.                                 */
/*                                                                              */
/*     - The name of the queue to be created is a parameter to the program.     */
/*                                                                              */
/*     - A PCF command is built by placing items into an MQAI bag.              */
/*       These are:-                                                            */
/*            - The name of the queue                                           */
/*            - The type of queue required, which, in this case, is local.      */
/*                                                                              */
/*     - The mqExecute call is executed with the command MQCMD_CREATE_Q.        */
/*       The call generates the correct PCF structure.                          */
/*       The call receives the reply from the command server and formats into   */
/*       the response bag.                                                      */
/*                                                                              */
/*     - The completion code from the mqExecute call is checked and if there    */
/*       is a failure from the command server then the code returned by the     */
/*       command server is retrieved from the system bag that is                */
/*       embedded in the response bag to the mqExecute call.                    */
/*                                                                              */
/* Note: The command server must be running.                                    */
/*                                                                              */
/*                                                                              */

/********************************************************************************/
/*                                                                              */
/* AMQSAICQ has 2 parameters - the name of the local queue to be created        */
/*                           - the queue manager name (optional)                */
/*                                                                              */
/********************************************************************************/
/********************************************************************************/
/* Includes                                                                     */
/********************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
```

```
#include <cmqc.h>                          /* MQI                              */
#include <cmqcfc.h>                        /* PCF                              */
#include <cmqbc.h>                         /* MQAI                             */

void CheckCallResult(MQCHAR *, MQLONG , MQLONG );
void CreateLocalQueue(MQHCONN, MQCHAR *);

int main(int argc, char *argv[])
{
   MQHCONN hConn;                          /* handle to WebSphere MQ connection   */
   MQCHAR QMName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name   */
   MQLONG connReason;                      /* MQCONN reason code               */
   MQLONG compCode;                        /* completion code                  */
   MQLONG reason;                          /* reason code                      */

   /*******************************************************************/
   /* First check the required parameters                             */
   /*******************************************************************/
   printf("Sample Program to Create a Local Queue\n");
   if (argc < 2)
   {
     printf("Required parameter missing - local queue name\n");
     exit(99);
   }

   /*******************************************************************/
   /* Connect to the queue manager                                    */
   /*******************************************************************/
   if (argc > 2)
      strncpy(QMName, argv[2], (size_t)MQ_Q_MGR_NAME_LENGTH);
      MQCONN(QMName, &hConn, &compCode, &connReason);

/*******************************************************************/
/* Report reason and stop if connection failed                     */
/*******************************************************************/
   if (compCode == MQCC_FAILED)
   {
      CheckCallResult("MQCONN", compCode, connReason);
      exit( (int)connReason);
   }


/*******************************************************************/
/* Call the routine to create a local queue, passing the handle to the        */
/* queue manager and also passing the name of the queue to be created.        */
/*******************************************************************/
   CreateLocalQueue(hConn, argv[1]);

   /*******************************************************************/
   /* Disconnect from the queue manager if not already connected      */
   /*******************************************************************/
   if (connReason != MQRC_ALREADY_CONNECTED)
   {
      MQDISC(&hConn, &compCode, &reason);
      CheckCallResult("MQDISC", compCode, reason);
   }
   return 0;

}
/*******************************************************************/
/*                                                                 */
/* Function:    CreateLocalQueue                                   */
/* Description: Create a local queue by sending a PCF command to the command  */
/*              server.                                            */
/*                                                                 */
/*******************************************************************/
/*                                                                 */
/* Input Parameters:  Handle to the queue manager                  */
/*                    Name of the queue to be created              */
/*                                                                 */
/* Output Parameters: None                                         */
/*                                                                 */
/* Logic: The mqExecute call is executed with the command MQCMD_CREATE_Q.     */
/*        The call generates the correct PCF structure.            */
/*        The default options to the call are used so that the command is sent*/
/*        to the SYSTEM.ADMIN.COMMAND.QUEUE.                       */
/*        The reply from the command server is placed on a temporary dynamic  */
/*        queue.                                                   */
/*        The reply is read from the temporary queue and formatted into the   */
/*        response bag.                                            */
/*                                                                 */
/*        The completion code from the mqExecute call is checked and if there */
```

```
/*        is a failure from the command server then the code returned by the  */
/*        command server is retrieved from the system bag that is             */
/*        embedded in the response bag to the mqExecute call.                 */
/*                                                                            */
/******************************************************************************/
void CreateLocalQueue(MQHCONN hConn, MQCHAR *qName)
{
   MQLONG reason;                        /* reason code                      */
   MQLONG compCode;                      /* completion code                  */
   MQHBAG commandBag = MQHB_UNUSABLE_HBAG; /* command bag for mqExecute      */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute     */
   MQHBAG resultBag;                     /* result bag from mqExecute        */
   MQLONG mqExecuteCC;                   /* mqExecute completion code        */
   MQLONG mqExecuteRC;                   /* mqExecute reason code            */

   printf("\nCreating Local Queue %s\n\n", qName);


   /******************************************************************************/
   /* Create a command Bag for the mqExecute call. Exit the function if the   */
   /* create fails.                                                              */
   /******************************************************************************/
   mqCreateBag(MQCBO_ADMIN_BAG, &commandBag, &compCode, &reason);
   CheckCallResult("Create the command bag", compCode, reason);
   if (compCode !=MQCC_OK)
      return;

   /******************************************************************************/
   /* Create a response Bag for the mqExecute call, exit the function if the  */
   /* create fails.                                                              */
   /******************************************************************************/
   mqCreateBag(MQCBO_ADMIN_BAG, &responseBag, &compCode, &reason);
   CheckCallResult("Create the response bag", compCode, reason);
   if (compCode !=MQCC_OK)
      return;

   /******************************************************************************/
   /* Put the name of the queue to be created into the command bag. This will */
   /* be used by the mqExecute call.                                             */
   /******************************************************************************/
   mqAddString(commandBag, MQCA_Q_NAME, MQBL_NULL_TERMINATED, qName, &compCode,
               &reason);
   CheckCallResult("Add q name to command bag", compCode, reason);

   /******************************************************************************/
   /* Put queue type of local into the command bag. This will be used by the  */
   /* mqExecute call.                                                            */
   /******************************************************************************/
   mqAddInteger(commandBag, MQIA_Q_TYPE, MQQT_LOCAL, &compCode, &reason);
   CheckCallResult("Add q type to command bag", compCode, reason);

   /******************************************************************************/
   /* Send the command to create the required local queue.                    */
   /* The mqExecute call will create the PCF structure required, send it to   */
   /* the command server and receive the reply from the command server into   */
   /* the response bag.                                                          */
   /******************************************************************************/
   mqExecute(hConn,                    /* WebSphere MQ connection handle          */
             MQCMD_CREATE_Q,           /* Command to be executed               */
             MQHB_NONE,                /* No options bag                       */
             commandBag,               /* Handle to bag containing commands    */
             responseBag,              /* Handle to bag to receive the response*/
             MQHO_NONE,                /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
             MQHO_NONE,                /* Create a dynamic q for the response  */
             &compCode,                /* Completion code from the mqExecute   */
             &reason);                 /* Reason code from mqExecute call      */

   if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
   {
      printf("Please start the command server: <strmqcsv QMgrName>\n")
      MQDISC(&hConn, &compCode, &reason);
      CheckCallResult("MQDISC", compCode, reason);
      exit(98);
   }


   /******************************************************************************/
   /* Check the result from mqExecute call and find the error if it failed.   */
   /******************************************************************************/
   if ( compCode == MQCC_OK )
      printf("Local queue %s successfully created\n", qName);
   else
```

```
   {
      printf("Creation of local queue %s failed: Completion Code = %d
             qName, compCode, reason);
      if (reason == MQRCCF_COMMAND_FAILED)
      {
         /*******************************************************************/
         /* Get the system bag handle out of the mqExecute response bag.    */
         /* This bag contains the reason from the command server why the    */
         /* command failed.                                                 */
         /*******************************************************************/
         mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &resultBag, &compCode,
                      &reason);
         CheckCallResult("Get the result bag handle", compCode, reason);

         /*******************************************************************/
         /* Get the completion code and reason code, returned by the command  */
         /* server, from the embedded error bag.                            */
         /*******************************************************************/
         mqInquireInteger(resultBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                          &compCode, &reason);
         CheckCallResult("Get the completion code from the result bag",
                          compCode, reason);
         mqInquireInteger(resultBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                          &compCode, &reason);
         CheckCallResult("Get the reason code from the result bag", compCode,
                          reason);
         printf("Error returned by the command server: Completion code = %d :
                 Reason = %d\n", mqExecuteCC, mqExecuteRC);
      }
   }
   /*********************************************************************/
   /* Delete the command bag if successfully created.                   */
   /*********************************************************************/
   if (commandBag != MQHB_UNUSABLE_HBAG)
   {
      mqDeleteBag(&commandBag, &compCode, &reason);
      CheckCallResult("Delete the command bag", compCode, reason);
   }

   /*********************************************************************/
   /* Delete the response bag if successfully created.                  */
   /*********************************************************************/
   if (responseBag != MQHB_UNUSABLE_HBAG)
   {
      mqDeleteBag(&responseBag, &compCode, &reason);
      CheckCallResult("Delete the response bag", compCode, reason);
   }
} /* end of CreateLocalQueue */


/*****************************************************************************/
/*                                                                           */
/* Function: CheckCallResult                                                 */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Input Parameters:  Description of call                                    */
/*                    Completion code                                        */
/*                    Reason code                                            */
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Logic: Display the description of the call, the completion code and the   */
/*        reason code if the completion code is not successful               */
/*                                                                           */
/*****************************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
   if (cc != MQCC_OK)
         printf("%s failed: Completion Code = %d :
                 Reason = %d\n", callText, cc, rc);

}
```

Programa C de ejemplo para visualizar sucesos utilizando un supervisor de sucesos (amqsaiem.c)

El programa C de ejemplo `amqsaiem.c` muestra un supervisor de sucesos básico utilizando MQAI.

```
/*****************************************************************************/
/*                                                                           */
/* Program name: AMQSAIEM.C                                                  */
/*                                                                           */
/* Description:  Sample C program to demonstrate a basic event monitor       */
/*               using the WebSphere MQ Admin Interface (MQAI).              */
/* Licensed Materials - Property of IBM                                      */
/*                                                                           */
/* 63H9336                                                                   */
/* (c) Copyright IBM Corp. 1999, 2024. All Rights Reserved.                  */
/*                                                                           */
/* US Government Users Restricted Rights - Use, duplication or               */
/* disclosure restricted by GSA ADP Schedule Contract with                   */
/* IBM Corp.                                                                 */
/*****************************************************************************/
/*                                                                           */
/* Function:                                                                 */
/*    AMQSAIEM is a sample C program that demonstrates how to write a simple */
/*    event monitor using the mqGetBag call and other MQAI calls.            */
/*                                                                           */
/*    The name of the event queue to be monitored is passed as a parameter   */
/*    to the program. This would usually be one of the system event queues:- */
/*            SYSTEM.ADMIN.QMGR.EVENT        Queue Manager events            */
/*            SYSTEM.ADMIN.PERFM.EVENT       Performance events              */
/*            SYSTEM.ADMIN.CHANNEL.EVENT     Channel events                  */
/*            SYSTEM.ADMIN.LOGGER.EVENT      Logger events                   */
/*                                                                           */
/*    To monitor the queue manager event queue or the performance event queue,*/
/*    the attributes of the queue manager needs to be changed to enable      */
/*    these events. For more information about this, see Part 1 of the       */
/*    Programmable System Management book. The queue manager attributes can  */
/*    be changed using either MQSC commands or the MQAI interface.           */
/*    Channel events are enabled by default.                                 */
/*                                                                           */
/* Program logic                                                             */
/*    Connect to the Queue Manager.                                          */
/*    Open the requested event queue with a wait interval of 30 seconds.     */
/*    Wait for a message, and when it arrives get the message from the queue */
/*    and format it into an MQAI bag using the mqGetBag call.                */
/*    There are many types of event messages and it is beyond the scope of   */
/*    this sample to program for all event messages. Instead the program     */
/*    prints out the contents of the formatted bag.                          */
/*    Loop around to wait for another message until either there is an error */
/*    or the wait interval of 30 seconds is reached.                         */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* AMQSAIEM has 2 parameters - the name of the event queue to be monitored   */
/*                           - the queue manager name (optional)             */
/*                                                                           */
/*****************************************************************************

/*****************************************************************************/
/* Includes                                                                  */
/*****************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#include <cmqc.h>                       /* MQI                               */
#include <cmqcfc.h>                     /* PCF                               */
#include <cmqbc.h>                      /* MQAI                              */

/*****************************************************************************/
/* Macros                                                                    */
/*****************************************************************************/
#if MQAT_DEFAULT == MQAT_WINDOWS_NT
  #define Int64 "I64"
#elif defined(MQ_64_BIT)
  #define Int64 "l"
#else
  #define Int64 "ll"
#endif

/*****************************************************************************/
/* Function prototypes                                                       */
/*****************************************************************************/
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);
void GetQEvents(MQHCONN, MQCHAR *);
```

```
int PrintBag(MQHBAG);
int PrintBagContents(MQHBAG, int);

/*****************************************************************************/
/* Function: main                                                            */
/*****************************************************************************/
int main(int argc, char *argv[])
{
   MQHCONN hConn;                          /* handle to connection           */
   MQCHAR QMName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QM name              */
   MQLONG reason;                          /* reason code                    */
   MQLONG connReason;                      /* MQCONN reason code             */
   MQLONG compCode;                        /* completion code                */

   /*****************************************************************************/
   /* First check the required parameters                                       */
   /*****************************************************************************/
   printf("Sample Event Monitor (times out after 30 secs)\n");
   if (argc < 2)
   {
     printf("Required parameter missing - event queue to be monitored\n");
     exit(99);
   }

   /*****************************************************************************/
   /* Connect to the queue manager                                              */
   /*****************************************************************************/
   if (argc > 2)
     strncpy(QMName, argv[2], (size_t)MQ_Q_MGR_NAME_LENGTH);
   MQCONN(QMName, &hConn, &compCode, &connReason);
   /*****************************************************************************/
   /* Report the reason and stop if the connection failed                       */
   /*****************************************************************************/
   if (compCode == MQCC_FAILED)
   {
      CheckCallResult("MQCONN", compCode, connReason);
      exit( (int)connReason);
   }

   /*****************************************************************************/
   /* Call the routine to open the event queue and format any event messages  */
   /* read from the queue.                                                      */
   /*****************************************************************************/
   GetQEvents(hConn, argv[1]);

   /*****************************************************************************/
   /* Disconnect from the queue manager if not already connected                */
   /*****************************************************************************/
   if (connReason != MQRC_ALREADY_CONNECTED)
   {
      MQDISC(&hConn, &compCode, &reason);
      CheckCallResult("MQDISC", compCode, reason);
   }

   return 0;

}

/*****************************************************************************/
/*                                                                           */
/* Function: CheckCallResult                                                 */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Input Parameters:  Description of call                                    */
/*                    Completion code                                        */
/*                    Reason code                                            */
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Logic: Display the description of the call, the completion code and the   */
/*        reason code if the completion code is not successful               */
/*                                                                           */
/*****************************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
   if (cc != MQCC_OK)
         printf("%s failed: Completion Code = %d : Reason = %d\n",
                 callText, cc, rc);

}
```

```
/*****************************************************************************/
/*                                                                           */
/* Function: GetQEvents                                                      */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Input Parameters:  Handle to the queue manager                            */
/*                    Name of the event queue to be monitored                */
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Logic:   Open the event queue.                                            */
/*          Get a message off the event queue and format the message into    */
/*          a bag.                                                           */
/*          A real event monitor would need to be programmed to deal with    */
/*          each type of event that it receives from the queue. This is      */
/*          outside the scope of this sample, so instead, the contents of    */
/*          the bag are printed.                                             */
/*          The program waits for 30 seconds for an event message and then   */
/*          terminates if no more messages are available.                    */
/*                                                                           */
/*****************************************************************************/
void GetQEvents(MQHCONN hConn, MQCHAR *qName)
{
   MQLONG openReason;                      /* MQOPEN reason code             */
   MQLONG reason;                          /* reason code                    */
   MQLONG compCode;                        /* completion code                */
   MQHOBJ eventQueue;                      /* handle to event queue          */

   MQHBAG eventBag = MQHB_UNUSABLE_HBAG;   /* event bag to receive event msg */
   MQOD   od = {MQOD_DEFAULT};             /* Object Descriptor              */
   MQMD   md = {MQMD_DEFAULT};             /* Message Descriptor             */
   MQGMO  gmo = {MQGMO_DEFAULT};           /* get message options            */
   MQLONG bQueueOK = 1;                    /* keep reading msgs while true   */

   /***************************************************************************/
   /* Create an Event Bag in which to receive the event.                      */
   /* Exit the function if the create fails.                                  */
   /***************************************************************************/
   mqCreateBag(MQCBO_USER_BAG, &eventBag, &compCode, &reason);
   CheckCallResult("Create event bag", compCode, reason);
   if (compCode !=MQCC_OK)
      return;

   /***************************************************************************/
   /* Open the event queue chosen by the user                                 */
   /***************************************************************************/
   strncpy(od.ObjectName, qName, (size_t)MQ_Q_NAME_LENGTH);
   MQOPEN(hConn, &od, MQOO_INPUT_AS_Q_DEF+MQOO_FAIL_IF_QUIESCING, &eventQueue,
          &compCode, &openReason);
   CheckCallResult("Open event queue", compCode, openReason);

   /***************************************************************************/
   /* Set the GMO options to control the action of the get message from the   */
   /* queue.                                                                  */
   /***************************************************************************/
   gmo.WaitInterval = 30000;               /* 30 second wait for message      */
   gmo.Options = MQGMO_WAIT + MQGMO_FAIL_IF_QUIESCING + MQGMO_CONVERT;
   gmo.Version = MQGMO_VERSION_2;          /* Avoid need to reset Message ID  */
   gmo.MatchOptions = MQMO_NONE;           /* and Correlation ID after every  */
                                 /* mqGetBag
   /***************************************************************************/
   /* If open fails, we cannot access the queue and must stop the monitor.    */
   /***************************************************************************/
   if (compCode != MQCC_OK)
      bQueueOK = 0;

   /***************************************************************************/
   /* Main loop to get an event message when it arrives                       */
   /***************************************************************************/
   while (bQueueOK)
   {
     printf("\nWaiting for an event\n");

     /*************************************************************************/
     /* Get the message from the event queue and convert it into the event    */
     /* bag.                                                                  */
     /*************************************************************************/
     mqGetBag(hConn, eventQueue, &md, &gmo, eventBag, &compCode, &reason);

     /*************************************************************************/
     /* If get fails, we cannot access the queue and must stop the monitor.   */
```

```
     /*********************************************************************/
     if (compCode != MQCC_OK)
     {
        bQueueOK = 0;

        /*****************************************************************/
        /* If get fails because no message available then we have timed out, */
        /* so report this, otherwise report an error.                    */
        /*****************************************************************/
        if (reason == MQRC_NO_MSG_AVAILABLE)
        {
           printf("No more messages\n");
        }
        else
        {
           CheckCallResult("Get bag", compCode, reason);
        }
     }

     /*********************************************************************/
     /* Event message read - Print the contents of the event bag          */
     /*********************************************************************/
     else
     {
       if ( PrintBag(eventBag) )
           printf("\nError found while printing bag contents\n");

     }  /* end of msg found */
   } /* end of main loop */
   /*********************************************************************/
   /* Close the event queue if successfully opened                      */
   /*********************************************************************/
   if (openReason == MQRC_NONE)
   {
      MQCLOSE(hConn, &eventQueue, MQCO_NONE, &compCode, &reason);
      CheckCallResult("Close event queue", compCode, reason);
   }

   /*********************************************************************/
   /* Delete the event bag if successfully created.                     */
   /*********************************************************************/
   if (eventBag != MQHB_UNUSABLE_HBAG)
   {
      mqDeleteBag(&eventBag, &compCode, &reason);
      CheckCallResult("Delete the event bag", compCode, reason);
   }

} /* end of GetQEvents */

/*****************************************************************************/
/*                                                                           */
/* Function: PrintBag                                                        */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* Input Parameters:  Bag Handle                                             */
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Returns:           Number of errors found                                 */
/*                                                                           */
/* Logic: Calls PrintBagContents to display the contents of the bag.         */
/*                                                                           */
/*****************************************************************************

int PrintBag(MQHBAG dataBag)
{
    int errors;

    printf("\n");
    errors = PrintBagContents(dataBag, 0);
    printf("\n");

    return errors;
}

/*****************************************************************************/
/*                                                                           */
/* Function: PrintBagContents                                                */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
```

```
/* Input Parameters:  Bag Handle                                          */
/*                    Indentation level of bag                            */
/*                                                                        */
/* Output Parameters: None                                                */
/*                                                                        */
/* Returns:           Number of errors found                              */
/*                                                                        */
/* Logic: Count the number of items in the bag                            */
/*        Obtain selector and item type for each item in the bag.         */
/*        Obtain the value of the item depending on item type and display the */
/*        index of the item, the selector and the value.                  */
/*        If the item is an embedded bag handle then call this function again */
/*        to print the contents of the embedded bag increasing the        */
/*        indentation level.                                              */
/*                                                                        */
/**************************************************************************/
int PrintBagContents(MQHBAG dataBag, int indent)
{
   /************************************************************************/
   /* Definitions                                                          */
   /************************************************************************/
   #define LENGTH 500                      /* Max length of string to be read*/
   #define INDENT 4                        /* Number of spaces to indent     */
                                           /* embedded bag display           */


   /************************************************************************/
   /* Variables                                                            */
   /************************************************************************/
   MQLONG  itemCount;                      /* Number of items in the bag     */
   MQLONG  itemType;                       /* Type of the item               */
   int     i;                              /* Index of item in the bag       */
   MQCHAR  stringVal[LENGTH+1];            /* Value if item is a string      */
   MQBYTE  byteStringVal[LENGTH];          /* Value if item is a byte string */
   MQLONG  stringLength;                   /* Length of string value         */
   MQLONG  ccsid;                          /* CCSID of string value          */
   MQINT32 iValue;                         /* Value if item is an integer    */
   MQINT64 i64Value;                       /* Value if item is a 64-bit      */
                                           /* integer                        */
   MQLONG  selector;                       /* Selector of item               */
   MQHBAG  bagHandle;                      /* Value if item is a bag handle  */
   MQLONG  reason;                         /* reason code                    */
   MQLONG  compCode;                       /* completion code                */
   MQLONG  trimLength;                     /* Length of string to be trimmed */
   int     errors = 0;                     /* Count of errors found          */
   char    blanks[] = "                      "; /* Blank string used to      */
                                                /* indent display            */

   /************************************************************************/
   /* Count the number of items in the bag                                 */
   /************************************************************************/
   mqCountItems(dataBag, MQSEL_ALL_SELECTORS, &itemCount, &compCode, &reason);

   if (compCode != MQCC_OK)
      errors++;
   else
   {
      printf("
      printf("
      printf("
   }

   /************************************************************************/
   /* If no errors found, display each item in the bag                     */
   /************************************************************************/
   if (!errors)
   {
      for (i = 0; i < itemCount; i++)
      {

          /*****************************************************************/
          /* First inquire the type of the item for each item in the bag   */
          /*****************************************************************/
          mqInquireItemInfo(dataBag,             /* Bag handle             */
                            MQSEL_ANY_SELECTOR,  /* Item can have any selector*/
                            i,                   /* Index position in the bag */
                            &selector,           /* Actual value of selector  */
                                                 /* returned by call          */
                            &itemType,           /* Actual type of item       */
                                                 /* returned by call          */
                            &compCode,           /* Completion code           */
                            &reason);            /* Reason Code               */
```

```
 if (compCode != MQCC_OK)
    errors++;

 switch(itemType)
 {
 case MQITEM_INTEGER:
      /*****************************************************************/
      /* Item is an integer. Find its value and display its index,   */
      /* selector and value.                                           */
      /*****************************************************************/
      mqInquireInteger(dataBag,         /* Bag handle                  */
                       MQSEL_ANY_SELECTOR, /* Allow any selector     */
                       i,               /* Index position in the bag */
                       &iValue,         /* Returned integer value
                       &compCode,       /* Completion code           */
                       &reason);        /* Reason Code               */

      if (compCode != MQCC_OK)
         errors++;
      else
         printf("%.*s  %-2d     %-4d     (%d)\n",
                 indent, blanks, i, selector, iValue);
      break

 case MQITEM_INTEGER64:
      /*****************************************************************/
      /* Item is a 64-bit integer. Find its value and display its    */
      /* index, selector and value.                                    */
      /*****************************************************************/
      mqInquireInteger64(dataBag,       /* Bag handle                  */
                         MQSEL_ANY_SELECTOR, /* Allow any selector   */
                         i,             /* Index position in the bag */
                         &i64Value,     /* Returned integer value    */
                         &compCode,     /* Completion code           */
                         &reason);      /* Reason Code               */

      if (compCode != MQCC_OK)
         errors++;
      else
         printf("%.*s  %-2d     %-4d     (%"Int64"d)\n",
                 indent, blanks, i, selector, i64Value);
      break;


case MQITEM_STRING:
      /*****************************************************************/
      /* Item is a string. Obtain the string in a buffer, prepare    */
      /* the string for displaying and display the index, selector,  */
      /* string and Character Set ID.                                  */
      /*****************************************************************/
      mqInquireString(dataBag,          /* Bag handle                  */
                      MQSEL_ANY_SELECTOR, /* Allow any selector      */
                      i,                /* Index position in the bag */
                      LENGTH,           /* Maximum length of buffer  */
                      stringVal,        /* Buffer to receive string  */
                      &stringLength,    /* Actual length of string   */
                      &ccsid,           /* Coded character set id    */
                      &compCode,        /* Completion code           */
                      &reason);         /* Reason Code               */

      /*****************************************************************/
      /* The call can return a warning if the string is too long for */
      /* the output buffer and has been truncated, so only check     */
      /* explicitly for call failure.                                  */
      /*****************************************************************/
      if (compCode == MQCC_FAILED)
          errors++;
      else
      {
         /**************************************************************/
         /* Remove trailing blanks from the string and terminate with*/
         /* a null. First check that the string should not have been */
         /* longer than the maximum buffer size allowed.              */
         /**************************************************************/
         if (stringLength > LENGTH)
            trimLength = LENGTH;
         else
            trimLength = stringLength;
         mqTrim(trimLength, stringVal, stringVal, &compCode, &reason);
         printf("%.*s  %-2d     %-4d     '%s' %d\n",
                 indent, blanks, i, selector, stringVal, ccsid);
```

```
                }
                break;

           case MQITEM_BYTE_STRING:
                /*****************************************************************/
                /* Item is a byte string. Obtain the byte string in a buffer,  */
                /* prepare the byte string for displaying and display the      */
                /* index, selector and string.                                   */
                /*****************************************************************/
                mqInquireByteString(dataBag,      /* Bag handle                */
                                    MQSEL_ANY_SELECTOR, /* Allow any selector  */
                                    i,            /* Index position in the bag */
                                    LENGTH,       /* Maximum length of buffer  */
                                    byteStringVal, /* Buffer to receive string */
                                    &stringLength, /* Actual length of string  */
                                    &compCode,    /* Completion code           */
                                    &reason);     /* Reason Code

                /*****************************************************************/
                /* The call can return a warning if the string is too long for */
                /* the output buffer and has been truncated, so only check     */
                /* explicitly for call failure.                                  */
                /*****************************************************************/
                if (compCode == MQCC_FAILED)
                    errors++;
                else
                {
                   printf("%.*s  %-2d     %-4d     X'",
                          indent, blanks, i, selector);

                   for (i = 0 ; i < stringLength ; i++)
                      printf("

                   printf("'\n");
                }
                break;

           case MQITEM_BAG:
                /*****************************************************************/
                /* Item is an embedded bag handle, so call the PrintBagContents*/
                /* function again to display the contents.                       */
                /*****************************************************************/
                mqInquireBag(dataBag,             /* Bag handle                */
                             MQSEL_ANY_SELECTOR,  /* Allow any selector        */
                             i,                   /* Index position in the bag */
                             &bagHandle,          /* Returned embedded bag hdle*/
                             &compCode,           /* Completion code           */
                             &reason);            /* Reason Code               */

                if (compCode != MQCC_OK)
                   errors++;
                else
                {
                   printf("%.*s  %-2d     %-4d     (%d)\n", indent, blanks, i,
                             selector, bagHandle);
                   if (selector == MQHA_BAG_HANDLE)
                      printf("
                   else
                      printf("
                   PrintBagContents(bagHandle, indent+INDENT);
                }
                break;

           default:
                printf("
           }
      }
   }
   return errors;
}
```

Programa C de ejemplo para realizar consultas sobre objetos de canal (amqsaicl.c)

El programa C de ejemplo `amqsaicl.c` consulta los objetos de canal utilizando la MQAI.

```
 /*****************************************************************************/
/*                                                                           */
```

```
/* Program name: AMQSAICL.C                                                  */
/*                                                                           */
/* Description:  Sample C program to inquire channel objects                 */
/*               using the WebSphere MQ Administration Interface (MQAI)      */
/*                                                                           */
/* <N_OCO_COPYRIGHT>                                                         */
/* Licensed Materials - Property of IBM                                      */
/*                                                                           */
/* 63H9336                                                                   */
/* (c) Copyright IBM Corp. 2008, 2024. All Rights Reserved.                  */
/*                                                                           */
/* US Government Users Restricted Rights - Use, duplication or               */
/* disclosure restricted by GSA ADP Schedule Contract with                   */
/* IBM Corp.                                                                 */
/* <NOC_COPYRIGHT>                                                           */
/*****************************************************************************/
/*                                                                           */
/* Function:                                                                 */
/*    AMQSAICL is a sample C program that demonstrates how to inquire        */
/*    attributes of the local queue manager using the MQAI interface. In     */
/*    particular, it inquires all channels and their types.                  */
/*                                                                           */
/*     - A PCF command is built from items placed into an MQAI administration */
/*       bag.                                                                */
/*       These are:-                                                         */
/*            - The generic channel name "*"                                 */
/*            - The attributes to be inquired. In this sample we just want   */
/*              name and type attributes                                     */
/*                                                                           */
/*     - The mqExecute MQCMD_INQUIRE_CHANNEL call is executed.               */
/*       The call generates the correct PCF structure.                       */
/*       The default options to the call are used so that the command is sent */
/*       to the SYSTEM.ADMIN.COMMAND.QUEUE.                                  */
/*       The reply from the command server is placed on a temporary dynamic  */
/*       queue.                                                              */
/*       The reply from the MQCMD_INQUIRE_CHANNEL is read from the           */
/*       temporary queue and formatted into the response bag.                */
/*                                                                           */
/*     - The completion code from the mqExecute call is checked and if there */
/*       is a failure from the command server, then the code returned by the */
/*       command server is retrieved from the system bag that has been       */
/*       embedded in the response bag to the mqExecute call.                 */
/*                                                                           */
/* Note: The command server must be running.                                 */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* AMQSAICL has 2 parameter - the queue manager name (optional)              */
/*                          - output file (optional) default varies          */
/*****************************************************************************/

/*****************************************************************************/
/* Includes                                                                  */
/*****************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#if (MQAT_DEFAULT == MQAT_OS400)
#include <recio.h>
#endif

#include <cmqc.h>                       /* MQI                               */
#include <cmqcfc.h>                     /* PCF                               */
#include <cmqbc.h>                      /* MQAI                              */
#include <cmqxc.h>                      /* MQCD                              */

/*****************************************************************************/
/* Function prototypes                                                       */
/*****************************************************************************/
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);

/*****************************************************************************/
/* DataTypes                                                                 */
/*****************************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
typedef _RFILE OUTFILEHDL;
#else
typedef FILE OUTFILEHDL;
#endif

/*****************************************************************************/
```

```
/* Constants                                                                 */
/*****************************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
const struct
{
  char name[9];
} ChlTypeMap[9] =
{
  "*SDR     ",    /* MQCHT_SENDER    */
  "*SVR     ",    /* MQCHT_SERVER    */
  "*RCVR    ",    /* MQCHT_RECEIVER  */
  "*RQSTR   ",    /* MQCHT_REQUESTER */
  "*ALL     ",    /* MQCHT_ALL       */
  "*CLTCN   ",    /* MQCHT_CLNTCONN  */
  "*SVRCONN ",    /* MQCHT_SVRCONN   */
  "*CLUSRCVR",    /* MQCHT_CLUSRCVR  */
  "*CLUSSDR "     /* MQCHT_CLUSSDR   */
};
#else
const struct
{
  char name[9];
} ChlTypeMap[9] =
{
  "sdr      ",    /* MQCHT_SENDER    */
  "svr      ",    /* MQCHT_SERVER    */
  "rcvr     ",    /* MQCHT_RECEIVER  */
  "rqstr    ",    /* MQCHT_REQUESTER */
  "all      ",    /* MQCHT_ALL       */
  "cltconn  ",    /* MQCHT_CLNTCONN  */
  "svrcn    ",    /* MQCHT_SVRCONN   */
  "clusrcvr ",    /* MQCHT_CLUSRCVR  */
  "clussdr  "     /* MQCHT_CLUSSDR   */
};
#endif

/*****************************************************************************/
/* Macros                                                                    */
/*****************************************************************************/
#if (MQAT_DEFAULT == MQAT_OS400)
  #define OUTFILE "QTEMP/AMQSAICL(AMQSAICL)"
  #define OPENOUTFILE(hdl, fname) \
    (hdl) = _Ropen((fname),"wr, rtncode=Y");
  #define CLOSEOUTFILE(hdl) \
    _Rclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    _Rwrite((hdl),(buf),(buflen));

#elif (MQAT_DEFAULT == MQAT_UNIX)
  #define OUTFILE "/tmp/amqsaicl.txt"
  #define OPENOUTFILE(hdl, fname) \
    (hdl) = fopen((fname),"w");
  #define CLOSEOUTFILE(hdl) \
    fclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    fwrite((buf),(buflen),1,(hdl)); fflush((hdl));

#else
  #define OUTFILE "amqsaicl.txt"
  #define OPENOUTFILE(fname) \
    fopen((fname),"w");
  #define CLOSEOUTFILE(hdl) \
    fclose((hdl));
  #define WRITEOUTFILE(hdl, buf, buflen) \
    fwrite((buf),(buflen),1,(hdl)); fflush((hdl));

#endif

#define ChlType2String(t) ChlTypeMap[(t)-1].name

/*****************************************************************************/
/* Function: main                                                            */
/*****************************************************************************/
int main(int argc, char *argv[])
{
   /**************************************************************************/
   /* MQAI variables                                                         */
   /**************************************************************************/
   MQHCONN hConn;                          /* handle to MQ connection       */
   MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name           */
   MQLONG reason;                          /* reason code                   */
   MQLONG connReason;                      /* MQCONN reason code            */
```

```
MQLONG compCode;                        /* completion code                 */
MQHBAG adminBag = MQHB_UNUSABLE_HBAG;   /* admin bag for mqExecute         */
MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute      */
MQHBAG cAttrsBag;                       /* bag containing chl attributes   */
MQHBAG errorBag;                        /* bag containing cmd server error */
MQLONG mqExecuteCC;                     /* mqExecute completion code       */
MQLONG mqExecuteRC;                     /* mqExecute reason code           */
MQLONG chlNameLength;                   /* Actual length of chl name       */
MQLONG chlType;                         /* Channel type                    */
MQLONG i;                               /* loop counter                    */
MQLONG numberOfBags;                    /* number of bags in response bag  */
MQCHAR chlName[MQ_OBJECT_NAME_LENGTH+1];/* name of chl extracted from bag  */
MQCHAR OutputBuffer[100];               /* output data buffer              */
OUTFILEHDL *outfp = NULL;               /* output file handle              */

/*************************************************************************/
/* Connect to the queue manager                                          */
/*************************************************************************/
if (argc &gt; 1)
   strncpy(qmName, argv[1], (size_t)MQ_Q_MGR_NAME_LENGTH);
MQCONN(qmName, &hConn;, &compCode;, &connReason;);

/*************************************************************************/
/* Report the reason and stop if the connection failed.                  */
/*************************************************************************/
if (compCode == MQCC_FAILED)
{
   CheckCallResult("Queue Manager connection", compCode, connReason);
   exit( (int)connReason);
}

/*************************************************************************/
/* Open the output file                                                  */
/*************************************************************************/
if (argc &gt; 2)
{
  OPENOUTFILE(outfp, argv[2]);
}
else
{
  OPENOUTFILE(outfp, OUTFILE);
}

if(outfp == NULL)
{
  printf("Could not open output file.\n");
  goto MOD_EXIT;
}
/*************************************************************************/
/* Create an admin bag for the mqExecute call                            */
/*************************************************************************/
mqCreateBag(MQCBO_ADMIN_BAG, &adminBag;, &compCode;, &reason;);
CheckCallResult("Create admin bag", compCode, reason);

/*************************************************************************/
/* Create a response bag for the mqExecute call                          */
/*************************************************************************/
mqCreateBag(MQCBO_ADMIN_BAG, &responseBag;, &compCode;, &reason;);
CheckCallResult("Create response bag", compCode, reason);

/*************************************************************************/
/* Put the generic channel name into the admin bag                       */
/*************************************************************************/
mqAddString(adminBag, MQCACH_CHANNEL_NAME, MQBL_NULL_TERMINATED, "*",
            &compCode;, &reason;);
CheckCallResult("Add channel name", compCode, reason);

/*************************************************************************/
/* Put the channel type into the admin bag                               */
/*************************************************************************/
mqAddInteger(adminBag, MQIACH_CHANNEL_TYPE, MQCHT_ALL, &compCode;, &reason;);
CheckCallResult("Add channel type", compCode, reason);

/*************************************************************************/
/* Add an inquiry for various attributes                                 */
/*************************************************************************/
mqAddInquiry(adminBag, MQIACH_CHANNEL_TYPE, &compCode;, &reason;);
CheckCallResult("Add inquiry", compCode, reason);

/*************************************************************************/
/* Send the command to find all the channel names and channel types.     */
/* The mqExecute call creates the PCF structure required, sends it to    */
```

```
/* the command server, and receives the reply from the command server into */
/* the response bag. The attributes are contained in system bags that are  */
/* embedded in the response bag, one set of attributes per bag.            */
/*************************************************************************/
mqExecute(hConn,                  /* MQ connection handle                */
          MQCMD_INQUIRE_CHANNEL,  /* Command to be executed              */
          MQHB_NONE,              /* No options bag                      */
          adminBag,               /* Handle to bag containing commands   */
          responseBag,            /* Handle to bag to receive the response*/
          MQHO_NONE,              /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
          MQHO_NONE,              /* Create a dynamic q for the response */
          &compCode;,              /* Completion code from the mqexecute  */
          &reason);               /* Reason code from mqexecute call     */

/*************************************************************************/
/* Check the command server is started. If not exit.                     */
/*************************************************************************/
if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
   printf("Please start the command server: <strmqcsv QMgrName="">\n");
   goto MOD_EXIT;
}

/*************************************************************************/
/* Check the result from mqExecute call. If successful find the channel  */
/* types for all the channels. If failed find the error.                 */
/*************************************************************************/
if ( compCode == MQCC_OK )                    /* Successful mqExecute    */
{
  /***********************************************************************/
  /* Count the number of system bags embedded in the response bag from the */
  /* mqExecute call. The attributes for each channel are in separate bags. */
  /***********************************************************************/
  mqCountItems(responseBag, MQHA_BAG_HANDLE, &numberOfBags;,
               &compCode;, &reason;);
  CheckCallResult("Count number of bag handles", compCode, reason);

  for ( i=0; i<numberOfbags; i++)
  {
    /*********************************************************************/
    /* Get the next system bag handle out of the mqExecute response bag.   */
    /* This bag contains the channel attributes                          */
    /*********************************************************************/
    mqInquireBag(responseBag, MQHA_BAG_HANDLE, i, &cAttrsbag,
                 &compCode, &reason);
    CheckCallResult("Get the result bag handle", compCode, reason);

    /*********************************************************************/
    /* Get the channel name out of the channel attributes bag              */
    /*********************************************************************/
    mqInquireString(cAttrsBag, MQCACH_CHANNEL_NAME, 0, MQ_OBJECT_NAME_LENGTH,
                    chlName, &chlNameLength, NULL, &compCode, &reason);
    CheckCallResult("Get channel name", compCode, reason);

    /*********************************************************************/
    /* Get the channel type out of the channel attributes bag              */
    /*********************************************************************/

 mqInquireInteger(cAttrsBag, MQIACH_CHANNEL_TYPE, MQIND_NONE, &chlType,
                   &compCode, &reason);
    CheckCallResult("Get type", compCode, reason);

    /*********************************************************************/
    /* Use mqTrim to prepare the channel name for printing.                */
    /* Print the result.                                                 */
    /*********************************************************************/
    mqTrim(MQ_CHANNEL_NAME_LENGTH, chlName, chlName, &compCode, &reason);
    sprintf(OutputBuffer, "%-20s%-9s", chlName, ChlType2String(chlType));
    WRITEOUTFILE(outfp,OutputBuffer,29)
  }
}

else                                          /* Failed mqExecute     */
{
  printf("Call to get channel attributes failed: Cc = %ld : Rc = %ld\n",
               compCode, reason);
  /***********************************************************************/
  /* If the command fails get the system bag handle out of the mqexecute   */
  /* response bag.This bag contains the reason from the command server     */
  /* why the command failed.                                             */
  /***********************************************************************/
  if (reason == MQRCCF_COMMAND_FAILED)
```

```
     {
       mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag,
                    &compCode, &reason);
       CheckCallResult("Get the result bag handle", compCode, reason);

       /*********************************************************************/
       /* Get the completion code and reason code, returned by the command  */
       /* server, from the embedded error bag.                              */
       /*********************************************************************/
       mqInquireInteger(errorBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                        &compCode, &reason );
       CheckCallResult("Get the completion code from the result bag",
                       compCode, reason);
       mqInquireInteger(errorBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                         &compCode, &reason);
       CheckCallResult("Get the reason code from the result bag",
                       compCode, reason);
       printf("Error returned by the command server: Cc = %ld : Rc = %ld\n",
                mqExecuteCC, mqExecuteRC);
     }
   }

MOD_EXIT:
   /*********************************************************************/
   /* Delete the admin bag if successfully created.                     */
   /*********************************************************************/
   if (adminBag != MQHB_UNUSABLE_HBAG)
   {
      mqDeleteBag(&adminBag, &compCode, &reason);
      CheckCallResult("Delete the admin bag", compCode, reason);
   }

   /*********************************************************************/
   /* Delete the response bag if successfully created.                  */
   /*********************************************************************/
   if (responseBag != MQHB_UNUSABLE_HBAG)
   {
      mqDeleteBag(&responseBag, &compCode, &reason);
      CheckCallResult("Delete the response bag", compCode, reason);
   }

   /*********************************************************************/
   /* Disconnect from the queue manager if not already connected        */
   /*********************************************************************/
   if (connReason != MQRC_ALREADY_CONNECTED)
   {
      MQDISC(&hConn, &compCode, &reason);
       CheckCallResult("Disconnect from Queue Manager", compCode, reason);
   }

   /*********************************************************************/
   /* Close the output file if open                                     */
   /*********************************************************************/
   if(outfp != NULL)
      CLOSEOUTFILE(outfp);

   return 0;
}


/*********************************************************************/
/*                                                                   */
/* Function: CheckCallResult                                         */
/*                                                                   */
/*********************************************************************/
/*                                                                   */
/* Input Parameters:  Description of call                            */
/*                    Completion code                                */
/*                    Reason code                                    */
/*                                                                   */
/* Output Parameters: None                                           */
/*                                                                   */
/* Logic: Display the description of the call, the completion code and the    */
/*        reason code if the completion code is not successful       */
/*                                                                   */
/*********************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
   if (cc != MQCC_OK)
         printf("%s failed: Completion Code = %ld : Reason = %ld\n", callText,
```

```
                cc, rc);
}
```

Programa C de ejemplo para realizar consultas sobre colas e imprimir información (amqsailq.c)

El programa C de ejemplo `amqsailq.c` indaga la profundidad actual de las colas locales utilizando MQAI.

```
/********************************************************************************/
/*                                                                              */
/* Program name: AMQSAILQ.C                                                     */
/*                                                                              */
/* Description:  Sample C program to inquire the current depth of the local     */
/*               queues using the WebSphere MQ Administration Interface (MQAI)  */
/*                                                                              */
/* Statement:    Licensed Materials - Property of IBM                           */
/*                                                                              */
/*               84H2000, 5765-B73                                              */
/*               84H2001, 5639-B42                                              */
/*               84H2002, 5765-B74                                              */
/*               84H2003, 5765-B75                                              */
/*               84H2004, 5639-B43                                              */
/*                                                                              */
/*               (C) Copyright IBM Corp. 1999, 2024.                            */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
/* Function:                                                                    */
/*    AMQSAILQ is a sample C program that demonstrates how to inquire           */
/*    attributes of the local queue manager using the MQAI interface. In        */
/*    particular, it inquires the current depths of all the local queues.       */
/*                                                                              */
/*     - A PCF command is built by placing items into an MQAI administration    */
/*       bag.                                                                   */
/*       These are:-                                                            */
/*            - The generic queue name "*"                                      */
/*            - The type of queue required. In this sample we want to           */
/*              inquire local queues.                                           */
/*            - The attribute to be inquired. In this sample we want the        */
/*              current depths.                                                 */
/*                                                                              */
/*     - The mqExecute call is executed with the command MQCMD_INQUIRE_Q.       */
/*       The call generates the correct PCF structure.                          */
/*       The default options to the call are used so that the command is sent   */
/*       to the SYSTEM.ADMIN.COMMAND.QUEUE.                                     */
/*       The reply from the command server is placed on a temporary dynamic     */
/*       queue.                                                                 */
/*       The reply from the MQCMD_INQUIRE_Q command is read from the            */
/*       temporary queue and formatted into the response bag.                   */
/*                                                                              */
/*     - The completion code from the mqExecute call is checked and if there    */
/*       is a failure from the command server, then the code returned by        */
/*       command server is retrieved from the system bag that has been          */
/*       embedded in the response bag to the mqExecute call.                    */
/*                                                                              */
/*     - If the call is successful, the depth of each local queue is placed     */
/*       in system bags embedded in the response bag of the mqExecute call.     */
/*       The name and depth of each queue is obtained from each of the bags     */
/*       and the result displayed on the screen.                                */
/*                                                                              */
/* Note: The command server must be running.                                    */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
/* AMQSAILQ has 1 parameter - the queue manager name (optional)                 */
/*                                                                              */
/********************************************************************************/

/********************************************************************************/
/* Includes                                                                     */
/********************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#include <cmqc.h>                       /* MQI                                  */
```

```
#include <cmqcfc.h>                     /* PCF                              */
#include <cmqbc.h>                      /* MQAI                             */

/******************************************************************************/
/* Function prototypes                                                        */
/******************************************************************************/
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);

/******************************************************************************/
/* Function: main                                                             */
/******************************************************************************/
int main(int argc, char *argv[])
{
   /*************************************************************************/
   /* MQAI variables                                                        */
   /*************************************************************************/
   MQHCONN hConn;                       /* handle to WebSphere MQ connection   */
   MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name            */
   MQLONG reason;                       /* reason code                      */
   MQLONG connReason;                   /* MQCONN reason code               */
   MQLONG compCode;                     /* completion code                  */
   MQHBAG adminBag = MQHB_UNUSABLE_HBAG;   /* admin bag for mqExecute         */
   MQHBAG responseBag = MQHB_UNUSABLE_HBAG;/* response bag for mqExecute      */
   MQHBAG qAttrsBag;                    /* bag containing q attributes      */
   MQHBAG errorBag;                     /* bag containing cmd server error */
   MQLONG mqExecuteCC;                  /* mqExecute completion code        */
   MQLONG mqExecuteRC;                  /* mqExecute reason code            */
   MQLONG qNameLength;                  /* Actual length of q name          */
   MQLONG qDepth;                       /* depth of queue                   */
   MQLONG i;                            /* loop counter                     */
   MQLONG numberOfBags;                 /* number of bags in response bag  */
   MQCHAR qName[MQ_Q_NAME_LENGTH+1];    /* name of queue extracted from bag*/


   printf("Display current depths of local queues\n\n");

   /*************************************************************************/
   /* Connect to the queue manager                                          */
   /*************************************************************************/
   if (argc > 1)
      strncpy(qmName, argv[1], (size_t)MQ_Q_MGR_NAME_LENGTH);
   MQCONN(qmName, &hConn, &compCode, &connReason);

   /*************************************************************************/
   /* Report the reason and stop if the connection failed.                  */
   /*************************************************************************/
   if (compCode == MQCC_FAILED)
   {
      CheckCallResult("Queue Manager connection", compCode, connReason
);
      exit( (int)connReason);
   }

   /*************************************************************************/
   /* Create an admin bag for the mqExecute call                            */
   /*************************************************************************/
   mqCreateBag(MQCBO_ADMIN_BAG, &adminBag, &compCode, &reason);
   CheckCallResult("Create admin bag", compCode, reason);
   /*************************************************************************/
   /* Create a response bag for the mqExecute call                          */
   /*************************************************************************/
   mqCreateBag(MQCBO_ADMIN_BAG, &responseBag, &compCode, &reason);
   CheckCallResult("Create response bag", compCode, reason);

   /*************************************************************************/
   /* Put the generic queue name into the admin bag                         */
   /*************************************************************************/
   mqAddString(adminBag, MQCA_Q_NAME, MQBL_NULL_TERMINATED, "*",
               &compCode, &reason);
   CheckCallResult("Add q name", compCode, reason);

   /*************************************************************************/
   /* Put the local queue type into the admin bag                           */
   /*************************************************************************/
   mqAddInteger(adminBag, MQIA_Q_TYPE, MQQT_LOCAL, &compCode, &reason);
   CheckCallResult("Add q type", compCode, reason);

   /*************************************************************************/
   /* Add an inquiry for current queue depths                               */
   /*************************************************************************/
   mqAddInquiry(adminBag, MQIA_CURRENT_Q_DEPTH, &compCode, &reason);
   CheckCallResult("Add inquiry", compCode, reason);
```

```
/*****************************************************************/
/* Send the command to find all the local queue names and queue depths.    */
/* The mqExecute call creates the PCF structure required, sends it to      */
/* the command server, and receives the reply from the command server into */
/* the response bag. The attributes are contained in system bags that are  */
/* embedded in the response bag, one set of attributes per bag.            */
/*****************************************************************/
mqExecute(hConn,                  /* WebSphere MQ connection handle          */
          MQCMD_INQUIRE_Q,        /* Command to be executed                  */
          MQHB_NONE,              /* No options bag                          */
          adminBag,               /* Handle to bag containing commands       */
          responseBag,            /* Handle to bag to receive the response*/
          MQHO_NONE,              /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
          MQHO_NONE,              /* Create a dynamic q for the response  */
          &compCode,              /* Completion code from the mqExecute   */
          &reason);               /* Reason code from mqExecute call      */


/*****************************************************************/
/* Check the command server is started. If not exit.                       */
/*****************************************************************/
if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
   printf("Please start the command server: <strmqcsv QMgrName>\n");
   MQDISC(&hConn, &compCode, &reason);
   CheckCallResult("Disconnect from Queue Manager", compCode, reason);
   exit(98);
}

/*****************************************************************/
/* Check the result from mqExecute call. If successful find the current    */
/* depths of all the local queues. If failed find the error.               */
/*****************************************************************/
if ( compCode == MQCC_OK )                      /* Successful mqExecute    */
{
  /***************************************************************/
  /* Count the number of system bags embedded in the response bag from the */
  /* mqExecute call. The attributes for each queue are in a separate bag.  */
  /***************************************************************/
  mqCountItems(responseBag, MQHA_BAG_HANDLE, &numberOfBags, &compCode,
               &reason);
  CheckCallResult("Count number of bag handles", compCode, reason);

  for ( i=0; i<numberOfBags; i++)
  {
    /*************************************************************/
    /* Get the next system bag handle out of the mqExecute response bag.   */
    /* This bag contains the queue attributes                              */
    /*************************************************************/
    mqInquireBag(responseBag, MQHA_BAG_HANDLE, i, &qAttrsBag, &compCode,
                 &reason);
    CheckCallResult("Get the result bag handle", compCode, reason);

    /*************************************************************/
    /* Get the queue name out of the queue attributes bag                  */
    /*************************************************************/
    mqInquireString(qAttrsBag, MQCA_Q_NAME, 0, MQ_Q_NAME_LENGTH, qName,
                    &qNameLength, NULL, &compCode, &reason);
    CheckCallResult("Get queue name", compCode, reason);

    /*************************************************************/
    /* Get the depth out of the queue attributes bag                       */
    /*************************************************************/
    mqInquireInteger(qAttrsBag, MQIA_CURRENT_Q_DEPTH, MQIND_NONE, &qDepth,
                     &compCode, &reason);
    CheckCallResult("Get depth", compCode, reason);

    /*************************************************************/
    /* Use mqTrim to prepare the queue name for printing.                  */
    /* Print the result.                                                   */
    /*************************************************************/
    mqTrim(MQ_Q_NAME_LENGTH, qName, qName, &compCode, &reason)
    printf("%4d  %-48s\n", qDepth, qName);
  }
}

else                                            /* Failed mqExecute     */
{
  printf("Call to get queue attributes failed: Completion Code = %d :
          Reason = %d\n", compCode, reason);
```

```
     /*******************************************************************/
     /* If the command fails get the system bag handle out of the mqExecute   */
     /* response bag. This bag contains the reason from the command server    */
     /* why the command failed.                                               */
     /*******************************************************************/
     if (reason == MQRCCF_COMMAND_FAILED)
     {
       mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag, &compCode,
                    &reason);
       CheckCallResult("Get the result bag handle", compCode, reason);

      /*******************************************************************/
      /* Get the completion code and reason code, returned by the command     */
      /* server, from the embedded error bag.                                 */
      /*******************************************************************/
      mqInquireInteger(errorBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                       &compCode, &reason );
      CheckCallResult("Get the completion code from the result bag",
                       compCode, reason);
      mqInquireInteger(errorBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                        &compCode, &reason);
      CheckCallResult("Get the reason code from the result bag",
                       compCode, reason);
      printf("Error returned by the command server: Completion Code = %d :
              Reason = %d\n", mqExecuteCC, mqExecuteRC);
    }
  }

  /*******************************************************************/
  /* Delete the admin bag if successfully created.                        */
  /*******************************************************************/
  if (adminBag != MQHB_UNUSABLE_HBAG)
  {
     mqDeleteBag(&adminBag, &compCode, &reason);
     CheckCallResult("Delete the admin bag", compCode, reason);
  }

  /*******************************************************************/
  /* Delete the response bag if successfully created.                     */
  /*******************************************************************/
  if (responseBag != MQHB_UNUSABLE_HBAG)
  {
     mqDeleteBag(&responseBag, &compCode, &reason);
     CheckCallResult("Delete the response bag", compCode, reason);
  }

  /*******************************************************************/
  /* Disconnect from the queue manager if not already connected           */
  /*******************************************************************/
  if (connReason != MQRC_ALREADY_CONNECTED)
  {
     MQDISC(&hConn, &compCode, &reason);
      CheckCallResult("Disconnect from queue manager", compCode, reason);
  }
  return 0;
}

********************************************************************************/
*                                                                              */
* Function: CheckCallResult                                                    */
*                                                                              */
********************************************************************************/
*                                                                              */
* Input Parameters:  Description of call                                       */
*                    Completion code                                           */
*                    Reason code                                               */
*                                                                              */
* Output Parameters: None                                                      */
*                                                                              */
* Logic: Display the description of the call, the completion code and the      */
*        reason code if the completion code is not successful                  */
*                                                                              */
********************************************************************************/
void  CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
  if (cc != MQCC_OK)
        printf("%s failed: Completion Code = %d : Reason = %d\n",
                callText, cc, rc);
}
```

Consejos y sugerencias para configurar IBM WebSphere MQ

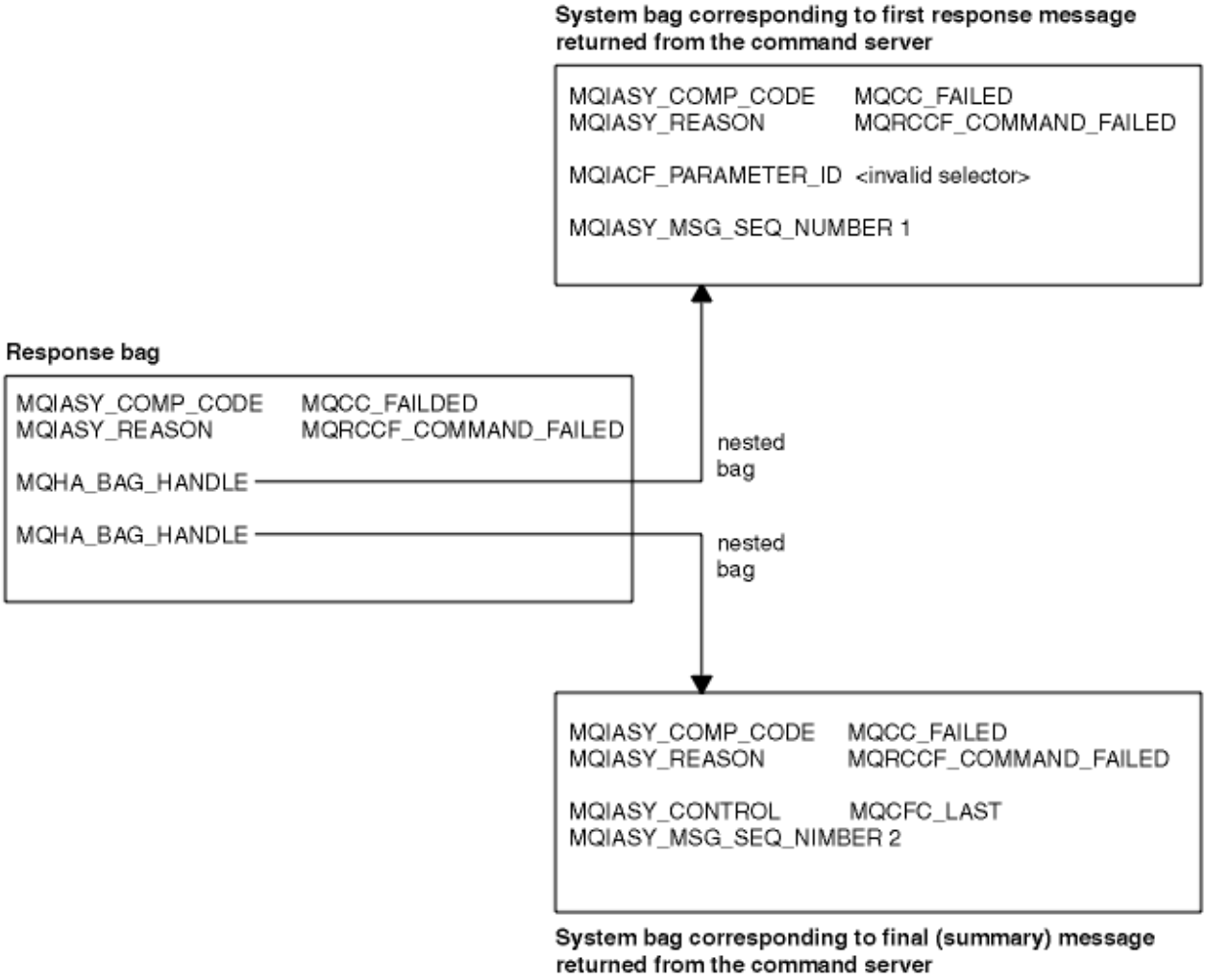
Consejos y sugerencias de programación al utilizar MQAI.

La MQAI utiliza los mensajes PCF para enviar mandatos de administración al servidor de mandatos, en lugar de tratar directamente con el propio servidor de mandatos. A continuación se proporcionan algunas sugerencias para configurar WebSphere MQ utilizando la MQAI:

- Las series de caracteres en WebSphere MQ se rellenan con blancos hasta una longitud fija. Utilizando C, las series terminadas en nulo normalmente pueden proporcionarse como parámetros de entrada a las interfaces de programación de WebSphere MQ.
- Para borrar el valor de un atributo de serie, establézcalo en un único blanco en lugar de una serie vacía.
- Piense con antelación los atributos que desea cambiar y realice consultas únicamente sobre los mismos.
- Determinados atributos no pueden cambiarse, por ejemplo un nombre de cola o un tipo de canal. Asegúrese de que intenta cambiar únicamente los atributos que se pueden modificar. Consulte la lista de parámetros necesarios y opcionales para el objeto de cambio PCF específico. Consulte Definiciones de los formatos de mandatos programables.
- Si una llamada MQAI falla, algunos detalles del error se devuelven al paquete de respuesta. Se pueden encontrar más detalles en un paquete anidado al que puede acceder el selector MQHA_BAG_HANDLE. Por ejemplo, si una llamada mqExecute falla con un código de razón MQRCCF_COMMAND_FAILED, esta información se devuelve en el paquete de respuesta. Una causa posible de este código de razón es que un selector especificado no es válido para el tipo de mensaje de mandato y este detalle de la información se encuentra en un paquete anidado al que puede acceder un manejador de paquete.

  Para obtener más información sobre MQExecute, consulte “Envío de mandatos de administración al servidor de mandatos utilizando la llamada mqExecute” en la página 56

  El siguiente diagrama muestra este escenario:



Temas de la interfaz de administración de IBM MQ avanzados

Información sobre la indexación, conversión de datos y uso del descriptor de mensaje

- Indexación

  Los índices se utilizan cuando se sustituyen o eliminan los elementos de datos existentes de un paquete para conservar el orden de inserción. Los detalles completos sobre la indexación se pueden encontrar en “Indexación en la interfaz de administración de IBM MQ” en la página 43.

- Conversión de datos

  Las series contenidas en un paquete de datos MQAI pueden estar en distintos juegos de caracteres codificados y pueden convertirse utilizando la llamada mqSetInteger. Los detalles completos sobre la conversión de datos se pueden encontrar en “Conversión de datos en la MQAI” en la página 44.

- Uso del descriptor de mensaje

  LA MQAI genera un descriptor de mensaje que se establece en un valor inicial cuando se crea el paquete de datos. Los detalles completos del uso del descriptor de mensaje se pueden encontrar en “Uso del descriptor de mensaje en la interfaz de administración de IBM MQ” en la página 46.

Indexación en la interfaz de administración de IBM MQ

Los índices se utilizan cuando se sustituyen o eliminan los elementos de datos existentes de un paquete. Existen tres tipos de indexación, que permite recuperar fácilmente los elementos de datos.

Cada selector y valor dentro de un elemento de datos de un paquete tiene tres números de asociados:

- El índice relativo a otros elementos que tienen el mismo selector.

- El índice relativo a la categoría de selector (usuario o sistema) al que pertenece el elemento.
- El índice relativo a todos los elementos de datos del paquete (usuario y sistema).

Esto permite la indexación según los selectores de usuario, los selectores del sistema, o ambos como se muestra en la Figura 1 en la página 44.

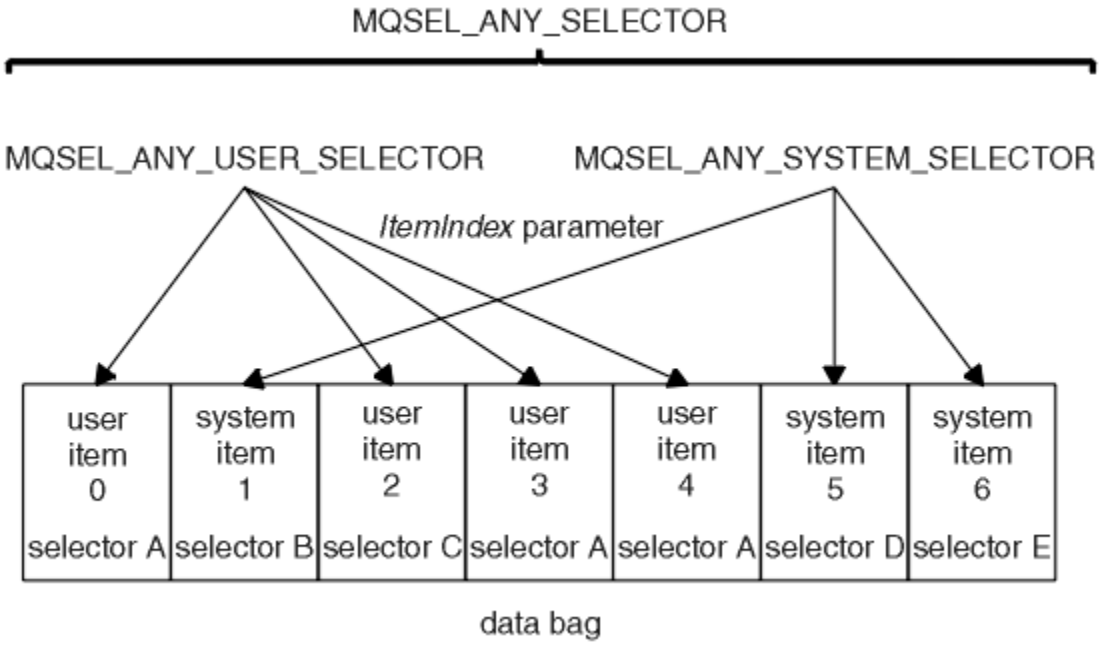


Figura 1. Indexación

En la figura Figura 1 en la página 44, se puede hacer referencia al elemento de usuario 3 (selector A) mediante los siguientes pares de índice:

Selector	***ItemIndex***
selector A	1
MQSEL_ANY_USER_SELECTOR	2
MQSEL_ANY_SELECTOR	3

El índice tiene base cero como una matriz en c; si hay ‘n' apariciones, el índice oscila entre cero y ‘n-1', sin espacios.

Los índices se utilizan cuando se sustituyen o eliminan los elementos de datos existentes de un paquete. Cuando se utilizan de esta manera, el orden de inserción se conserva pero los índices de otros elementos de datos pueden verse afectados. Para obtener ejemplos de esto, consulte Cambiar información dentro de un paquete y Supresión de elementos de datos.

Los tres tipos de indexación permiten una recuperación fácil de elementos de datos. Por ejemplo, si hay tres instancias de un selector concreto en un paquete, la llamada mqCountItems puede contar el número de instancias de ese selector y las llamadas mqInquire* pueden especificar tanto el selector como el índice para consultar solamente los valores. Esto es útil para los atributos que pueden tener una lista de valores como algunas de las salidas en los canales.

Conversión de datos en la MQAI

Las series contenidas en un paquete de datos MQAI pueden estar en diversos juegos de caracteres codificados. Estas series se pueden convertir utilizando la llamada mqSetInteger.

Al igual que los mensajes PCF, las series contenidas en un paquete de datos MQAI pueden estar en diversos juegos de caracteres codificados. Normalmente, todas las series en un mensaje PCF están en el mismo juego de caracteres codificado; es decir, el mismo conjunto que el gestor de colas.

Cada elemento de serie del paquete de datos contiene dos valores; la propia serie y el CCSID. La serie que se añade al paquete se obtiene del parámetro *Buffer* de la llamada de serie mqAddo mqSet. El CCSID se obtiene del elemento del sistema que contiene un selector de MQIASY_CODED_CHAR_SET_ID. Esto se conoce como *CCSID de paquete* y se puede cambiar utilizando la llamada mqSetInteger.

Al consultar el valor de una serie incluida en un paquete de datos, el CCSID es un parámetro de salida de la llamada.

Tabla 2 en la página 45 muestra las normas aplicadas al convertir los paquetes de datos en mensajes y viceversa:

Tabla 2. Proceso de CCSID

Llamada de MQAI	CCSID	Entrada para llamada	Salida para llamada
mqBagToBuffer	CCSID de paquete (1)	Se ignora	Sin modificar
mqBagToBuffer	CCSID de series del paquete	Se utiliza	Sin modificar
mqBagToBuffer	CCSID de series del almacenamiento intermedio	No aplicable	Se copia de los CCSID de series del paquete
mqBufferToBag	CCSID de paquete (1)	Se ignora	Sin modificar
mqBufferToBag	CCSID de series del almacenamiento intermedio	Se utiliza	Sin modificar
mqBufferToBag	CCSID de series del paquete	No aplicable	Se copia de los CCSID de serie del almacenamiento intermedio
mqPutBag	CCSID de MQMD	Se utiliza	Sin cambios (2)
mqPutBag	CCSID de paquete (1)	Se ignora	Sin modificar
mqPutBag	CCSID de series del paquete	Se utiliza	Sin modificar
mqPutBag	CCSID de series del mensaje enviado	No aplicable	Se copia de los CCSID de series del paquete
mqGetBolsa	CCSID de MQMD	Se utiliza para la conversión de datos de mensaje	Establecer en CCSID de datos devueltos (3)
mqGetBolsa	CCSID de paquete (1)	Se ignora	Sin modificar
mqGetBolsa	CCSID de series de mensaje	Se utiliza	Sin modificar
mqGetBolsa	CCSID de series del paquete	No aplicable	Se copia de los CCSID de series del mensaje
mqExecute	CCSID de paquete de solicitud	Se utiliza para MQMD de mensaje de solicitud (4)	Sin modificar
mqExecute	CCSID de paquete de respuesta	Se utiliza para la conversión de datos del mensaje de respuesta (4)	Establecer en CCSID de datos devueltos (3)

Tabla 2. Proceso de CCSID (continuación)			
Llamada de MQAI	**CCSID**	**Entrada para llamada**	**Salida para llamada**
mqExecute	CCSID de series de paquete de solicitud	Se utiliza para el mensaje de solicitud	Sin modificar
mqExecute	CCSID de series en paquete de respuesta	No aplicable	Se copia de los CCSID de serie del mensaje de respuesta

Notas:

1. El CCSID de paquete es el elemento del sistema con el selector MQIASY_CODED_CHAR_SET_ID.
2. MQCCSI_Q_MGR se cambia por el CCSID del gestor de colas real.
3. Si se solicita la conversión de datos, el CCSID de datos devuelto es el mismo que el valor de salida. Si no se solicita la conversión de datos, el CCSID de datos devueltos es el mismo que el valor de mensaje. Tenga en cuenta que se devuelve ningún mensaje si se solicita la conversión de datos pero falla.
4. Si el CCSID es MQCCSI_DEFAULT, se utilizará el CCSID del gestor de colas.

Uso del descriptor de mensaje en la interfaz de administración de IBM MQ

El descriptor de mensaje que se la interfaz de administración de IBM MQ genera se establece en un valor inicial cuando se crea el paquete de datos.

El tipo de mandato PCF se obtiene del elemento del sistema con el selector MQIASY_TYPE. Al crear el paquete de datos, el valor inicial de este elemento se establece en función del tipo de paquete que se crea:

Tabla 3. Tipo de mandato PCF	
Tipo de paquete	**Valor inicial del elemento MQIASY_TYPE**
MQCBO_ADMIN_BAG	MQCFT_COMMAND
MQCBO_COMMAND_BAG	MQCFT_COMMAND
MQCBO_*	MQCFT_USER

Cuando la MQAI genera un descriptor de mensaje, los valores utilizados en los parámetros *Format* y *MsgType* dependen del valor del elemento del sistema con el selector MQIASY_TYPE tal como se muestra en la Tabla 3 en la página 46.

Tabla 4. Formato y parámetros MsgType del MQMD		
Tipo de mandato PCF	**Formato**	**MsgType**
MQCFT_COMMAND	MQFMT_ADMIN	MQMT_REQUEST
MQCFT_REPORT	MQFMT_ADMIN	MQMT_REPORT
MQCFT_RESPONSE	MQFMT_ADMIN	MQMT_REPLY
MQCFT_TRACE_ROUTE	MQFMT_ADMIN	MQMT_DATAGRAM
MQCFT_EVENT	MQFMT_EVENT	MQMT_DATAGRAM
MQCFT_*	MQFMT_PCF	MQMT_DATAGRAM

Tabla 4 en la página 46 muestra que si crea un paquete de administración o un paquete de mandatos, el *Format* del descriptor de mensaje es MQFMT_ADMIN y el *MsgType* es MQMT_REQUEST. Esto resulta adecuado para un mensaje de petición PCF que se envía al servidor de mandatos cuando se espera una respuesta.

Otros parámetros del descriptor de mensaje adoptan los valores que se muestran en la Tabla 5 en la página 47.

Tabla 5. Valores de descriptor de mensaje

Parámetro	**Valor**
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_1
Report	MQRO_NONE
MsgType	Consulte Tabla 4 en la página 46.
Expiry	30 segundos (vea "1" en la página 47)
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	depende del CCSID del paquete (nota "2" en la página 47)
Format	Consulte Tabla 4 en la página 46.
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NOT_PERSISTENT
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	consulte la nota "3" en la página 47
ReplyToQMgr	en blanco

Notas:

1. Este valor se puede alterar temporalmente en la llamada **mqExecute** utilizando el parámetro **OptionsBag** . Para obtener información sobre esto, consulte **mqExecute**.
2. Consulte "Conversión de datos en la MQAI" en la página 44.
3. Nombre de la cola de respuestas especificada por el usuario o cola dinámica temporal generada por MQAI para mensajes de tipo **MQMT_REQUEST**. De lo contrario, espacio en blanco.

Paquetes de datos

Un paquete de datos es una manera de manejar propiedades o parámetros de objetos utilizando la MQAI.

Paquetes de datos

- El paquete de datos contiene cero o más *elementos de datos*. Estos elementos de datos se ordenan dentro del paquete cuando se colocan en el paquete. Esto se denomina el *orden de inserción*. Cada elemento de datos contiene un *selector* que identifica el elemento de datos y un *valor* de ese elemento de datos que puede ser un entero, un entero de 64 bits, un filtro de enteros, una serie, un filtro de texto, una serie de bytes, un filtro de serie de bytes, o un manejador de otro paquete. Los elementos de datos se describen en detalle en "Elemento de datos" en la página 50

  Existen dos tipos de selector; *selectores de usuario* y *selectores de sistema*. Éstos se describen en la sección Selectores de MQAI. Los selectores normalmente son exclusivos, pero es posible tener varios valores para el mismo selector. En este caso, un *índice* identifica la aparición concreta del selector que

es necesario. Los índices se describen en “Indexación en la interfaz de administración de IBM MQ” en la página 43.

Una jerarquía de estos conceptos se muestra en la Figura 1.

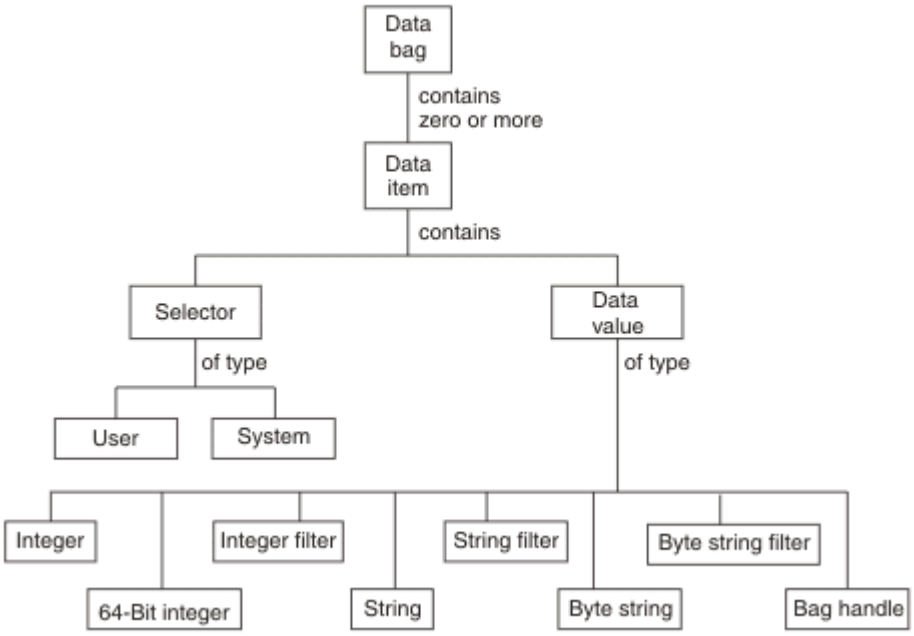


Figura 2. Jerarquía de los conceptos de MQAI

La jerarquía se ha explicado en un párrafo anterior.

Tipos de paquetes de datos

Puede elegir el tipo de paquete de datos que desea crear según la tarea que desea realizar:

paquete de usuario
Un paquete simple que se utiliza para datos de usuario.

paquete de administración
Un paquete creado para los datos utilizados para administrar objetos de WebSphere MQ enviando mensajes de administración para un servidor de mandatos. El paquete de administración supone automáticamente determinadas opciones tal como se describe en “Creación y supresión de paquetes de datos” en la página 49.

paquete de mandatos
También se crea un paquete para mandatos para administrar objetos de WebSphere MQ. Sin embargo, a diferencia del paquete de administración, el paquete de mandatos no implica automáticamente determinadas opciones aunque estas opciones están disponibles. Para obtener más información sobre las opciones, consulte “Creación y supresión de paquetes de datos” en la página 49.

paquete de grupo
Un paquete que se utiliza para mantener un conjunto de elementos de datos agrupados. Los paquetes de grupo no pueden utilizarse para administrar objetos de WebSphere MQ.

Además, el **paquete de sistema** lo crea la MQAI cuando se devuelve un mensaje de respuesta desde el servidor de mandatos y se coloca en el paquete de salida de un usuario. Un paquete de sistema no puede ser modificado por el usuario.

Utilización de paquetes de datos.En este tema se listan las distintas formas de utilizar paquetes de datos:

Utilización de paquetes de datos

En la siguiente lista se muestran las distintas formas de utilizar paquetes de datos:

- Puede crear y suprimir paquetes de datos “Creación y supresión de paquetes de datos” en la página 49.
- Puede enviar datos entre aplicaciones utilizando paquetes de datos “Transferir y recibir paquetes de datos” en la página 50.
- Puede añadir elementos de datos a paquetes de datos “Adición de elementos de datos a paquetes” en la página 51.
- Puede añadir un mandato de consulta dentro de un paquete de datos de “Adición de un mandato de consulta a un paquete” en la página 52.
- Puede consultar dentro de paquetes de datos “Realizar consultas dentro de paquetes de datos” en la página 52.
- Puede contar los elementos de datos dentro de un paquete de datos “Recuento de elementos de datos” en la página 55.
- Puede cambiar información dentro de un paquete de datos “Cambiar información dentro de un paquete” en la página 53.
- Puede borrar un paquete de datos “Borrado de un paquete utilizando la llamada mqClearBag” en la página 54.
- Puede truncar un paquete de datos “Truncamiento de un paquete utilizando la llamada mqTruncateBag” en la página 54.
- Puede convertir paquetes y almacenamientos intermedios “Conversión de paquetes y almacenamientos intermedios” en la página 54.

Creación y supresión de paquetes de datos

Creación de paquetes de datos

Para utilizar la MQAI, primero debe crear un paquete de datos utilizando la llamada mqCreateBag. Como entrada para esta llamada, debe proporcionar una o más opciones para controlar la creación del paquete.

El parámetro *Options* de la llamada MQCreateBag le permite elegir si desea crear un paquete de usuario, un paquete de mandatos, un paquete de grupo o un paquete de administración.

Para crear un paquete de usuario, un paquete de mandatos, o un paquete de grupo, puede elegir una o más opciones adicionales para:

- Utilizar el formato de lista cuando haya dos o más apariciones adyacentes del mismo selector de un paquete.
- Volver a ordenar los elementos de datos cuando se añaden a un mensaje PCF para asegurarse de que los parámetros están en el orden correcto. Para obtener más información sobre los elementos de datos, consulte “Elemento de datos” en la página 50.
- Comprobar los valores de selectores de usuario para los elementos que se añaden al paquete.

Los paquetes de administración implican automáticamente estas opciones.

Un paquete de datos se identifica por su manejador. El manejador de paquete se devuelve de mqCreateBag y debe suministrarse en todas las otras llamadas que utilizan el paquete de datos.

Para obtener una descripción completa de la llamada mqCreateBag, consulte mqCreateBag.

Supresión de paquetes de datos

Cualquier paquete de datos creado por el usuario también debe suprimirse utilizando la llamada mqDeleteBag. Por ejemplo, si se crea un paquete en el código de usuario, también debe suprimirse en el código de usuario.

La MQAI crea y suprime automáticamente paquetes de sistema. Para obtener más información, consulte el apartado "Envío de mandatos de administración al servidor de mandatos utilizando la llamada mqExecute" en la página 56. El código de usuario no puede suprimir un paquete de sistema.

Para obtener una descripción completa de la llamada mqDeleteBag, consulte mqDeleteBag.

Transferir y recibir paquetes de datos

Los datos también se pueden enviar entre aplicaciones transfiriendo y obteniendo paquetes de datos utilizando las llamadas mqPutBag y mqGetBag. Esto permite a la MQAI manejar el almacenamiento intermedio en lugar de la aplicación. La llamada mqPutBag convierte el contenido del paquete especificado en un mensaje PCF y envía el mensaje a la cola especificada y la llamada mqGetBag elimina el mensaje de la cola especificada y la vuelve a convertir en un paquete de datos. Por lo tanto, la llamada mqPutBag es equivalente a la llamada mqBagToBuffer seguida de MQPUT y la llamada mqGetBag es equivalente a la llamada MQGET seguida de mqBufferToBag.

Para obtener más información sobre el envío y recepción de mensajes PCF en una cola específica, consulte "Envío y recepción de mensajes PCF en una cola especificada" en la página 12

Nota: Si decide utilizar la llamada mqGetBag, los detalles de PCF dentro del mensaje deben ser correctos; si no, se produce el correspondiente error y no se devuelve el mensaje PCF.

Elemento de datos

Los elementos de datos se utilizan para llenar los paquetes de datos cuando se crean. Estos elementos de datos pueden ser elementos del usuario o del sistema.

Estos elementos del usuario contienen datos de usuario como atributos de objetos que se están administrando. Los elementos del sistema deben utilizarse para un tener mayor control sobre los mensajes generados: por ejemplo, la generación de las cabeceras de mensaje. Para obtener más información sobre elementos del sistema, consulte "Elementos del sistema" en la página 51.

Tipos de elementos de datos

Cuando haya creado un paquete de datos, puede rellenarlo con elementos de enteros o elementos de series de caracteres. Puede consultar acerca de los tres tipos de elementos.

El elemento de datos puede ser un elemento de entero o un elemento de serie de caracteres. A continuación se muestran los tipos de elementos de datos disponibles dentro de la MQAI:

- Entero
- Entero de 64 bits
- Filtro de entero
- Serie de caracteres
- Filtro de texto
- Serie de bytes
- Filtro de serie de bytes
- Manejador de paquete

Utilización de elementos de datos

A continuación se muestran las diversas formas de utilización de elementos de datos:

- "Recuento de elementos de datos" en la página 55.
- "Supresión de elementos de datos" en la página 55.
- "Adición de elementos de datos a paquetes" en la página 51.

- “Filtrado y consulta de elementos de datos” en la página 52.

Elementos del sistema

Los elementos del sistema puede utilizarse para:

- La generación de cabeceras PCF. Los elementos del sistema puede controlar el identificador de mandatos PCF, las opciones de control, el número de secuencia de mensaje y el tipo de mandato.
- Conversión de datos. Los elementos del sistema manejan el identificador de juego de caracteres para los elementos de serie de caracteres del paquete.

Al igual que todos los elementos de datos, los elementos del sistema constan de un selector y un valor. Para obtener información sobre estos selectores y para lo que sirven, consulte Selectores MQAI.

Los elementos del sistema son exclusivos. Uno o varios elementos del sistema se pueden identificar mediante un selector del sistema. Sólo hay una aparición de cada selector del sistema.

La mayoría de los elementos del sistema pueden modificarse (consulte “Cambiar información dentro de un paquete” en la página 53), pero el usuario no puede modificar las opciones de creación de paquete. No puede suprimir elementos del sistema. (Consulte el apartado “Supresión de elementos de datos” en la página 55).

Adición de elementos de datos a paquetes

Cuando se crea un paquete de datos, puede rellenarlo con elementos de datos. Estos elementos de datos pueden ser elementos del usuario o del sistema. Para obtener más información sobre los elementos de datos, consulte “Elemento de datos” en la página 50.

La MQAI permite añadir elementos de entero, elementos de entero de 64 bits, elementos de filtro de enteros, elementos de serie de caracteres, un filtro de texto, elementos de serie de bytes y elementos de filtro de serie de bytes a paquetes y esto se muestra en Figura 3 en la página 51. Los elementos se identifican mediante un selector. Normalmente un selector identifica un solo elemento, pero este no siempre es el caso. Si un elemento de datos con el selector especificado ya existe en el paquete, se añade una instancia adicional de ese selector al final del paquete.

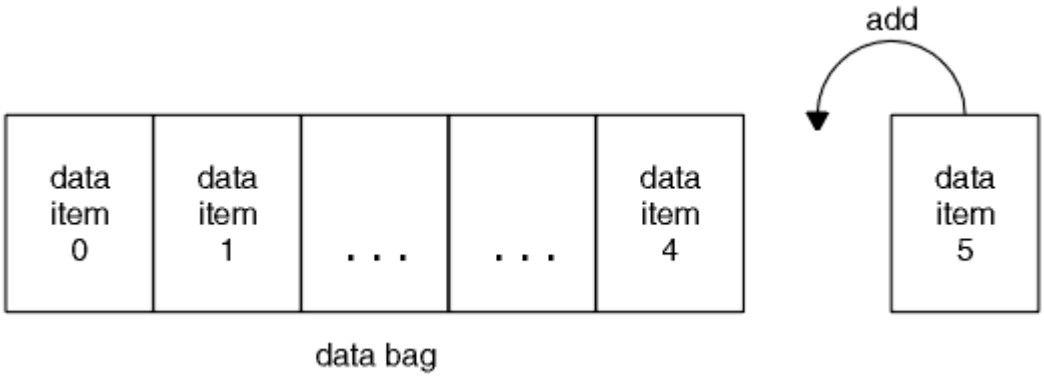


Figura 3. Adición de elementos de datos

Añadir elementos de datos a un paquete utilizando las llamadas mqAdd*:

- Para añadir elementos de entero, utilice la llamada mqAddInteger tal como se describe en mqAddInteger
- Para añadir elementos de entero de 64 bits, utilice la llamada mqAddInteger64 tal como se describe en mqAddInteger64
- Para añadir elementos de filtro de enteros, utilice la llamada mqAddIntegerFilter tal como se describe en mqAddIntegerFilter
- Para añadir elementos de serie de caracteres, utilice la llamada mqAddString tal como se describe en mqAddString
- Para añadir elementos de filtro de texto, utilice la llamada mqAddStringFilter tal como se describe en mqAddStringFilter

- Para añadir elementos de serie de bytes, utilice la llamada mqAddByteString tal como se describe en in mqAddByteString
- Para añadir elementos de filtro de serie de bytes, utilice la llamada mqAddByteStringFilter tal como se describe en mqAddByteStringFilter

Para obtener más información sobre la adición de elementos de datos a un paquete, consulte “Elementos del sistema” en la página 51

Adición de un mandato de consulta a un paquete

La llamada mqAddInquiry se utiliza para añadir un mandato de consulta a un paquete. La llamada es específicamente para fines de administración, por lo que sólo puede utilizarse con paquetes de administración. Permite especificar los selectores de atributos de los que desea realizar consultas desde WebSphere MQ.

Para obtener una descripción completa de la llamada mqAddInquiry, consulte mqAddInquiry.

Filtrado y consulta de elementos de datos

Cuando se utiliza MQAI para consultar sobre los atributos de los objetos de WebSphere MQ, puede controlar los datos que se devuelven al programa de dos formas.

- Puede ***filtrar*** los datos que se devuelven utilizando las llamadas mqAddInteger y mqAddString. Este enfoque permite especificar un par de *Selector* y *ItemValue*, por ejemplo:

  ```
  mqAddInteger(inputbag, MQIA_Q_TYPE, MQQT_LOCAL)
  ```

  Este ejemplo especifica que el tipo de cola (*Selector*) debe ser local (*ItemValue*) y esta especificación debe coincidir con los atributos del objeto (en este caso, una cola) sobre el que se realiza la consulta.

  Otros atributos que se pueden filtrar corresponden a las mandatos Inquire* de PCF que se pueden encontrar en “Introducción a los Formatos de mandato programable” en la página 9. Por ejemplo, para consultar sobre los atributos de un canal, consulte el mandato Consultar canal en esta documentación del producto. Los "parámetros obligatorios" y los "parámetros opcionales" del mandato Consultar canal identifican los selectores que puede utilizar para el filtrado.
- Puede ***consultar*** atributos concretos de un objeto utilizando la llamada mqAddInquiry. Esto especifica el selector en los que está interesado. Si no especifica el selector, se devuelven todos los atributos del objeto.

A continuación se muestra un ejemplo del filtro y la consulta de los atributos de una cola:

```
/* Request information about all queues */
mqAddString(adminbag, MQCA_Q_NAME, "*")

/* Filter attributes so that local queues only are returned */
mqAddInteger(adminbag, MQIA_Q_TYPE, MQQT_LOCAL)

/* Query the names and current depths of the local queues */
mqAddInquiry(adminbag, MQCA_Q_NAME)
mqAddInquiry(adminbag, MQIA_CURRENT_Q_DEPTH)

/* Send inquiry to the command server and wait for reply */
mqExecute(MQCMD_INQUIRE_Q, ...)
```

Para obtener más ejemplos de filtrado y consulta de elementos de datos, consulte “Ejemplos de cómo utilizar la MQAI” en la página 21.

Realizar consultas dentro de paquetes de datos

Puede realizar consultas acerca de:

- El valor de un elemento de entero utilizando la llamada mqInquireInteger. Consulte mqInquireInteger.

- El valor de un elemento de entero de 64 bits utilizando la llamada mqinquireinteger64. Consulte mqInquireInteger64.
- El valor de un elemento de filtro de enteros utilizando la llamada mqInquireIntegerFilter. Consulte mqInquireIntegerFilter.
- El valor de un elemento de serie de caracteres utilizando la llamada mqInquireString. Consulte mqInquireString.
- El valor de un elemento de filtro de texto utilizando la llamada mqInquireStringFilter. Consulte mqInquireStringFilter.
- El valor de un elemento de serie de bytes utilizando la llamada mqInquireByteString. Consulte mqInquireByteString.
- El valor de un elemento de filtro de serie de bytes utilizando la llamada mqInquireByteStringFilter. Consulte mqInquireByteStringFilter.
- El valor de un manejador de paquete utilizando la llamada mqInquireBag. Consulte mqInquireBag.

También puede consultar sobre el tipo (entero, entero de 64 bits, filtro de enteros, serie de caracteres, filtro de texto, serie de bytes, filtro de serie de bytes o manejador de paquete) de un elemento específico utilizando la llamada mqInquireItemInfo. Consulte mqInquireItemInfo.

Cambiar información dentro de un paquete

La MQAI permite cambiar información dentro de un paquete utilizando las llamadas mqSet*. Puede:

1. Modificar elementos de datos dentro de un paquete. El índice permite sustituir una instancia individual de un parámetro identificando la aparición del elemento que se debe modificar (consulte Figura 4 en la página 53).

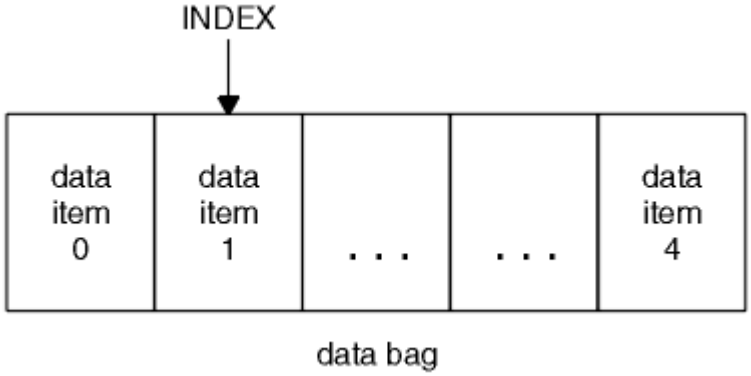


Figura 4. Modificación de un único elemento de datos

2. Suprimir todas las apariciones existentes del selector especificado y añadir una nueva aparición al final del paquete. (Consulte el apartado Figura 5 en la página 53). Un valor de índice especial permite sustituir ***todas*** las instancias de un parámetro.

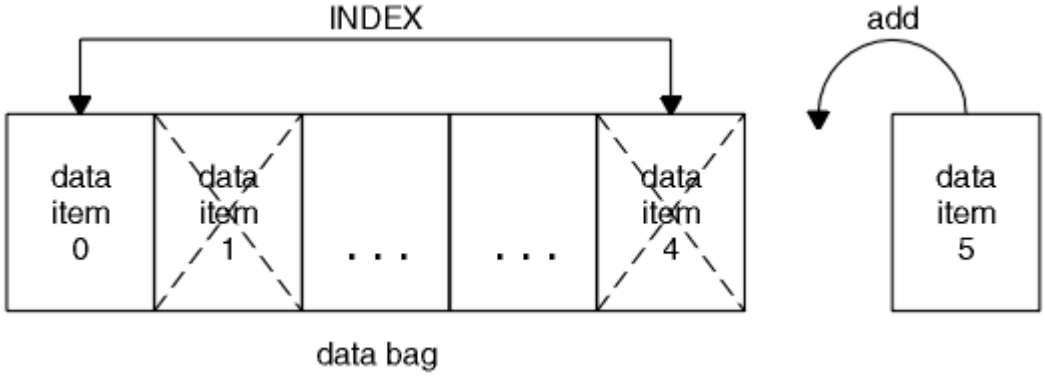


Figura 5. Modificación de todos los elementos de datos

Nota: El índice conserva el orden de inserción en el paquete aunque puede afectar a los índices de otros elementos de datos.

La llamada mqSetInteger permite modificar elementos de entero dentro de un paquete. La llamada mqSetInteger64 permite modificar elementos de enteros de 64 bits. La llamada mqSetIntegerFilter permite modificar elementos de filtro de enteros. La llamada mqSetString permite modificar elementos de serie de caracteres. La llamada mqSetStringFilter permite modificar elementos de filtro de texto. La llamada mqSetByteString permite modificar elementos de serie de bytes. La llamada mqSetByteStringFilter permite modificar elementos de filtro de serie de bytes. Como alternativa, puede utilizar estas llamadas para suprimir todas las apariciones existentes del selector especificado y añadir una nueva aparición al final del paquete. El elemento de datos puede ser un elemento de usuario o un elemento de sistema.

Para obtener una descripción completa de estas llamadas, consulte:

- mqSetInteger
- mqSetInteger64
- mqSetIntegerFilter
- mqSetString
- mqSetStringFilter
- mqSetByteString
- mqSetByteStringFilter

Borrado de un paquete utilizando la llamada mqClearBag

La llamada mqClearBag elimina todos los elementos de usuario de un paquete de usuario y restablece los elementos del sistema a sus valores iniciales. También se suprimen los paquetes de sistema incluidos dentro del paquete.

Para obtener una descripción completa de la llamada mqClearBag, consulte mqClearBag.

Truncamiento de un paquete utilizando la llamada mqTruncateBag

La llamada mqTruncateBag reduce el número de elementos de usuario de un paquete de usuario suprimiendo los elementos del final del paquete, empezando por el elemento añadido más recientemente. Por ejemplo, puede usarse cuando se utiliza la misma información de cabecera para generar más de un mensaje.

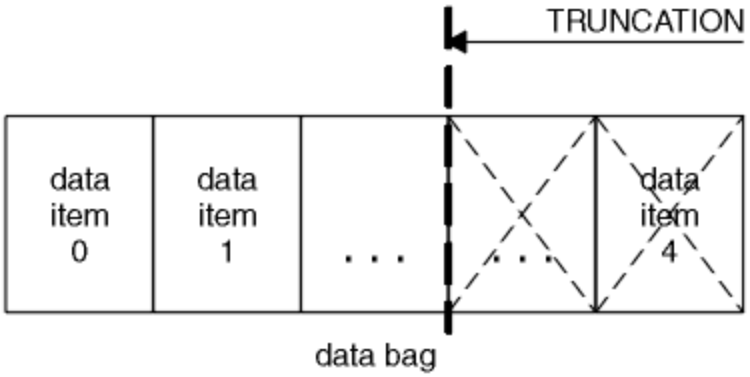


Figura 6. Truncar un paquete

Para obtener una descripción completa de la llamada mqTruncateBag, consulte mqTruncateBag.

Conversión de paquetes y almacenamientos intermedios

Para enviar datos entre aplicaciones, primero los datos del mensaje se colocan en un paquete. A continuación, los datos del paquete se convierten en un mensaje PCF utilizando la llamada mqBagToBuffer. El mensaje PCF se envía a la cola necesaria utilizando la llamada MQPUT. Esto se

muestra en la figura Figura 7 en la página 55. Para obtener una descripción completa de la llamada mqBagToBuffer, consulte mqBagToBuffer.

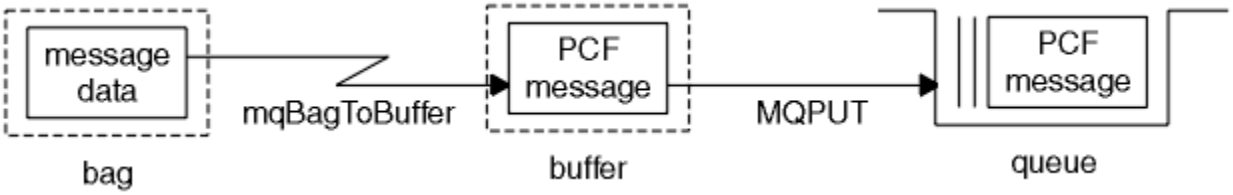


Figura 7. Conversión de paquetes en mensajes PCF

Para recibir datos, el mensaje se recibe en un almacenamiento intermedio utilizando la llamada MQGET. Los datos del almacenamiento intermedio a continuación se convierten en un paquete utilizando la llamada mqBufferToBag, siempre que el almacenamiento intermedio contenga un mensaje PCF válido. Esto se muestra en la Figura 8 en la página 55. Para obtener una descripción completa de la llamada mqBufferToBag, consulte mqBufferToBag.

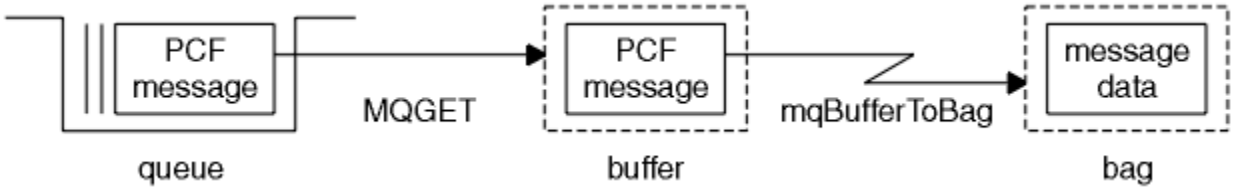


Figura 8. Conversión de mensajes PCF a formato de paquete

Recuento de elementos de datos

La llamada mqCountItems cuenta el número de elementos de usuario, elementos del sistema, o ambos, que están almacenados en un paquete de datos, y devuelve este número. Por ejemplo, `mqCountItems(`*`Bag`*`, 7, ...)`, devuelve el número de elementos del paquete con un selector de 7. Puede contar elementos por selector individual, por selectores de usuario, por selectores del sistema o por todos los selectores.

Nota: Esta llamada cuenta el número de elementos de datos, no el número de selectores exclusivos del paquete. Un selector puede darse varias veces, por lo que puede haber menos selectores exclusivos en el paquete que elementos de datos.

Para obtener una descripción completa de la llamada mqCountItems, consulte mqCountItems.

Supresión de elementos de datos

Puede suprimir elementos de paquetes de varias formas. Puede:

- Eliminar uno o más elementos de usuario de un paquete. Para obtener información detallada, consulte “Supresión de elementos de datos de un paquete utilizando la llamada mqDeleteItem” en la página 55.
- Suprimir ***todos*** los elementos de usuario de un paquete, es decir, *borre* un paquete. Para obtener información detallada, consulte “Borrado de un paquete utilizando la llamada mqClearBag” en la página 54.
- Suprimir todos los elementos de usuario del final de un paquete, es decir, *trunque* un paquete. Para obtener información detallada, consulte “Truncamiento de un paquete utilizando la llamada mqTruncateBag” en la página 54.

Supresión de elementos de datos de un paquete utilizando la llamada mqDeleteItem

La llamada mqDeleteItem elimina uno o más elementos de usuario de un paquete. El índice se utiliza para suprimir uno de los siguientes valores:

1. Una única aparición del selector especificado. (Consulte Figura 9 en la página 56.)

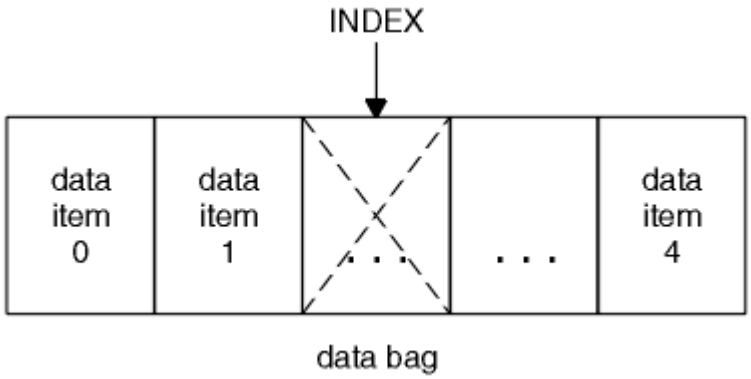


Figura 9. Supresión de un único elemento de datos

o

2. Todas las apariciones del selector especificado. (Consulte Figura 10 en la página 56.)

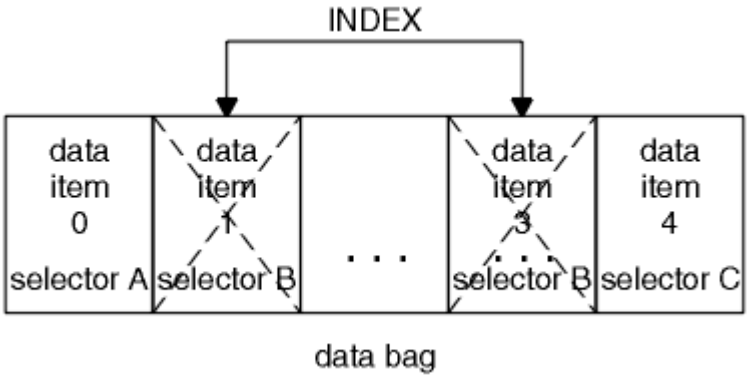


Figura 10. Supresión de todos los elementos de datos

Nota: El índice conserva el orden de inserción en el paquete aunque puede afectar a los índices de otros elementos de datos. Por ejemplo, la llamada mqDeleteItem no conserva los valores de índice de los elementos de datos que siguen el elemento suprimido porque los índices se reorganizan para rellenar el hueco que permanece del elemento suprimido.

Para obtener una descripción completa de la llamada mqDeleteItem, consulte mqDeleteItem.

Envío de mandatos de administración al servidor de mandatos utilizando la llamada mqExecute

Una vez que se ha creado y llenado un paquete de datos, puede enviarse un mensaje de mandato administrativo al servidor de mandatos de un gestor de colas utilizando la llamada mqExecute. Esto maneja el intercambio con el servidor de mandatos y devuelve respuestas en un paquete.

Después de haber creado y llenado los paquetes de datos, puede enviar un mensaje de mandato de administración al servidor de mandatos de un gestor de colas. La forma más fácil de hacerlo es utilizando la llamada mqExecute. La llamada mqExecute envía un mensaje de mandato de administración como mensaje no persistente y espera las respuestas. Las respuestas se devuelven en un paquete de respuesta. Por ejemplo, estas pueden incluir información sobre los atributos relacionados con varios objetos de WebSphere MQ o una serie de mensajes de respuesta de error PCF. Por lo tanto, el paquete de respuesta podría contener un código de retorno o sólo *paquetes anidados*.

Los mensajes de respuesta se colocan en paquetes del sistema que crea el sistema. Por ejemplo, para consultas sobre los nombres de los objetos, se crea un paquete de sistema para mantener esos nombres de objeto y el paquete se inserta en el paquete de usuario. Los manejadores de estos paquetes se insertarán en el paquete de respuesta y el selector MQHA_BAG_HANDLE podrá acceder al paquete anidado. El paquete de sistema permanece en el almacenamiento, si no se suprime, hasta que se suprime el paquete de respuestas.

El concepto de *anidamiento* es muestra en la Figura 11 en la página 57.

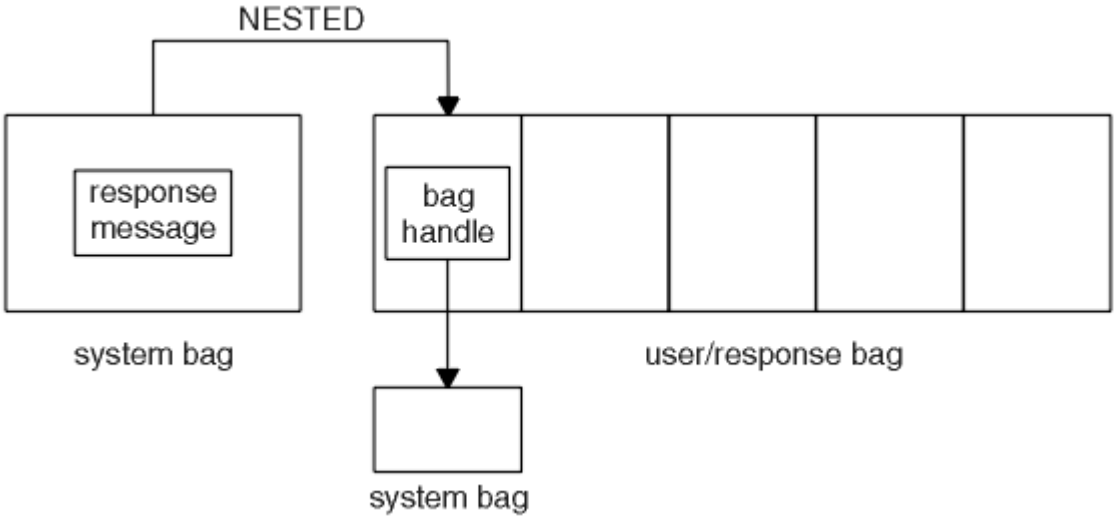


Figura 11. Anidamiento

Como entrada para la llamada mqExecute debe proporcionar:

- Un manejador de conexión MQI.
- El mandato que se va a ejecutar. Este debe ser uno de los valores MQCMD_*.

  Nota: Si la MQAI no reconoce este valor, el valor se sigue aceptando. Sin embargo, si se utilizó la llamada mqAddInquiry para insertar valores en el paquete, este parámetro debe ser un mandato INQUIRE reconocido por la MQAI. Es decir, el parámetro debe tener el formato MQCMD_INQUIRE_*.
- Opcionalmente, un manejador del paquete que contiene las opciones que controlan el proceso de la llamada. Aquí también es donde puede especificar el tiempo máximo en milisegundos que la MQAI debe esperar para cada mensaje de respuesta.
- Un manejador del paquete de administración que contiene detalles del mandato de administración que va a emitirse.
- Un manejador del mensaje de respuesta que recibe los mensajes de respuesta.

Los siguientes son opcionales:

- Un manejador de objeto de la cola donde se colocará el mandato de administración.

  Si no se especifica ningún manejador de objeto, el mandato de administración se coloca en la cola SYSTEM.ADMIN.COMMAND.QUEUE que pertenece al gestor de colas conectado actualmente. Éste es el valor predeterminado.
- Un manejador de objeto de la cola donde se colocarán los mensajes de respuesta.

  Puede elegir colocar los mensajes de respuesta en una cola dinámica que la MQAI crea automáticamente. La cola creada sólo existe durante la llamada y la MQAI la suprime al salir de la llamada mqExecute.

Para obtener ejemplos de usos de la llamada mqExecute, consulte el apartado Código de ejemplo

Administración utilizando IBM WebSphere MQ Explorer

IBM WebSphere MQ Explorer le permite realizar una administración local o remota de la red desde un sistema que ejecuta sólo Windows o Linux (plataformas x86 y x86-64).

IBM WebSphere MQ para Windows y IBM WebSphere MQ para Linux (plataformas x86 y x86-64) proporcionan una interfaz de administración llamada IBM WebSphere MQ Explorer para realizar tareas de administración como alternativa al uso del control o los mandatos MQSC. Comparación de juegos de mandatos muestra lo que puede hacer utilizando IBM WebSphere MQ Explorer.

IBM WebSphere MQ Explorer le permite realizar la administración local o remota de la red desde un sistema que ejecuta Windows, o Linux (plataformas x86-64), apuntando a IBM WebSphere MQ Explorer en los gestores de colas y los clústeres en los que está interesado. Las plataformas y los niveles de IBM WebSphere MQ que se pueden administrar utilizando IBM WebSphere MQ Explorer se describen en "Gestores de colas remotos" en la página 59.

Para configurar gestores de colas remotas de IBM WebSphere MQ para que IBM WebSphere MQ Explorer los pueda administrar, consulte "Software de requisito previo y definiciones" en la página 60.

Le permite realizar tareas, normalmente asociadas a la configuración y el ajuste preciso del entorno de trabajo para IBM WebSphere MQ, ya sea de forma local o remota dentro de un dominio de sistema Windows o Linux (plataformas x86 y x86-64).

En Linux, puede que IBM WebSphere MQ Explorer no se inicie correctamente si dispone de una instalación Eclipse. Si esto sucediera, inicie IBM WebSphere MQ Explorer utilizando un ID de usuario diferente del que utiliza para la otra instalación de Eclipse.

En Linux, para iniciar correctamente IBM WebSphere MQ Explorer, debe poder grabar un archivo en el directorio de inicio y debe existir un directorio de inicio.

Qué puede hacer con IBM WebSphere MQ Explorer

Esta es una lista de las tareas que puede realizar mediante IBM WebSphere MQ Explorer.

Con IBM WebSphere MQ Explorer, puede:

- Crear y suprimir un gestor de colas (sólo en la máquina local).
- Iniciar y detener un gestor de colas (sólo en la máquina local).
- Definir, visualizar y modificar las definiciones de objetos WebSphere MQ, tales como colas y canales.
- Examinar los mensajes de una cola.
- Iniciar y detener un canal.
- Ver información de estado sobre un canal, escucha, cola u objetos de servicio.
- Ver los gestores de colas de un clúster.
- Comprobar qué aplicaciones, usuarios o canales tienen una determinada cola abierta.
- Crear un nuevo clúster de gestor de colas utilizando el asistente para *Crear un nuevo clúster*.
- Añadir un gestor de colas a un clúster utilizando el asistente para *Añadir un gestor de colas a un clúster*.
- Gestionar el objeto de información de autenticación, que se utiliza con la seguridad de canal de Secure Sockets Layer (SSL).
- Crear y suprimir iniciadores de canal, supervisores desencadenantes y escuchas.
- Iniciar o detener los servidores de mandatos, iniciadores de canal, supervisores desencadenantes y escuchas.
- Configurar servicios específicos para que se inicien automáticamente cuando se inicie un gestor de colas.
- Modificar las propiedades de los gestores de colas.
- Cambiar el gestor de colas local predeterminado.
- Invocar la GUI de ikeyman para gestionar certificados SSL (Capa de sockets seguros), asociar certificados con gestores de colas y configurar y establecer almacenes de certificados (sólo en la máquina local).
- Crear objetos JMS a partir de objetos WebSphere MQ y objetos WebSphere MQ a partir de objetos JMS.
- Crear una fábrica de conexiones JMS para cualquiera de los tipos soportados actualmente.
- Modificar los parámetros de cualquier servicio, como el número de puerto TCP de un escucha o el nombre de una cola de iniciador de canal.
- Iniciar o detener el rastreo de servicio.

Las tareas de administración se realizan utilizando una serie de *vistas de contenido* y *diálogos de propiedades*.

Vista de Contenido
Una vista de Contenido es un panel que puede mostrar lo siguiente:

- Atributos y opciones administrativas relativos al propio producto WebSphere MQ.
- Atributos y opciones administrativas relativos a uno o más objetos relacionados.
- Atributos y opciones administrativas de un clúster.

Diálogos de propiedades
Un diálogo de propiedades es un panel que muestra atributos relativos a un objeto en una serie de campos, algunos de los cuales se pueden editar.

Para navegar por WebSphere MQ Explorer se utiliza la *vista de navegador*. El navegador le permite seleccionar la vista de Contenido que necesite.

Gestores de colas remotos

Hay dos excepciones respecto a los gestores de colas soportados a los que puede conectarse.

Desde un sistema Windows o Linux (plataformasx86 y x86-64), WebSphere MQ Explorer puede conectarse a todos los gestores de colas soportados con las excepciones siguientes:

- Gestores de colas de WebSphere MQ para z/OS anteriores a la Versión 6.0.
- Gestores de colas de MQSeries V2 soportados actualmente.

IBM WebSphere MQ Explorer maneja las diferencias en las funcionalidades entre los distintos niveles de mandatos y plataformas. Sin embargo, si encuentra un atributo que no reconoce, dicho atributo no será visible.

Si desea administrar de forma remota un gestor de colas V6.0 o posterior en Windows utilizando IBM WebSphere MQ Explorer en un sistema WebSphere MQ V5.3, debe instalar el Fix Pack 9 (CSD9) o posterior en su sistema WebSphere MQ para Windows V5.3.

Si desea administrar de forma remota un gestor de colas V5.3 en iSeries utilizando WebSphere MQ Explorer en un sistema WebSphere MQ V6.0 o posterior, debe instalar el Fix Pack 11 (CSD11) o posterior en su sistema WebSphere MQ para iSeries V5.3. Este fixpack corrige los problemas de conexión entre WebSphere MQ Explorer y el gestor de colas iSeries.

Decidir si se debe utilizar el IBM WebSphere MQ Explorer

A la hora de decidir si desea utilizar IBM WebSphere MQ Explorer en su instalación, tenga en cuenta la información que figura en este tema.

Debe tener en cuenta los puntos siguientes:

Nombres de objetos
Si utiliza nombres en minúsculas para gestores de colas y otros objetos con IBM WebSphere MQ Explorer, cuando trabaje con los objetos utilizando mandatos MQSC, debe encerrar los nombres de objetos entre comillas simples o WebSphere MQ no los reconoce.

Gestores de colas grandes
IBM WebSphere MQ Explorer funciona mejor con gestores de colas pequeños. Si tiene un gran número de objetos en un solo gestor de colas, puede sufrir retrasos mientras WebSphere MQ Explorer extrae la información necesaria para presentarla en una vista.

Clústeres
Los clústeres de WebSphere MQ pueden potencialmente contener cientos o miles de gestores de colas. WebSphere MQ Explorer presenta los gestores de colas de un clúster mediante una estructura de árbol. El tamaño físico de un clúster no afecta drásticamente la velocidad de IBM WebSphere MQ Explorer, ya que IBM WebSphere MQ Explorer no se conecta a los gestores de colas del clúster hasta que el usuario los selecciona.

Configuración de IBM WebSphere MQ Explorer

En esta sección se describen los pasos que debe realizar para configurar IBM WebSphere MQ Explorer.

Software de requisito previo y definiciones

Asegúrese de cumplir los siguientes requisitos antes de intentar utilizar el IBM WebSphere MQ Explorer.

El IBM WebSphere MQ Explorer sólo puede conectarse con gestores de colas remotos utilizando el protocolo de comunicaciones TCP/IP.

Compruebe que:

1. Se está ejecutando un servidor de mandatos en cada gestor de colas administrado de forma remota.
2. Se está ejecutando un objeto de escucha TCP/IP adecuado en cada gestor de colas remoto. Este objeto puede ser el escucha de IBM WebSphere MQ o, en sistemas UNIX and Linux, el daemon inetd.
3. Existe un canal de conexión de servidor, cuyo nombre predeterminado es SYSTEM.ADMIN.SVRCONN, en todos los gestores de colas remotos.

   Puede crear el canal utilizando el siguiente mandato MQSC:

   ```
   DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(SVRCONN)
   ```

   Este mandato crea una definición de canal básica. Si desea una definición más compleja (por ejemplo, para configurar la seguridad), necesita parámetros adicionales. Para obtener más información, consulte DEFINE CHANNEL .
4. Debe existir la cola del sistema, SYSTEM.MQEXPLORER.REPLY.MODEL.

Seguridad

Si utiliza WebSphere MQ en un entorno en el que es importante para usted controlar el acceso de usuarios a determinados objetos, tal vez necesite considerar los aspectos de seguridad que implica la utilización de IBM WebSphere MQ Explorer.

Autorización para utilizar IBM WebSphere MQ Explorer

Cualquier usuario puede utilizar IBM WebSphere MQ Explorer, pero se requieren ciertas autorizaciones para conectarse y acceder a los gestores de colas así como para gestionarlos.

Para realizar tareas de administración locales utilizando WebSphere MQ Explorer, un usuario debe tener la autorización necesaria para realizar las tareas de administración. Si el usuario es miembro del grupo mqm, el usuario tiene autorización para realizar todas las tareas administrativas locales.

Para conectarse a un gestor de colas remoto y realizar tareas de administración remotas utilizando WebSphere MQ Explorer, el usuario que ejecuta WebSphere MQ Explorer debe tener las siguientes autorizaciones:

- Autorización CONNECT en el objeto de gestor de colas de destino
- Autorización INQUIRE en el objeto de gestor de colas de destino
- Autorización DISPLAY para el objeto de gestor de colas de destino
- Autorización INQUIRE para la cola SYSTEM.MQEXPLORER.REPLY.MODEL
- Autorización DISPLAY para la cola SYSTEM.MQEXPLORER.REPLY.MODEL
- Autorización INPUT (get) para la cola SYSTEM.MQEXPLORER.REPLY.MODEL

- Autorización OUTPUT (put) para la cola SYSTEM.ADMIN.COMMAND.QUEUE
- Autorización INQUIRE en la cola SYSTEM.ADMIN.COMMAND.QUEUE
- Autorización para realizar la acción seleccionada

Nota: La autorización INPUT está relacionada con la entrada al usuario desde una cola (una operación de obtener). Una autorización OUTPUT está relacionada con la salida desde el usuario a una cola (una operación de transferir).

Para conectarse a un gestor de colas remoto en WebSphere MQ para z/OS y realizar tareas de administración remotas utilizando IBM WebSphere MQ Explorer, debe proporcionarse lo siguiente:

- Un perfil RACF para la cola del sistema, SYSTEM.MQEXPLORER.REPLY.MODEL
- Un perfil RACF para la colas, AMQ.MQEXPLORER.*

Además, el usuario que ejecute WebSphere MQ Explorer debe tener las siguientes autorizaciones:

- Autorización RACF UPDATE para la cola del sistema, SYSTEM.MQEXPLORER.REPLY.MODEL
- Autorización RACF UPDATE para las colas, AMQ.MQEXPLORER.*
- Autorización CONNECT en el objeto de gestor de colas de destino
- Autorización para realizar la acción seleccionada
- Autorización READ para todos los perfiles hlq.DISPLAY.object de la clase MQCMDS

Para obtener información acerca de cómo otorgar autorización sobre objetos de WebSphere MQ, consulte la sección Otorgar acceso a un objeto de WebSphere MQ en sistemas UNIX o Linux y Windows.

Si un usuario intenta realizar una operación para la que no está autorizado, el gestor de colas de destino invoca procedimientos de error de autorización y la operación no se realiza correctamente.

El filtro predeterminado en WebSphere MQ Explorer es mostrar todos los objetos WebSphere MQ. Si hay algún objeto WebSphere MQ para el que un usuario no tiene autorización DISPLAY, se generan errores de autorización. Si se están registrando sucesos de autorización, restrinja el rango de objetos que se muestran a aquellos para los que el usuario tiene autorización DISPLAY.

Seguridad para la conexión con gestores de colas remotos

Debe proteger el canal entre IBM WebSphere MQ Explorer y cada gestor de colas remoto.

IBM WebSphere MQ Explorer se conecta con los gestores de colas remotos como una aplicación cliente MQI. Esto significa que cada gestor de colas remoto debe tener una definición de un canal de conexión de servidor y un escucha TCP/IP adecuado. Si no protege el canal de conexión del servidor, es posible que una aplicación maliciosa preparada para ello se conecte al mismo canal de conexión de servidor y obtenga acceso a los objetos de gestor de colas con autorización ilimitada. Para proteger el canal de conexión del servidor, especifique un valor, que no sea dejarlo en blanco, para el atributo MCAUSER del canal, utilice los registros de autenticación de canal o bien utilice una salida de seguridad.

El valor predeterminado del atributo MCAUSER es el ID de usuario local. Si especifica un nombre de usuario que no sea un blanco como el atributo MCAUSER del canal de conexión de servidor, todos los programas que se conecten al gestor de colas utilizando este canal se ejecutan con la identidad del usuario nombrado y tiene el mismo nivel de autorización. Esto no ocurre si utiliza registros de autenticación de canal.

Utilización de una salida de seguridad con WebSphere MQ Explorer

Puede especificar una salida de seguridad predeterminada y salidas de seguridad específicas del gestor de colas mediante WebSphere MQ Explorer.

Puede definir una salida de seguridad predeterminada, que se puede utilizar para todas las conexiones de cliente nuevas desde WebSphere MQ Explorer. Esta salida predeterminada se puede alterar temporalmente en el momento en que se realiza una conexión. También puede definir una salida de seguridad para un gestor de colas individual o para un conjunto de gestores de colas, lo que se lleva a cabo cuando se realiza la conexión. Las salidas se especifican mediante WebSphere MQ Explorer. Para obtener más información, consulte el centro de ayuda de WebSphere MQ.

Utilización del IBM WebSphere MQ Explorer para conectarse a un gestor de colas remoto utilizando canales MQI habilitados para SSL

IBM WebSphere MQ Explorer se conecta a los gestores de colas remotos utilizando un canal MQI. Si desea proteger el canal MQI con el protocolo de seguridad SSL, debe establecer el canal utilizando una tabla de definiciones de canal de cliente.

Para obtener información sobre cómo establecer un canal MQI utilizando una tabla de definiciones de canal de cliente, consulteVisión general de clientes de IBM WebSphere MQ MQI.

Cuando haya establecido el canal utilizando una tabla de definiciones de canal de cliente, puede utilizar el IBM WebSphere MQ Explorer para conectarse a un gestor de colas remoto utilizando canales MQI habilitados para SSL, tal como se describe en "Tareas en el sistema en el que se aloja el gestor de colas remoto" en la página 62 y en "Tareas en el sistema que aloja IBM WebSphere MQ Explorer" en la página 62.

Tareas en el sistema en el que se aloja el gestor de colas remoto

En el sistema en el que se aloje el gestor de colas remoto, efectúe las tareas siguientes:

1. Defina un par de canales de conexión con el servidor y de conexión con el cliente, y especifique el valor apropiado para la variable *SSLCIPH* en la conexión con el servidor en ambos canales. Para obtener más información sobre la variable *SSLCIPH*, consulte Protección de canales con SSL
2. Envíe la tabla de definiciones de canal `AMQCLCHL.TAB`, que se encuentra en el directorio `@ipcc` del gestor de colas, al sistema que aloja el IBM WebSphere MQ Explorer.
3. Inicie un escucha TCP/IP en un puerto designado.
4. Coloque ambos certificados SSL, CA y el personal, en el directorio SSL del gestor de colas:
   - `/var/mqm/qmgrs/+QMNAME+/SSL` para sistemas UNIX and Linux
   - `C:\Program Files\WebSphere MQ\qmgrs\+QMNAME+\SSL` para sistemas Windows

   Donde +QMNAME+ es una señal que representa el nombre del gestor de colas.
5. Cree un archivo de base de datos de claves de tipo CMS con el nombre `key.kdb`. Oculta la contraseña en un archivo marcando la opción en la GUI de iKeyman o utilizando la opción `-stash` con los mandatos **runmqckm** .
6. Añada los certificados CA a la base de datos de claves creada en el paso anterior.
7. Importe el certificado personal del gestor de colas a la base de datos de claves.

Para obtener información más detallada sobre cómo trabajar con Secure Sockets Layer en sistemas Windows , consulte Trabajar con SSL o TLS en sistemas UNIX, Linux y Windows .

Tareas en el sistema que aloja IBM WebSphere MQ Explorer

En el sistema que aloja IBM WebSphere MQ Explorer, realice las tareas siguientes:

1. Cree un archivo de base de datos de claves de tipo JKS denominado `key.jks`. Establezca una contraseña para este archivo.

   El IBM WebSphere MQ Explorer utiliza los archivos de almacén de claves Java (JKS) para la seguridad SSL, por lo que el archivo de almacén de claves que se crea para configurar SSL para el IBM WebSphere MQ Explorer debe coincidir con éste.
2. Añada los certificados CA a la base de datos de claves creada en el paso anterior.
3. Importe el certificado personal del gestor de colas a la base de datos de claves.
4. En sistemas Windows y Linux, inicie MQ Explorer utilizando el menú del sistema, el archivo ejecutable `MQExplorer` o el mandato **strmqcfg**.
5. En la barra de herramientas de IBM WebSphere MQ Explorer , pulse **Ventana-> Preferencias**y, a continuación, expanda **WebSphere MQ Explorer** y pulse **Almacenes de certificados de cliente SSL**. Entre el nombre y la contraseña del archivo JKS creado en el paso 1 de "Tareas en el sistema que aloja

IBM WebSphere MQ Explorer" en la página 62, tanto en el Almacén de certificados fiable como en el Almacén de certificados personal, y pulse **Aceptar**.

6. Cierre la ventana **Preferencias** y pulse el botón derecho del ratón en **Gestores de colas**. Pulse **Mostrar/ocultar gestores de colas** y luego pulse **Añadir** en la pantalla **Mostrar/ocultar gestores de colas**.
7. Escriba el nombre del gestor de colas y seleccione la opción **Conectar directamente**. Pulse Siguiente.
8. Seleccione **Utilizar tabla de definiciones de canal de cliente (CCDT)** y especifique la ubicación del archivo de tabla de canales que ha transferido del gestor de colas remoto en el paso 2 en "Tareas en el sistema en el que se aloja el gestor de colas remoto" en la página 62 en el sistema en el que se aloja el gestor de colas remoto.
9. Pulse **Finalizar**. Ahora puede acceder al gestor de colas remoto desde el IBM WebSphere MQ Explorer.

Conexión a través de otro gestor de colas

IBM WebSphere MQ Explorer le permite conectarse a un gestor de colas a través de un gestor de colas intermedio, al que IBM WebSphere MQ Explorer ya está conectado.

En este caso, IBM WebSphere MQ Explorer transfiere mensajes de mandato PCF al gestor de colas intermedio, especificando lo siguiente:

- El parámetro *NombreGestColasObjeto* del descriptor de objeto (MQOD) como nombre del gestor de colas de destino. Para obtener más información sobre la resolución de nombres de cola, consulte Resolución de nombres .
- El parámetro *IdentificadorUsuario* del descriptor de mensaje (MQMD) como el ID de usuario local.

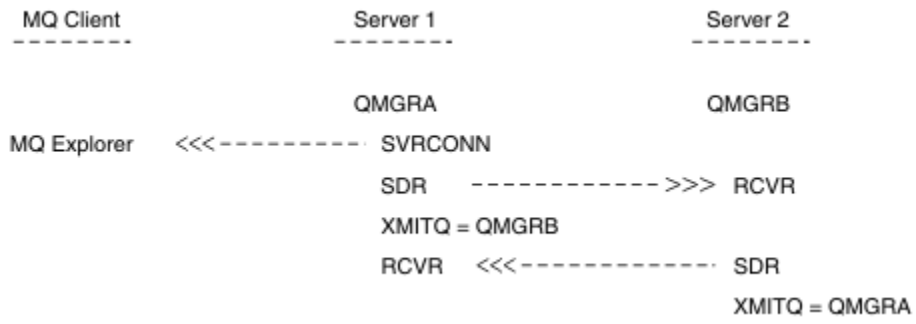
Si la conexión se utiliza después para conectar con el gestor de colas de destino mediante un gestor de colas intermediario, el ID de usuario se incorpora al parámetro *IdentificadorUsuario* del descriptor de mensaje (MQMD). Para que el escucha MCA del gestor de colas de destino acepte este mensaje, debe establecerse el atributo MCAUSER o el ID de usuario ya debe existir con autorización para transferencias.

El servidor de mandatos del gestor de colas de destino transfiere mensajes a la cola de transmisión especificando el ID de usuario del parámetro *IdentificadorUsuario* del descriptor de mensaje (MQMD). Para que esta transferencia se realice correctamente, el ID de usuario ya debe existir en el gestor de colas de destino con autorización para transferencias.

El ejemplo siguiente le muestra cómo conectar un gestor de colas, mediante un gestor de colas intermediario, a WebSphere MQ Explorer.

Establezca una conexión de administración remota con un gestor de colas. Verifique que:

- El gestor de colas en el servidor está activo y tiene un canal de conexión de servidor (SVRCONN) definido.
- El escucha está activo.
- El servidor de mandatos está activo.
- La cola SYSTEM.MQ EXPLORER.REPLY.MODEL se ha creado y que tiene suficiente autorización.
- Los escuchas, los servidores de mandatos y los canales emisores de los gestores de colas están iniciados.



En este ejemplo:

- IBM WebSphere MQ Explorer se conecta al gestor de colas QMGRA (que se ejecuta en el Servidor1) utilizando una conexión de cliente.
- El gestor de colas QMGRB en el Servidor2 ahora puede conectarse a IBM WebSphere MQ Explorer a través de un gestor de colas intermedio (QMGRA)
- Al conectarse a QMGRB con WebSphere MQ Explorer, seleccione QMGRA como el gestor de colas intermediario

En esta situación, no hay ninguna conexión directa a QMGRB desde IBM WebSphere MQ Explorer; la conexión a QMGRB se realiza a través de QMGRA.

El gestor de colas QMGRB en el Servidor2 se conecta a QMGRA en el Servidor1 utilizando canales emisor-receptor. El canal entre QMGRA y QMGRB debe configurarse de tal manera que sea posible la administración remota; consulte "Preparación de canales y colas de transmisión para la administración remota" en la página 110.

Mostrar y ocultar gestores de colas y clústeres

IBM WebSphere MQ Explorer puede mostrar más un gestor de colas a la vez. En el panel Mostrar/ocultar gestor de colas (que se puede seleccionar desde el menú del nodo de árbol Gestores de colas), puede decidir si se visualiza información sobre otra máquina (remota). Los gestores de colas locales se detectan automáticamente.

Para mostrar un gestor de colas remoto:

1. Pulse con el botón derecho del ratón en el nodo de árbol `Queue Managers` y, a continuación, seleccione *Mostrar/ocultar gestores de colas ...*.
2. Pulse **Añadir**. Aparece el panel Mostrar/ocultar gestores de colas.
3. Escriba el nombre del gestor de colas remoto y el nombre de host o dirección IP en los campos correspondientes.

   El nombre de host o dirección IP se utiliza para establecer una conexión de cliente con el gestor de colas remoto utilizando el canal de conexión de servidor predeterminado, SYSTEM.ADMIN.SVRCONN, o un canal de conexión de servidor definido por el usuario.
4. Pulse **Finalizar**.

El panel Mostrar/ocultar gestores de colas también muestra una lista de todos los gestores de colas visibles. Puede utilizar este panel para ocultar gestores de colas de la vista de navegación.

Si IBM WebSphere MQ Explorer muestra un gestor de colas que es miembro de un clúster, el clúster se detecta y se muestra automáticamente.

Para exportar la lista de gestores de colas remotos desde este panel:

1. Cierre el panel Mostrar/ocultar gestores de colas.
2. Pulse el botón derecho del ratón en el nodo **IBM WebSphere MQ**, situado en la parte superior del árbol, en el panel de navegación de WebSphere MQ Explorer y luego seleccione **Exportar configuración de MQ Explorer**.

3. Pulse **MQ Explorer > Valores de MQ Explorer**
4. Seleccione **Información de conexión > Gestores de colas remotos**.
5. Seleccione un archivo para almacenar los valores de configuración exportados.
6. Por último, pulse **Finalizar** para exportar la información de conexión de gestor de colas remoto al archivo especificado.

Para importar una lista de gestores de colas remotos:

1. Pulse el botón derecho del ratón en el nodo **IBM WebSphere MQ**, situado en la parte superior del árbol, en el panel de navegación de WebSphere MQ Explorer y luego seleccione **Importar configuración de MQ Explorer**.
2. Pulse **MQ Explorer > Valores de MQ Explorer**
3. Pulse **Examinar** y desplácese a la vía de acceso del archivo que contiene la información de conexión de gestor de colas remoto.
4. Pulse **Abrir**. Si el archivo contiene una lista de gestores de colas remotos, se selecciona el recuadro **Información de conexión > Gestores de colas remotos**.
5. Por último, pulse **Finalizar** para importar la información de conexión de gestor de colas remoto a WebSphere MQ Explorer.

Pertenencia al clúster

IBM WebSphere MQ Explorer necesita información sobre los gestores de colas que son miembros de un clúster.

Si un gestor de colas es miembro de un clúster, el nodo de árbol del clúster se llenará automáticamente.

Si gestores de colas se convierten en miembros de clústeres mientras IBM WebSphere MQ Explorer está ejecutándose, debe mantener IBM WebSphere MQ Explorer con datos de administración actualizados sobre los clústeres para que éste pueda comunicarse eficazmente con ellos y mostrar información correcta sobre los clústeres cuando se le solicite. Para ello, WebSphere MQ Explorer necesita la siguiente información:

- El nombre de un gestor de colas de depósito
- El nombre de conexión del gestor de colas de repositorio si éste se encuentra en un gestor de colas remoto.

Con esta información, WebSphere MQ Explorer puede:

- Utilizar el gestor de colas de repositorio para obtener una lista de los gestores de colas del clúster.
- Administrar los gestores de colas que sean miembros del clúster y estén en plataformas y niveles de mandatos soportados.

La administración no es posible si:

- El depósito elegido deja de estar disponible. WebSphere MQ Explorer no cambia automáticamente a un repositorio alternativo.
- No se puede contactar con el depósito elegido a través de TCP/IP.
- El repositorio elegido está ejecutándose en un gestor de colas que se está ejecutando en una plataforma y un nivel de mandatos no soportados por WebSphere MQ Explorer.

Los miembros del clúster que pueden administrarse pueden ser locales o remotos (si es posible comunicar con ellos a través de TCP/IP). IBM WebSphere MQ se conecta directamente con los gestores de colas locales que son miembros de un clúster, sin utilizar una conexión de cliente.

Conversión de datos

IBM WebSphere MQ Explorer trabaja en CCSID 1208 (UTF-8). Esto permite a IBM WebSphere MQ Explorer mostrar los datos de gestores de colas remotos correctamente. Tanto si se conecta a un gestor

de colas directamente como si lo hace a través de un gestor de colas intermedio, IBM WebSphere MQ Explorer requiere que todos los mensajes entrantes se conviertan a CCSID 1208 (UTF-8).

Se emite un mensaje de error si se intenta establecer una conexión entre IBM WebSphere MQ Explorer y un gestor de colas con un CCSID que IBM WebSphere MQ Explorer no reconoce.

Las conversaciones soportadas se describen en Conversión de página de códigos.

Seguridad en Windows

El asistente Preparar WebSphere MQ crea una cuenta de usuario especial de modo que el servicio de Windows lo puedan compartir los procesos que lo necesiten.

Un servicio de Windows se comparte entre procesos de cliente para una instalación de IBM WebSphere MQ. Se crea un servicio para cada instalación. Cada servicio se denomina `MQ_`*`InstallationName`*y tiene un nombre de visualización de `IBM WebSphere MQ(`*`InstallationName`*`)`. Antes de Version 7.1, con una sola instalación en un servidor, el servicio Windows se denominaba `MQSeriesServices` con el nombre de visualización `IBM MQSeries`.

Puesto que cada servicio se debe compartir entre sesiones de inicio de sesión interactivas y no interactivas, debe iniciar cada una bajo una cuenta de usuario especial. Puede utilizar una cuenta de usuario especial para todos los servicios o crear distintas cuentas de usuario especiales. Cada cuenta de usuario especial debe tener el derecho de usuario "Iniciar sesión como servicio"; para obtener más información, consulte “Derechos de usuario necesarios para un servicio de IBM WebSphere MQ Windows” en la página 67. Si el ID de usuario no tiene autorización para poder ejecutar el servicio, el servicio no se inicia y se devuelve un error al registro de sucesos del sistema de Windows. Normalmente, tendrá que ejecutar el asistente de preparación de IBM WebSphere MQ y configurar el ID de usuario correctamente. Sin embargo, si ha configurado el ID de usuario manualmente, es posible que pueda tener un problema que será necesario resolver.

Cuando instala IBM WebSphere MQ y ejecuta el Asistente de preparación de IBM WebSphere MQ por primera vez, crea una cuenta de usuario especial para el servicio denominada MUSR_MQADMIN con los valores y permisos necesarios, incluido "Iniciar sesión como servicio".

Para instalaciones posteriores, el Asistente de preparación de IBM WebSphere MQ crea una cuenta de usuario denominada MUSR_MQADMINx, donde x es el siguiente número disponible que representa un ID de usuario que no existe. La contraseña para MUSR_MQADMINx se genera aleatoriamente cuando se crea la cuenta y se utiliza para configurar el entorno de conexión para el servicio. La contraseña generada no caduca.

Esta cuenta de IBM WebSphere MQ MQ no se ve afectada por ninguna de las políticas de cuentas definidas en el sistema que requieren que las contraseñas se cambien después de un cierto periodo de tiempo.

La contraseña no se conoce fuera de este proceso de un solo uso y la almacena el sistema operativo de Windows en una parte segura del registro.

Utilización de Active Directory (sólo Windows)

En algunas configuraciones de red, donde las cuentas de usuario se definen en controladores de dominios que utilizan Active Directory, la cuenta de usuario local bajo la cual se ejecuta IBM WebSphere MQ podría no tener la autoridad que necesita para consultar la pertenencia a grupos de otras cuentas de usuario de dominio. El Asistente de preparación de IBM WebSphere MQ identifica si esto es así realizando pruebas y formulando preguntas al usuario sobre la configuración de red.

Si la cuenta de usuario local bajo la cual se ejecuta IBM WebSphere MQ no tiene la autoridad necesaria, el asistente de preparación de IBM WebSphere MQ solicita al usuario los detalles de cuenta de una cuenta de usuario de dominio con derechos de usuario particulares. Para ver los derechos de usuario que requiere la cuenta de usuario de dominio, consulte “Derechos de usuario necesarios para un servicio de IBM WebSphere MQ Windows” en la página 67. Una vez que el usuario ha entrado los detalles de la cuenta válidos para la cuenta de usuario de dominio en el asistente de preparación de IBM WebSphere

MQ, configura un servicio IBM WebSphere MQ Windows para que se ejecute bajo la nueva cuenta. Los detalles de la cuenta se guardan en una parte segura del Registro que no pueden leer los usuarios.

Cuando el servicio se ejecuta, se inicia un servicio IBM WebSphere MQ Windows y permanece en ejecución mientras se ejecuta el servicio. Un administrador de IBM WebSphere MQ que inicie la sesión en el servidor después de haber lanzado el servicio de Windows puede utilizar IBM WebSphere MQ Explorer para administrar gestores de colas en el servidor. Esto conecta IBM WebSphere MQ Explorer al proceso del servicio Windows existente. Estas dos acciones necesitan niveles de permiso diferentes para poder funcionar:

- El proceso de inicio necesita un permiso de inicio.
- El administrador de IBM WebSphere MQ requiere permiso de acceso.

Derechos de usuario necesarios para un servicio de IBM WebSphere MQ Windows

La siguiente tabla muestra los derechos de usuario necesarios para la cuenta de usuario local y de dominio bajo la que se ejecuta el servicio de Windows para una instalación de IBM WebSphere MQ.

Iniciar sesión como proceso por lotes	Permite que un servicio de IBM WebSphere MQ Windows se ejecute bajo esta cuenta de usuario.
Iniciar sesión como servicio	Permite a los usuarios definir el servicio de IBM WebSphere MQ Windows para iniciar una sesión utilizando la cuenta configurada.
Concluir el sistema	Permite al servicio de IBM WebSphere MQ Windows reiniciar el servidor si está configurado para hacerlo cuando falla la recuperación de un servicio.
Aumentar cuotas	Necesario para la llamada `CreateProcessAsUser` del sistema operativo.
Actuar como parte del sistema operativo	Necesario para la llamada `LogonUser` del sistema operativo.
Eludir la comprobación cruzada	Necesario para la llamada `LogonUser` del sistema operativo.
Sustituir una señal de nivel de proceso	Necesario para la llamada `LogonUser` del sistema operativo.

Nota: Es posible que se necesiten derechos para depurar programas en entornos que ejecutan aplicaciones ASP e IIS.

Su cuenta de usuario de dominio debe tener estos derechos de usuario Windows establecidos como derechos de usuario efectivos tal como se indica en la aplicación Directiva de seguridad local. Si no lo están, establézcalos utilizando la aplicación Directiva de seguridad local localmente en el servidor, o utilizando la aplicación Directiva de seguridad de dominio para todo el dominio.

Cambio del nombre de usuario asociado al servicio IBM WebSphere MQ

Es posible que necesite cambiar el nombre de usuario asociado al servicio IBM WebSphere MQ de MUSR_MQADMIN a cualquier otro nombre. Por ejemplo, tal vez tenga que hacerlo si el gestor de colas está asociado a DB2, donde no se aceptan nombres de usuario de más de 8 caracteres.

Procedimiento

1. Cree una nueva cuenta de usuario (por ejemplo, **NEW_NAME**)
2. Utilice el Asistente de preparación de IBM WebSphere MQ para entrar los detalles de la nueva cuenta de usuario.

Cambio de la contraseña de la cuenta de usuario de servicio de IBM WebSphere MQ Windows

Acerca de esta tarea
Para cambiar la contraseña de la cuenta de usuario local del servicio IBM WebSphere MQ Windows, realice los pasos siguientes:

Procedimiento
1. Identifique el usuario en el que se ejecuta el servicio.
2. Detenga el servicio IBMIBM WebSphere MQ desde el panel Administración de equipos.
3. Cambie la contraseña necesaria igual que lo haría con una contraseña personal.
4. Vaya a las propiedades del servicio de IBM WebSphere MQ desde el panel Administración de equipos.
5. Seleccione la página **Iniciar sesión**.
6. Confirme que el nombre de cuenta especificado coincida con el usuario para el que se ha modificado la contraseña.
7. Escriba la contraseña en los campos **Contraseña** y **Confirmar contraseña** y pulse **Aceptar**.

El servicio de IBM WebSphere MQ Windows para una instalación que se ejecuta bajo una cuenta de usuario de dominio

Acerca de esta tarea
Si el servicio IBM WebSphere MQ Windows para una instalación se ejecuta bajo una cuenta de usuario de dominio, también puede cambiar la contraseña para la cuenta del modo siguiente:

Procedimiento
1. Cambie la contraseña de la cuenta de dominio en el controlador de dominio. Es posible que debe pedirle al administrador del sistema que realice esta tarea.
2. Siga los pasos para modificar la página **Iniciar sesión** para el servicio de IBMIBM WebSphere MQ.

   La cuenta de usuario bajo la cual se ejecute el servicio IBM WebSphere MQ Windows ejecuta los mandatos MQSC que han emitido las aplicaciones de la interfaz de usuario, o que se han realizado automáticamente durante el arranque del sistema, la conclusión o la recuperación del servicio. Por lo tanto, esta cuenta de usuario debe tener derechos de administración de IBM WebSphere MQ. De forma predeterminada, se añade al grupo **mqm** local del servidor. Si se elimina esta pertenencia, el servicio IBM WebSphere MQ Windows no funciona. Para obtener más información sobre derechos de usuario, consulte “Derechos de usuario necesarios para un servicio de IBM WebSphere MQ Windows” en la página 67

   Si surge un problema de seguridad con la cuenta de usuario bajo la que se ejecuta el servicio IBM WebSphere MQ Windows, aparecerán mensajes de error y descripciones en el registro de sucesos del sistema.

Conceptos relacionados

“Utilización de Active Directory (sólo Windows)” en la página 66
En algunas configuraciones de red, donde las cuentas de usuario se definen en controladores de dominios que utilizan Active Directory, la cuenta de usuario local bajo la cual se ejecuta IBM WebSphere MQ podría no tener la autoridad que necesita para consultar la pertenencia a grupos de otras cuentas de usuario de dominio. El Asistente de preparación de IBM WebSphere MQ identifica si esto es así realizando pruebas y formulando preguntas al usuario sobre la configuración de red.

Coordinación de IBM WebSphere MQ con Db2 como gestor de recursos

Si inicia los gestores de colas desde IBM WebSphere MQ Explorer o utiliza IBM WebSphere MQ V7 y tiene problemas para poder coordinar Db2, compruebe los registros de errores del gestor de colas.

Compruebe los registros de errores del gestor de colas para ver si hay un error como el siguiente:

```
23/09/2008 15:43:54 - Process(5508.1) User(MUSR_MQADMIN) Program(amqzxma0.exe)
Host(HOST_1) Installation(Installation1)
VMRF(7.1.0.0) QMgr(A.B.C)
AMQ7604: The XA resource manager 'DB2 MQBankDB database' was not available when called
    for xa_open. The queue manager is continuing without this resource manager.
```

Explicación: el ID de usuario (el nombre predeterminado es MUSR_MQADMIN) que ejecuta el proceso del servicio de IBM WebSphere MQ `amqsvc.exe` se sigue ejecutando con una señal de acceso que no contiene información de pertenencia a grupos para el grupo DB2USERS.

Solución: Cuando se haya asegurado de que el ID de usuario del Servicio IBM WebSphere MQ es miembro de DB2USERS, utilice la siguiente secuencia de mandatos:

- detenga el servicio.
- detenga cualquier proceso adicional que se ejecute bajo el mismo ID de usuario.
- reinicie estos procesos.

El reinicio de la máquina garantizará los pasos anteriores, pero no es necesario hacerlo.

Ampliación de IBM WebSphere MQ Explorer

IBM WebSphere MQ para Windowsy IBM WebSphere MQ para Linux (plataformasx86 y x86-64) proporcionan una interfaz de administración denominada IBM WebSphere MQ Explorer para realizar tareas de administración como alternativa a la utilización de mandatos de control o MQSC.

Esta información se aplica únicamente a WebSphere MQ for Windows y WebSphere MQ for Linux (plataformasx86 y x86-64).

IBM WebSphere MQ Explorer presenta la información en un estilo coherente con el de la infraestructura de Eclipse y las aplicaciones de plug-in que soporta Eclipse.

Mediante la ampliación de IBM WebSphere MQ Explorer, los administradores del sistema tienen la posibilidad de personalizar WebSphere MQ Explorer para mejorar la forma en que administran WebSphere MQ.

Para obtener más información, consulte *Ampliación de IBM WebSphere MQ Explorer* en la documentación de producto IBM WebSphere MQ Explorer.

Utilizar la aplicación de barra de tareas de IBM WebSphere MQ (solo Windows)

La aplicación de barra de tareas de IBM WebSphere MQ muestra un icono en la bandeja del sistema Windows del servidor. El icono le proporciona el estado actual de IBM WebSphere MQ y un menú desde el que puede ejecutar algunas acciones simples.

En Windows, el icono de WebSphere MQ está en la bandeja del sistema del servidor y está cubierto por un símbolo de estado diferenciado con colores que puede tener uno de los siguientes significados:

Verde
Funciona correctamente, no hay alertas en este momento

Azul
Indeterminado; WebSphere MQ se está iniciando o cerrando

Amarillo
Alerta; uno o más servicios están fallando o ya han fallado.

Para visualizar el menú, pulse el botón derecho del ratón sobre el icono de WebSphere MQ. Desde el menú puede realizar las acciones siguientes:

- Pulse **Abrir** para abrir el supervisor de alertas de WebSphere MQ
- Pulse **Salir** para salir de la aplicación Barra de tareas de WebSphere MQ
- Pulse **WebSphere MQ Explorer** para iniciar IBM WebSphere MQ Explorer
- Pulse **Detener WebSphere MQ** para detener WebSphere MQ
- Pulse **Acerca de WebSphere MQ** para visualizar la información sobre el monitor de alertas de WebSphere MQ

Aplicación de supervisor de alertas de IBM WebSphere MQ (solo Windows)

El supervisor de alertas de IBM WebSphere MQ es una herramienta de detección de errores que identifica y registra problemas con IBM WebSphere MQ en una máquina local.

El supervisor de alertas muestra información sobre el estado actual de la instalación local de un servidor WebSphere MQ. También supervisa la Interfaz avanzada de configuración y energía (ACPI) de Windows y asegura que se utilicen los valores de la ACPI.

Desde el supervisor de alertas de WebSphere MQ, puede:

- Acceder a IBM WebSphere MQ Explorer directamente
- Ver información relacionada con todas las alertas pendientes
- Concluir el servicio WebSphere MQ en la máquina local
- Dirigir mensajes de alerta a través de la red a una cuenta de usuario configurable o a una estación de trabajo o servidor de Windows.

Administración de objetos de IBM WebSphere MQ locales

En esta sección se explica cómo administrar objetos de IBM WebSphere MQ locales para dar soporte a los programas de aplicación que utilizan la Interfaz de cola de mensajes (MQI). En este contexto, administración local significa crear, visualizar, cambiar, copiar y suprimir objetos de IBM WebSphere MQ.

Además de los enfoques que se describen con detalle en este apartado, puede utilizar IBM WebSphere MQ Explorer para administrar los objetos locales de WebSphere MQ; consulte “Administración utilizando IBM WebSphere MQ Explorer” en la página 57.

Este apartado contiene la información siguiente:

- Programas de aplicación que utilizan la MQI
- “Realización de tareas de administración local utilizando mandatos MQSC” en la página 74
- “Trabajar con gestores de colas” en la página 83
- “Trabajar con colas locales” en la página 85
- “Trabajar con colas alias” en la página 90
- “Trabajar con colas modelo” en la página 92
- “Trabajar con servicios” en la página 98
- “Gestión de objetos para desencadenamiento” en la página 105

Iniciar y detener un gestor de colas

Utilice este tema como introducción para detener e iniciar un gestor de colas.

Inicio de un gestor de colas

Para iniciar un gestor de colas, utilice el mandato `strmqm` de la siguiente manera:

```
strmqm saturn.queue.manager
```

En sistemas WebSphere MQ para Windows y WebSphere MQ para Linux (plataformasx86 y x86-64), puede iniciar un gestor de colas como se indica a continuación:

1. Abra IBM WebSphere MQ Explorer.
2. Seleccione el gestor de colas en la vista de navegador.
3. Pulse `Start`. El gestor de colas se inicia.

Si el inicio del gestor de colas dura más de unos segundos, WebSphere MQ emite mensajes informativos de forma intermitente detallando el progreso del inicio.

El mandato `strmqm` no devolverá el control hasta que el gestor de colas que se haya iniciado y esté preparado para aceptar solicitudes de conexión.

Inicio automático de un gestor de colas

En WebSphere MQ para Windows, puede iniciar un gestor de colas automáticamente cuando se inicia el sistema utilizando WebSphere MQ Explorer. Para obtener más información, consulte “Administración utilizando IBM WebSphere MQ Explorer” en la página 57.

Detención de un gestor de colas

Para detener un gestor de colas, se utiliza el mandato **endmqm**.

Nota: Debe utilizar el mandato **endmqm** desde la instalación asociada al gestor de colas con el que está trabajando. Puede averiguar con qué instalación está asociado un gestor de colas mediante el mandato `dspmq -o installation`.

Por ejemplo, para detener un gestor de colas llamado QMB, entre el siguiente mandato:

```
endmqm QMB
```

En sistemas WebSphere MQ para Windows y WebSphere MQ para Linux (plataformasx86 y x86-64), puede detener un gestor de colas de la siguiente manera:

1. Abra IBM WebSphere MQ Explorer.
2. Seleccione el gestor de colas en la vista de navegador.
3. Pulse `Stop...`. Aparece el panel Finalizar el gestor de colas.
4. Seleccione Controlada o Inmediata.
5. Pulse OK. El gestor de colas se detiene.

Conclusión progresiva

De forma predeterminada, el mandato **endmqm** realiza una conclusión por desactivación temporal del gestor de colas especificado. Esta operación puede tardar un poco en completarse. Una conclusión progresiva espera hasta que todas las aplicaciones conectadas se hayan desconectado.

Utilice este tipo de conclusión para comunicar a las aplicaciones que deben detenerse. Si emite:

```
endmqm -c QMB
```

no recibirá ninguna notificación cuando se hayan detenido todas las aplicaciones. (Un mandato `endmqm -c QMB` es equivalente a un mandato `endmqm QMB` .)

Sin embargo, si emite:

```
endmqm -w QMB
```

el mandato esperará hasta que todas las aplicaciones se hayan detenido y el gestor de colas haya finalizado.

Conclusión inmediata

Para una conclusión inmediata, se permite que terminen las llamadas MQI actuales, pero las llamadas nuevas no se realizarán correctamente. Este tipo de conclusión no espera a que las aplicaciones se desconecten del gestor de colas.

Para una conclusión inmediata, escriba:

```
endmqm -i QMB
```

Conclusión preferente

Nota: No utilice este método a menos que todos los demás intentos de detener el gestor de colas con el mandato **endmqm** hayan fracasado. Este método puede tener consecuencias imprevisibles para las aplicaciones conectadas.

Si una conclusión inmediata no funciona, deberá recurrir a una conclusión *preferente*, especificando el indicador `-p`. Por ejemplo:

```
endmqm -p QMB
```

Este mandato detiene el gestor de colas inmediatamente. Si este método tampoco funciona, busque una solución alternativa en “Detención manual de un gestor de colas” en la página 72.

Para ver una explicación detallada del mandato **endmqm** y sus opciones, consulte el apartado endmqm.

Problemas con la conclusión de un gestor de colas

A menudo, la causa de los problemas que pueden surgir al concluir un gestor de colas provienen de las aplicaciones. Por ejemplo, cuando las aplicaciones:

- No comprueban adecuadamente los códigos de retorno MQI
- No solicitan notificación de una conclusión progresiva
- Terminan sin desconectarse del gestor de colas (emitiendo una llamada `MQDISC`)

Si se produce un problema al detener el gestor de colas, salga del mandato **endmqm** utilizando Control+C. Después, puede emitir otro mandato **endmqm**, pero esta vez con un indicador que especifique el tipo de conclusión que desea realizar.

Detención manual de un gestor de colas

Si los métodos empleados habitualmente para detener gestores de colas fracasan, puede recurrir a los métodos aquí descritos.

La forma normal de detener los gestores de colas es utilizar el mandato `endmqm`. Para detener un gestor de colas manualmente, utilice uno de los procedimientos que se describen en este apartado. Para ver detalles sobre cómo realizar operaciones en los gestores de colas utilizando los mandatos de control, consulte Creación y gestión de gestores de colas.

Detención de gestores de colas en Windows

Cómo finalizar los procesos y el servicio de IBM WebSphere MQ para detener los gestores de colas de IBM WebSphere MQ para Windows.

Para detener un gestor de colas que se ejecute bajo WebSphere MQ para Windows:

1. Enumere los nombres (ID) de los procesos que están ejecutándose actualmente utilizando el Administrador de tareas de Windows.
2. Finalice los procesos utilizando el Gestor de tareas de Windows , o el mandato **taskkill** , en el orden siguiente (si se están ejecutando):

AMQZMUC0	Gestor de procesos críticos
AMQZXMA0	Controlador de ejecución
AMQZFUMA	Proceso del OAM
AMQZLAA0	Agentes LQM
AMQZLSA0	Agentes LQM
AMQZMUF0	Gestor de programas de utilidad
AMQZMGR0	Controlador de procesos
AMQZMUR0	Gestor de procesos reiniciable
AMQFQPUB	Proceso de Publicación/suscripción
AMQFCXBA	Proceso de operador de intermediario
AMQRMPPA	Proceso de agrupación de procesos
AMQCRSTA	Proceso de trabajo de respuesta sin hebra
AMQCRS6B	Conexión de cliente y canal receptor LU62
AMQRRMFA	Proceso de depósito (para clústeres)
AMQZDMAA	Procesador de mensajes diferidos
AMQPCSEA	El servidor de mandatos
RUNMQTRM	Invocar un supervisor de activación para un servidor
RUNMQDLQ	Invocar manejador de la cola de mensajes no entregados
RUNMQCHI	El proceso de iniciado de canal
RUNMQLSR	El proceso de escucha de canal
AMQXSSVN	Servidores de memoria compartida
AMQZTRCN	Rastreo

3. Detenga el servicio WebSphere MQ desde **Herramientas de administración** > **Servicios** en el Panel de control de Windows.
4. Si ha probado todos los métodos y el gestor de colas no se ha detenido, reinicie el sistema.

El Administrador de tareas de Windows y el mandato **tasklist** proporcionan información limitada sobre las tareas. Para obtener más información sobre cómo determinar qué procesos están relacionados con un gestor de colas específico, puede utilizar una herramienta como *Process Explorer* (procexp.exe), que puede descargarse desde el sitio web de Microsoft, https://www.microsoft.com.

Detención de gestores de colas en sistemas UNIX and Linux

Cómo finalizar los procesos y el servicio de IBM WebSphere MQ para detener los gestores de colas de IBM WebSphere MQ para UNIX and Linux. Puede probar los métodos descritos en este tema si los métodos estándar para detener y eliminar gestores de colas no funcionan.

Para detener un gestor de colas que se ejecute bajo WebSphere MQ en sistemas UNIX and Linux:

1. Busque los ID de proceso de los programas del gestor de colas que estén ejecutándose aún, utilizando el mandato ps. Por ejemplo, si el nombre del gestor de colas es QMNAME, puede utilizar el siguiente mandato:

   ```
   ps -ef | grep QMNAME
   ```

2. Termine todos los procesos del gestor de colas que sigan ejecutándose. Utilice el mandato kill, especificando los ID de proceso encontrados mediante el mandato ps.

   Finalice los procesos en el orden siguiente:

amqzmuc0	Gestor de procesos críticos
amqzxma0	Controlador de ejecución
amqzfuma	Proceso del OAM
amqzlaa0	Agentes LQM
amqzlsa0	Agentes LQM
amqzmuf0	Gestor de programas de utilidad
amqzmur0	Gestor de procesos reiniciable
amqzmgr0	Controlador de procesos
amqfqpub	Proceso de Publicación/suscripción
amqfcxba	Proceso de operador de intermediario
amqrmppa	Proceso de agrupación de procesos
amqcrsta	Proceso de trabajo de respuesta sin hebra
amqcrs6b	Conexión de cliente y canal receptor LU62
amqrrmfa	Proceso de depósito (para clústeres)
amqzdmaa	Procesador de mensajes diferidos
amqpcsea	El servidor de mandatos
runmqtrm	Invocar un supervisor de activación para un servidor
runmqdlq	Invocar manejador de la cola de mensajes no entregados
runmqchi	El proceso de iniciado de canal
runmqlsr	El proceso de escucha de canal

Nota: Puede utilizar el mandato **kill -9** para finalizar los procesos que no se han detenido.

Si detiene el gestor de colas manualmente, es posible que se tomen FFST y que los archivos FDC colocados en /var/mqm/errors. no consideren esto como un defecto en el gestor de colas.

El gestor de colas se reiniciará normalmente, incluso después de haberlo detenido utilizando este método.

Realización de tareas de administración local utilizando mandatos MQSC

Esta sección presenta los mandatos MQSC y explica cómo utilizarlos para algunas tareas comunes.

Si utiliza IBM WebSphere MQ para Windows o IBM WebSphere MQ para Linux (plataformasx86 y x86-64), también puede realizar las operaciones descritas en esta sección utilizando IBM WebSphere MQ Explorer. Consulte “Administración utilizando IBM WebSphere MQ Explorer” en la página 57 para obtener más información.

Puede utilizar mandatos MQSC para gestionar objetos de gestor de colas, incluido el propio gestor de colas, colas, definiciones de proceso, canales, canales de conexión de cliente, escuchas, servicios, listas de nombres, clústeres y objetos de información de autenticación. En esta sección se tratan los gestores

de colas, las colas y las definiciones de proceso; para obtener información sobre cómo administrar el canal, el canal de conexión de cliente y los objetos de escucha, consulte Objetos . Si desea información sobre todos los mandatos MQSC para gestionar objetos de gestor de colas, consulte "Mandatos (MQSC) de script" en la página 75.

Para emitir mandatos MQSC a un gestor de colas se utiliza el mandato `runmqsc`. (Para conocer detalles sobre este mandato, consulte runmqsc). Esto puede hacerse interactivamente, entrando los mandatos en el teclado o puede redireccionarse el dispositivo de entrada estándar (`stdin`) para que ejecute una secuencia de mandatos de un archivo de texto ASCII. En ambos casos, el formato de los mandatos es el mismo. (Para obtener más información sobre cómo ejecutar los mandatos desde un archivo de texto, consulte "Ejecución de mandatos MQSC desde archivos de texto" en la página 79).

Puede ejecutar el mandato `runmqsc` de tres formas, dependiendo de los indicadores definidos en el mandato:

- Verificar un mandato sin ejecutarlo, en la que los mandatos MQSC se verifican en un gestor de colas local pero no se ejecutan.
- Ejecutar un mandato en un gestor de colas local, en la que los mandatos MQSC se ejecutan en un gestor de colas local.
- Ejecutar un mandato en un gestor de colas remoto, en la que los mandatos MQSC se ejecutan en un gestor de colas remoto.

También puede ejecutar el mandato seguido de un signo de interrogación para visualizar la sintaxis.

Los atributos de objeto especificados en los mandatos MQSC aparecen en mayúsculas en esta sección (por ejemplo RQMNAME), aunque no son sensibles a las mayúsculas y minúsculas. Los nombres de atributo de los mandatos MQSC están limitados a ocho caracteres. Los mandatos MQSC están disponibles en otras plataformas, incluyendo IBM i y z/OS.

Los mandatos MQSC están descritos en la colección de temas de la sección Consulta de MQSC.

Mandatos (MQSC) de script

Los mandatos MQSC proporcionan un método uniforme para emitir mandatos legibles por personas en las plataformas de WebSphere MQ. Para obtener más información sobre los mandatos de *formato de mandato programable* (PCF), consulte "Introducción a los Formatos de mandato programable" en la página 9.

El formato general de los mandatos se muestra en Mandatos MQSC.

Para utilizar mandatos MQSC debe respetar las siguientes reglas:

- Cada mandato empieza por un parámetro primario (un verbo) y a éste le sigue un parámetro secundario (un sustantivo). Después le sigue el nombre o nombre genérico del objeto (entre paréntesis) si es que lo hay, como suele ocurrir en la mayoría de los mandatos. A continuación, los parámetros pueden aparecer, normalmente, en cualquier orden; si un parámetro tiene un valor correspondiente, éste debe aparecer directamente después del parámetro con el que está relacionado.
- Las palabras clave, los paréntesis y los valores pueden estar separados por el número de blancos y de comas que se desee. Una coma mostrada en los diagramas de sintaxis siempre puede ser reemplazada por uno o más espacios en blanco. Debe haber como mínimo un blanco inmediatamente antes de cada parámetro (después del parámetro primario).
- Al principio o final de un mandato puede aparecer un número cualquiera de espacios en blanco, así como entre parámetros, signos de puntuación y valores. Por ejemplo, el mandato siguiente es válido:

  ```
  ALTER QLOCAL  ('Account'  )           TRIGDPTH  (  1)
  ```

  Los blancos dentro de un par de comillas son significativos.
- Las comas adicionales pueden aparecer en cualquier lugar donde estén permitidos los espacios en blanco y se tratan como si fueran espacios en blanco (a menos, por supuesto, que estén dentro de series entre comillas).

- Los parámetros repetidos no están permitidos. Repetir un parámetro con su versión "NO", como en REPLACE NOREPLACE, tampoco está permitido.
- Las series que contienen blancos, caracteres en minúsculas o caracteres especiales distintos de:
  - Punto (.)
  - Barra inclinada (/)
  - Subrayado (_)
  - Signo de porcentaje (%)

  deben estar entre comillas simples, a menos que:
  - Sean valores genéricos que terminen con un asterisco
  - Sean un solo asterisco (por ejemplo, TRACE(*))
  - Se trate de una especificación de rango que contenga dos puntos (por ejemplo, CLASS(01:03))

  Si la cadena propiamente dicha contiene comillas simples, las comillas simples se representan con dos comillas simples.Los caracteres en minúsculas que no estén entre comillas se convertirán en mayúsculas.
- En plataformas distintas de z/OS, una serie que no contiene caracteres (es decir, dos comillas simples sin espacio entre ellas) se interpreta como un espacio en blanco entre comillas simples, es decir, se interpreta de la misma forma que (''). La excepción a esto es si el atributo que se utiliza es uno de los siguientes:
  - TOPICSTR
  - SUB
  - USERDATA
  - SELECTOR

  entonces, las dos comillas simples sin ningún espacio en blanco entre ellas se interpretaría como una cadena de caracteres de longitud cero.
- En v7.0, cualquier espacio en blanco final en estos atributos de serie que se basan en tipos MQCHARV como, por ejemplo, SELECTOR, datos de subusuario, se trata como importante que significa que 'abc ' no equivale a 'abc'.
- Un paréntesis izquierdo seguido por un paréntesis derecho sin información significativa, como por ejemplo

  ```
  NAME ( )
  ```

  no es válido a menos que se indique específicamente lo contrario.
- Las palabras clave no son sensibles a las mayúsculas y minúsculas: AltER, alter y ALTER son todas correctas.Lo que no esté entre comillas se convertirá en mayúsculas.
- Se han definido sinónimos para algunos parámetros.Por ejemplo, DEF siempre es un sinónimo de DEFINE; por lo tanto, DEF QLOCAL es válido. Sin embargo, los sinónimos no son simplemente palabras abreviadas; DEFI no es un sinónimo de DEFINE.

  Nota: No hay ningún sinónimo para el parámetro DELETE. Esto es así para evitar borrar accidentalmente objetos al utilizar DEF, el sinónimo de DEFINE.

Si desea una visión general del uso de mandatos MQSC para administrar IBM WebSphere MQ, consulte “Realización de tareas de administración local utilizando mandatos MQSC” en la página 74.

Los mandatos MQSC utilizan determinados caracteres especiales que tienen determinados significados. Si desea más información sobre estos caracteres especiales y cómo utilizarlos, consulte Caracteres con significados especiales.

Para averiguar cómo puede crear scripts utilizando mandatos MQSC, consulte Creación de scripts de mandatos.

Para obtener la lista completa de los mandatos MQSC, consulte Los mandatos MQSC.

Tareas relacionadas
Creación de scripts de mandatos

Nombres de objetos WebSphere MQ

Cómo utilizar los nombres de objeto en los mandatos MQSC.

En los ejemplos se han utilizado nombres largos para los objetos. Esto le ayudará a identificar el tipo de objeto con el que está trabajando.

Cuando emita mandatos MQSC, sólo necesita especificar el nombre local de la cola. En los ejemplos de este manual se utilizan nombres de cola tales como:

```
ORANGE.LOCAL.QUEUE
```

La parte `LOCAL.QUEUE` del nombre sirve para indicar que esta es una cola local. ***No*** es necesario para los nombres de colas locales en general.

También se utiliza el nombre `saturn.queue.manager` como nombre de gestor de colas. La parte `queue.manager` del nombre sirve para indicar que este objeto es un gestor de colas. *No* es necesario para los nombres de gestores de colas en general.

Distinción entre mayúsculas y minúsculas en los mandatos MQSC

Los mandatos MQSC, incluidos sus atributos, pueden escribirse en mayúsculas o en minúsculas. Los nombres de objetos en los mandatos MQSC se convierten en mayúsculas (es decir, QUEUE y queue no se diferencian), a menos que los nombres se encierren entre comillas simples. Si no se utilizan comillas, el objeto se procesa con un nombre en mayúsculas. Consulte Referencia de MQSC para obtener más información.

La invocación del mandato `runmqsc`, al igual que todos los mandatos de control de WebSphere MQ, es sensible a las mayúsculas y minúsculas en algunos entornos WebSphere MQ. Consulte Utilización de mandatos de control para obtener más información.

Entrada y salida estándar

El *dispositivo de entrada estándar*, llamado también `stdin`, es el dispositivo del que se toma la entrada para el sistema. Normalmente es el teclado, pero puede especificar que la entrada puede proceder de un puerto serie o de un archivo de disco, por ejemplo. El *dispositivo de salida estándar*, llamado también `stdout`, es el dispositivo al que se envía la salida del sistema. Normalmente es una pantalla, pero puede redirigir la salida a un puerto serie o a un archivo.

En mandatos del sistema operativo y mandatos de control de WebSphere MQ , el operador < redirige la entrada. Si este operador va seguido de un nombre de archivo, la entrada se toma de dicho archivo. Del mismo modo, el operador > redirecciona la salida; si este operador va seguido de un nombre de archivo, la salida se redireccionará a dicho archivo.

Utilización de mandatos MQSC de forma interactiva

Puede utilizar los mandatos MQSC de forma interactiva, mediante una ventana de mandatos o shell.

Para utilizar mandatos MQSC interactivamente, abra un shell o una ventana de mandatos y entre:

```
runmqsc
```

Como en este mandato no se ha especificado un nombre de gestor de colas, los mandatos MQSC los procesa el gestor de colas predeterminado. Si desea utilizar un gestor de colas diferente, especifique el

nombre del gestor de colas en el mandato **runmqsc**. Por ejemplo, para ejecutar mandatos MQSC en el gestor de colas `jupiter.queue.manager`, utilice el mandato:

```
runmqsc jupiter.queue.manager
```

Después de esto, todos los mandatos MQSC que entre los procesará este gestor de colas, suponiendo que esté en el mismo nodo y ya esté en ejecución.

Ahora ya puede escribir cualquier mandato MQSC, según necesite. Por ejemplo, pruebe con este:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE)
```

En los mandatos que tienen demasiados parámetros y no caben en una sola línea, utilice los caracteres de continuación para indicar que un mandato continúa en la línea siguiente:

- Un signo menos (-) indica que el mandato debe continuar desde el principio de la siguiente línea.
- Un signo más (+) indica que el mandato debe continuar desde el primer carácter de la línea siguiente que no sea un blanco.

La entrada de mandatos termina con el último carácter que no sea un carácter de continuación de una línea que no esté en blanco. También puede terminar explícitamente la entrada de un mandato con un punto y coma (;). (Esto es especialmente útil si accidentalmente entra un carácter de continuación al final de la última línea de la entrada de mandato).

Información de retorno de los mandatos MQSC

Cuando emite mandatos MQSC, el gestor de colas devuelve mensajes de operador que confirman sus acciones o que le indican los errores que ha cometido. Por ejemplo:

```
AMQ8006: WebSphere MQ queue created.
```

Este mensaje confirma que se ha creado una cola.

```
AMQ8405: Syntax error detected at or near end of command segment below:-

AMQ8426: Valid MQSC commands are:


    ALTER
    CLEAR
    DEFINE
    DELETE
    DISPLAY
    END
    PING
    REFRESH
    RESET
    RESOLVE
    RESUME
    START
    STOP
    SUSPEND
     4 : end
```

Este mensaje indica que ha cometido un error de sintaxis.

Estos mensajes se envían al dispositivo de salida estándar. Si no ha entrado el mandato correctamente, consulte la publicación Referencia de MQSC para ver la sintaxis correcta.

Finalizar la entrada interactiva de mandatos MQSC

Para dejar de trabajar con mandatos MQSC, entre el mandato END.

Alternativamente, puede utilizar el carácter EOF correspondiente al sistema operativo.

Ejecución de mandatos MQSC desde archivos de texto

La ejecución interactiva de mandatos MQSC es adecuada para pruebas rápidas, pero si tiene mandatos muy largos o utiliza una secuencia determinada de mandatos repetidamente, quizá sea conveniente redirigir `stdin` desde un archivo de texto.

"Entrada y salida estándar" en la página 77 contiene información sobre `stdin`y `stdout`. Para redirigir `stdin` desde un archivo de texto, cree primero un archivo de texto que contenga los mandatos MQSC utilizando su editor de textos habitual. Si utiliza el mandato `runmqsc`, use los operadores de redirección. Por ejemplo, el mandato siguiente ejecuta una secuencia de mandatos contenida en el archivo de texto `myprog.in`:

```
runmqsc < myprog.in
```

De forma similar, también puede redirigir la salida a un archivo. Un archivo que contiene los mandatos MQSC para su entrada se denomina un *archivo de mandatos MQSC*. El archivo de salida que contiene las respuestas del gestor de colas se denomina el *archivo de salida*.

Para redirigir `stdin` y `stdout` en el mandato `runmqsc`, utilice esta forma del mandato:

```
runmqsc < myprog.in > myprog.out
```

Este mandato invoca los mandatos MQSC contenidos en el archivo de mandatos MQSC `myprog.in`. Debido a que no se ha especificado un nombre de gestor de colas, los mandatos MQSC se ejecutan en el gestor de colas predeterminado. La salida se envía al archivo de texto `myprog.out`.Figura 12 en la página 80 muestra un extracto del archivo de mandatos MQSC `myprog.in` y Figura 13 en la página 81 muestra el extracto correspondiente de la salida en `myprog.out`.

Para redirigir `stdin` y `stdout` en el mandato `runmqsc`, para un gestor de colas (`saturn.queue.manager`) que no es el gestor de colas predeterminado, utilice esta forma del mandato:

```
runmqsc saturn.queue.manager < myprog.in > myprog.out
```

Archivos de mandatos MQSC

Los mandatos MQSC se escriben en formato que puede leer el usuario, es decir, en texto ASCII.Figura 12 en la página 80 es un extracto de un archivo de mandatos MQSC que muestra un mandato MQSC (DEFINE QLOCAL) con sus atributos. En el manual Referencia de MQSC hay una descripción de todos los mandatos MQSC y de su sintaxis.

```
 .
 .
 .
DEFINE QLOCAL(ORANGE.LOCAL.QUEUE) REPLACE  +
       DESCR(' ') +
       PUT(ENABLED) +
       DEFPRTY(0) +
       DEFPSIST(NO) +
       GET(ENABLED) +
       MAXDEPTH(5000) +
       MAXMSGL(1024) +
       DEFSOPT(SHARED) +
       NOHARDENBO +
       USAGE(NORMAL) +
       NOTRIGGER;
 .
 .
 .
```

Figura 12. Extracto de un archivo de mandatos MQSC

Por motivos de portabilidad entre los entornos WebSphere MQ, limite la longitud de línea en los archivos de mandatos MQSC a 72 caracteres. El signo más indica que el mandato continúa en la línea siguiente.

Informes de mandatos MQSC

El mandato **runmqsc** devuelve un *informe*, que se envía a `stdout`. El informe contiene:

- Una cabecera que identifica los mandatos MQSC como el origen del informe:

  ```
  Starting MQSC for queue manager jupiter.queue.manager.
  ```

  Donde `jupiter.queue.manager` es el nombre del gestor de colas.
- Un listado numerado opcional de los mandatos MQSC emitidos. De forma predeterminada, el texto de la entrada se repite en la salida. En esta salida, cada mandato va precedido por un número de secuencia, tal como se muestra en la Figura 13 en la página 81. Sin embargo, puede utilizar el indicador -e en el mandato `runmqsc` para suprimir la salida.
- Un mensaje de error de sintaxis para todos los mandatos erróneos que se encuentren.
- Un *mensaje de operador* que indica el resultado de ejecutar cada mandato. Por ejemplo, el mensaje de operador que indica que un mandato DEFINE QLOCAL se ha llevado a cabo correctamente es el siguiente:

  ```
  AMQ8006: WebSphere MQ queue created.
  ```
- Otros mensajes que son el resultado de errores generales al ejecutar el archivo de script.
- Un breve resumen estadístico del informe que indica el número de mandatos leídos, el número de mandatos con errores de sintaxis y el número de mandatos que no se han podido procesar.

  Nota: El gestor de colas intenta procesar únicamente los mandatos que no tienen errores de sintaxis.

```
Starting MQSC for queue manager jupiter.queue.manager.
 .
 .
    12:     DEFINE QLOCAL('ORANGE.LOCAL.QUEUE') REPLACE  +
      :            DESCR(' ') +
      :            PUT(ENABLED) +
      :            DEFPRTY(0) +
      :            DEFPSIST(NO) +
      :            GET(ENABLED) +
      :            MAXDEPTH(5000) +
      :            MAXMSGL(1024) +
      :            DEFSOPT(SHARED) +
      :            NOHARDENBO +
      :            USAGE(NORMAL) +
      :            NOTRIGGER;
AMQ8006: WebSphere MQ queue created.
      :
 .
 .
```

Figura 13. Extracto de un archivo de informe de mandatos MQSC

Ejecución de los archivos de mandatos MQSC suministrados

Los archivos de mandatos MQSC siguientes se suministran con WebSphere MQ:

amqscos0.tst
Definiciones de objetos utilizadas por programas de ejemplo.

amqscic0.tst
Definiciones de colas para transacciones CICS.

En WebSphere MQ para Windows, estos archivos se encuentran en el directorio *MQ_INSTALLATION_PATH*\tools\mqsc\samples.*MQ_INSTALLATION_PATH* representa el directorio de alto nivel en el que está instalado WebSphere MQ .

En sistemas UNIX and Linux, estos archivos se encuentran en el directorio *MQ_INSTALLATION_PATH*/samp.*MQ_INSTALLATION_PATH* representa el directorio de alto nivel en el que está instalado WebSphere MQ .

El mandato que los ejecuta es:

```
runmqsc < amqscos0.tst >test.out
```

Utilización de runmqsc para verificar mandatos

Puede utilizar el mandato `runmqsc` para verificar mandatos MQSC en un gestor de colas local sin ejecutarlos realmente. Para ello, establezca el indicador -v en el mandato `runmqsc`, por ejemplo:

```
runmqsc -v < myprog.in > myprog.out
```

Cuando invoca `runmqsc` para un archivo de mandatos MQSC, el gestor de colas verifica cada mandato y devuelve un informe sin ejecutar realmente los mandatos MQSC. Esto le permite comprobar la sintaxis de los mandatos de su archivo de mandatos. Esto es particularmente importante si está:

- Ejecutando un gran número de mandatos desde un archivo de mandatos.
- Utilizando un archivo de mandatos MQSC repetidas veces.

El informe que se devuelve es similar al que se muestra en la Figura 13 en la página 81.

No puede utilizar este método para verificar los mandatos MQSC de forma remota. Por ejemplo, si intenta ejecutar este mandato:

```
runmqsc -w 30 -v jupiter.queue.manager < myprog.in > myprog.out
```

el indicador -w, que se utiliza para indicar que el gestor de colas es remoto, se ignora y el mandato se ejecuta localmente en modalidad de verificación. 30 es el número de segundos que WebSphere MQ espera una respuesta del gestor de colas remoto.

Ejecución de mandatos MQSC desde archivos de proceso por lotes

Si tiene mandatos muy largos o está utilizando repetidamente una secuencia determinada de mandatos, quizás sea conveniente redirigir `stdin` desde un archivo de proceso por lotes.

Para redirigir `stdin` desde un archivo por lotes, cree primero un archivo de proceso por lotes que contenga los mandatos MQSC utilizando su editor de textos habitual. Si utiliza el mandato `runmqsc`, use los operadores de redirección. El ejemplo siguiente:

1. Crea un gestor de colas de prueba, TESTQM
2. Crea un CLNTCONN coincidente y un escucha establecido para que utilice el puerto TCP/IP 1600
3. Crea una cola de prueba, TESTQ
4. Coloca un mensaje en la cola, utilizando el programa de ejemplo amqsputc.

```
export MYTEMPQM=TESTQM
export MYPORT=1600
export MQCHLLIB=/var/mqm/qmgrs/$MQTEMPQM/@ipcc

crtmqm $MYTEMPQM
strmqm $MYTEMPQM
runmqlsr -m $MYTEMPQM -t TCP -p $MYPORT &

runmqsc $MYTEMPQM << EOF
  DEFINE CHANNEL(NTLM) CHLTYPE(SVRCONN) TRPTYPE(TCP)
  DEFINE CHANNEL(NTLM) CHLTYPE(CLNTCONN) QMNAME('$MYTEMPQM') CONNAME('hostname($MYPORT)')
  ALTER  CHANNEL(NTLM) CHLTYPE(CLNTCONN)
  DEFINE QLOCAL(TESTQ)
EOF

amqsputc TESTQ $MYTEMPQM << EOF
hello world
EOF

endmqm -i $MYTEMPQM
```

Figura 14. Script de ejemplo para ejecutar mandatos MQSC desde un archivo de proceso por lotes

Resolución de problemas con mandatos MQSC

Si no consigue que se ejecuten los mandatos MQSC, utilice la información de este tema para ver si es debido a alguno de estos problemas comunes. La causa del problema no siempre es evidente cuando se lee el error que un genera un mandato.

Si no consigue que se ejecuten los mandatos MQSC, compruebe la lista siguiente para ver si es debido a alguno de estos problemas comunes. La causa del problema no siempre es evidente cuando se lee el error generado.

Al utilizar el mandato `runmqsc`, recuerde lo siguiente:

- Utilice el operador **<** para redirigir la entrada de un archivo. Si omite este operador, el gestor de colas interpretará el nombre del archivo como si se tratase de un nombre de gestor de colas y emitirá el siguiente mensaje de error:

```
AMQ8118: WebSphere MQ queue manager does not exist.
```

- Si redirige la salida a un archivo, utilice el operador de redirección **>**. De forma predeterminada, el archivo se coloca en el directorio de trabajo actual al invocar `runmqsc`. Especifique un nombre de archivo completamente calificado para enviar la salida a un archivo y directorio específicos.
- Compruebe que ha creado el gestor de colas que va a ejecutar los mandatos, mediante el mandato siguiente para visualizar todos los gestores de colas:

```
dspmq
```

- El gestor de colas debe estar en ejecución. Si no lo está, inícielo; (consulte Inicio de un gestor de colas). Recibirá un mensaje de error si intenta iniciar un gestor de colas que ya está ejecutándose.
- Especifique un nombre de gestor de colas en el mandato `runmqsc` si no ha definido un gestor de colas predeterminado, de lo contrario aparecerá este error:

```
AMQ8146: WebSphere MQ queue manager not available.
```

- No puede especificar un mandato MQSC como parámetro del mandato `runmqsc`. Por ejemplo, esto no es válido:

```
runmqsc DEFINE QLOCAL(FRED)
```

- No puede entrar mandatos MQSC antes de emitir el mandato `runmqsc`.
- No puede ejecutar mandatos de control desde `runmqsc`. Por ejemplo, no puede emitir el mandato `strmqm` para iniciar un gestor de colas mientras está ejecutando mandatos MQSC interactivamente. Si lo hace, recibirá mensajes de error similares a los siguientes:

```
runmqsc
 .
 .
Starting MQSC for queue manager jupiter.queue.manager.

    1 : strmqm saturn.queue.manager
AMQ8405: Syntax error detected at or near end of cmd segment below:-s

AMQ8426: Valid MQSC commands are:
    ALTER
    CLEAR
    DEFINE
    DELETE
    DISPLAY
    END
    PING
    REFRESH
    RESET
    RESOLVE
    RESUME
    START
    STOP
    SUSPEND
     2 : end
```

Trabajar con gestores de colas

Ejemplos de los mandatos MQSC que se pueden utilizar para visualizar o alterar atributos del gestor de colas.

Visualización de atributos del gestor de colas

Para visualizar los atributos del gestor de colas especificado en el mandato **runmqsc**, utilice el mandato MQSC siguiente:

```
DISPLAY QMGR
```

La salida normal de este mandato aparece en la Figura 15 en la página 84

```
DISPLAY QMGR
     1 : DISPLAY QMGR
AMQ8408: Display Queue Manager details.
   QMNAME(QM1)                             ACCTCONO(DISABLED)
   ACCTINT(1800)                           ACCTMQI(OFF)
   ACCTQ(OFF)                              ACTIVREC(MSG)
   ACTVCONO (DISABLED)                     ACTVTRC (OFF)
   ALTDATE(2012-05-27)                     ALTTIME(16.14.01)
   AUTHOREV(DISABLED)                      CCSID(850)
   CHAD(DISABLED)                          CHADEV(DISABLED)
   CHADEXIT( )                             CHLEV(DISABLED)
   CLWLDATA( )                             CLWLEXIT( )
   CLWLLEN(100)                            CLWLMRUC(999999999)
   CLWLUSEQ(LOCAL)                         CMDEV(DISABLED)
   CMDLEVEL(750)                           COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)
   CONFIGEV(DISABLED)                      CRDATE(2011-05-27)
   CRTIME(16.14.01)                        DEADQ()
   DEFXMITQ( )                             DESCR( )
   DISTL(YES)                              INHIBTEV(DISABLED)
   IPADDRV(IPV4)                           LOCALEV(DISABLED)
   LOGGEREV(DISABLED)                      MARKINT(5000)
   MAXHANDS(256)                           MAXMSGL(4194304)
   MAXPROPL(NOLIMIT)                       MAXPRTY(9)
   MAXUMSGS(10000)                         MONACLS(QMGR)
   MONCHL(OFF)                             MONQ(OFF)
   PARENT( )                               PERFMEV(DISABLED)
   PLATFORM(WINDOWSNT)                        PSRTYCNT(5)
   PSNPMSG(DISCARD)                        PSNPRES(NORMAL)
   PSSYNCPT(IFPER)                         QMID(QM1_2011-05-27_16.14.01)
   PSMODE(ENABLED)                         REMOTEEV(DISABLED)
   REPOS( )                                REPOSNL( )
   ROUTEREC(MSG)                           SCHINIT(QMGR)
   SCMDSERV(QMGR)                          SSLCRLNL( )
   SSLCRYP( )                              SSLEV(DISABLED)
   SSLFIPS(NO)                             SSLKEYR(C:\Program Files\IBM\WebSphere
MQ\Data\qmgrs\QM1\ssl\key)
   SSLRKEYC(0)                             STATACLS(QMGR)
   STATCHL(OFF)                            STATINT(1800)
   STATMQI(OFF)                            STATQ(OFF)
   STRSTPEV(ENABLED)                       SYNCPT
   TREELIFE(1800)                          TRIGINT(999999999)
```

Figura 15. Salida normal de un mandato DISPLAY QMGR

El parámetro ALL es el valor predeterminado en el mandato DISPLAY QMGR. Muestra todos los atributos del gestor de colas. En particular, la salida indica el nombre del gestor de colas predeterminado, el nombre de la cola de mensajes no entregados y el nombre de la cola de mandatos.

La existencia de estas colas puede confirmarse entrando el mandato:

```
DISPLAY QUEUE (SYSTEM.*)
```

Esto muestra una lista de colas que coinciden con la raíz SYSTEM.*. Los paréntesis son obligatorios.

Modificación de los atributos del gestor de colas

Para modificar los atributos del gestor de colas especificado en el mandato **runmqsc**, utilice el mandato MQSC ALTER QMGR, especificando los atributos y valores que desee cambiar. Por ejemplo, utilice los siguientes mandatos para modificar los atributos de jupiter.queue.manager:

```
runmqsc jupiter.queue.manager
ALTER QMGR DEADQ (ANOTHERDLQ) INHIBTEV (ENABLED)
```

El mandato ALTER QMGR cambia la cola de mensajes no entregados utilizada y habilita los sucesos de inhibición.

Referencia relacionada
Atributos para el gestor de colas

Trabajar con colas locales

Esta sección contiene ejemplos de algunos mandatos MQSC que puede utilizar para gestionar colas locales, modelo y alias.

En el manual Referencia de MQSC encontrará información detallada acerca de estos mandatos.

Definición de una cola local

Para una aplicación, el gestor de colas local es el gestor de colas al que está conectada la aplicación. Las colas que están gestionadas por el gestor de colas local se dice que son locales respecto a ese gestor de colas.

Utilice el mandato MQSC DEFINE QLOCAL para crear una cola local. También puede utilizar el valor predeterminado incluido en la definición de la cola local predeterminada, o puede modificar las características de la cola de las existentes para la cola local predeterminada.

Nota: El nombre de la cola local predeterminada es SYSTEM.DEFAULT.LOCAL.QUEUE, y se ha creado durante la instalación del sistema.

Por ejemplo, el mandato DEFINE QLOCAL que se define una cola llamada ORANGE.LOCAL.QUEUE con estas características:

- Está habilitada para operaciones get, para operaciones put y funciona de acuerdo con un orden de prioridades.
- Es una cola *normal*, es decir, no es una cola de inicio ni una cola de transmisión y no genera mensajes mensaje desencadenantes.
- La profundidad de cola máxima es 5.000 mensajes; la longitud máxima de un mensaje es de 4.194.304 bytes.

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) +
       DESCR('Queue for messages from other systems') +
       PUT (ENABLED) +
       GET (ENABLED) +
       NOTRIGGER +
       MSGDLVSQ (PRIORITY) +
       MAXDEPTH (5000) +
       MAXMSGL (4194304) +
       USAGE (NORMAL);
```

Nota:

1. Salvo el valor para la descripción, todos los valores de atributo que se muestran son valores predeterminados. No obstante, se muestran aquí con fines ilustrativos. Puede omitirlos si está seguro de que los valores predeterminados son los que desea o no han sido modificados. Consulte también el apartado “Visualización de los atributos de objeto predeterminados” en la página 86.
2. USAGE (NORMAL) indica que esta cola no es una cola de transmisión.
3. Si ya hay una cola local llamada ORANGE.LOCAL.QUEUE en el mismo gestor de colas, este mandato no podrá ejecutarse. Utilice el atributo REPLACE si desea escribir encima de la definición ya existente de una cola, pero vea también el apartado “Modificación de atributos de cola local” en la página 87.

Definición de una cola de mensajes no entregados

Cada gestor de colas debe tener una cola local que se utilizará como cola de mensajes no entregados, con el fin de que los mensajes que no se puedan entregar en su destino correcto se almacenen para recuperación posterior. La cola de mensajes no entregados debe indicarse al gestor de colas.

Para informar al gestor de colas sobre la cola de mensajes no entregados, especifique un nombre de cola de mensajes no entregados en el mandato `crtmqm` (`crtmqm -u DEAD.LETTER.QUEUE`, por ejemplo), o

utilizando el atributo DEADQ en el mandato ALTER QMGR para especificar uno más adelante. Debe definir la cola de mensajes no entregados antes de poder utilizarla.

Con el producto se entrega una cola de mensajes no entregados de ejemplo llamada SYSTEM.DEAD.LETTER.QUEUE. Esta cola se crea automáticamente cuando se crea el gestor de colas. Si es necesario, puede modificar esta definición y cambiar su nombre.

Una cola de mensajes no entregados no tiene ningún requisito especial excepto que:

- Debe ser una cola local
- Su atributo MAXMSGL (longitud máxima de mensajes) debe permitir que la cola pueda alojar los mensajes más grandes que el gestor de colas tenga que manejar **más** el tamaño de la cabecera de mensaje no entregado (MQDLH)

WebSphere MQ proporciona un manejador de cola de mensajes no entregados que permite especificar cómo se van a procesar o eliminar los mensajes de una cola de mensajes no entregados. Para obtener más información, consulte Manejo de mensajes no entregados con el manejador de cola de mensajes no entregados WebSphere MQ .

Visualización de los atributos de objeto predeterminados

Puede utilizar el mandato DISPLAY QUEUE para visualizar los atributos que se tomaron del objeto predeterminado al definir un objeto de WebSphere MQ.

Cuando define un objeto WebSphere MQ, éste toma del objeto predeterminado todos los atributos que no especifique. Por ejemplo, cuando define una cola local, la cola hereda todos los atributos que omita en la definición de la cola local predeterminada, que tiene el nombre SYSTEM.DEFAULT.LOCAL.QUEUE. Si desea saber cuáles son exactamente esos atributos, utilice este mandato:

```
DISPLAY QUEUE (SYSTEM.DEFAULT.LOCAL.QUEUE)
```

La sintaxis de este mandato es distinta de la del mandato DEFINE correspondiente. En el mandato DISPLAY puede proporcionar simplemente el nombre de la cola, mientras que en el mandato DEFINE tiene que especificar el tipo de cola, es decir, QLOCAL, QALIAS, QMODEL o QREMOTE.

Puede visualizar de forma selectiva los atributos especificándolos individualmente. Por ejemplo:

```
DISPLAY QUEUE (ORANGE.LOCAL.QUEUE) +
        MAXDEPTH +
        MAXMSGL +
        CURDEPTH;
```

Este mandato muestra los tres atributos especificados de la siguiente manera:

```
AMQ8409: Display Queue details.
   QUEUE(ORANGE.LOCAL.QUEUE)               TYPE(QLOCAL)
   CURDEPTH(0)                             MAXDEPTH(5000)
   MAXMSGL(4194304)
```

CURDEPTH es la profundidad de cola actual, es decir, el número de mensajes que hay en la cola. Visualizar este atributo resulta de gran utilidad, ya que puede utilizarlo para supervisar la profundidad de cola y asegurarse de que no se llena.

Copiar una definición de cola local

Puede copiar una definición de cola utilizando el atributo LIKE en el mandato DEFINE.

Por ejemplo:

```
DEFINE QLOCAL (MAGENTA.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE)
```

Este mandato crea una cola que tiene los mismos atributos que la cola ORANGE.LOCAL.QUEUE original de nuestro ejemplo, en vez de tener los atributos de la cola local predeterminada del sistema. Escriba el nombre de la cola que se va a copiar ***exactamente*** como lo ha entrado al crear la cola. Si el nombre contiene caracteres en minúsculas, encierre el nombre entre comillas simples.

También puede utilizar esta forma del mandato DEFINE para copiar una definición de cola y sustituir una o más modificaciones en los atributos de la original. Por ejemplo:

```
DEFINE QLOCAL (THIRD.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE) +
       MAXMSGL(1024);
```

Este mandato copia los atributos de la cola ORANGE.LOCAL.QUEUE en la cola THIRD.QUEUE, pero especifica que la longitud máxima de mensaje en la nueva cola será 1024 bytes en lugar de 4194304.

Nota:

1. Cuando se utiliza el atributo LIKE en un mandato DEFINE, sólo se copian los atributos de la cola. No se copian los mensajes existente en la cola.
2. Si define una cola local sin especificar LIKE, es lo mismo que DEFINE LIKE(SYSTEM.DEFAULT.LOCAL.QUEUE).

Modificación de atributos de cola local

Los atributos de cola se pueden modificar de dos maneras, utilizando el mandato ALTER QLOCAL o utilizando el mandato DEFINE QLOCAL con el atributo REPLACE.

En “Definición de una cola local” en la página 85, se ha definido la cola llamada ORANGE.LOCAL.QUEUE. Supongamos, por ejemplo, que desea reducir la longitud máxima de los mensajes en esta cola a 10.000 bytes.

- Utilice el mandato ALTER:

  ```
  ALTER QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000)
  ```

  Este mandato modifica un solo atributo, el de la longitud máxima del mensaje; todos los demás atributos no cambian.
- Utilizando el mandato DEFINE con la opción REPLACE, por ejemplo:

  ```
  DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000) REPLACE
  ```

  Este mandato no sólo modifica la longitud máxima del mensaje, sino también todos los demás atributos, a los que se les asignan sus valores predeterminados. La cola está ahora habilitada para operaciones de transferencia mientras que anteriormente estaba inhibida para dichas operaciones. El valor predeterminado es habilitada para operaciones de transferencia, según lo especificado por la cola SYSTEM.DEFAULT.LOCAL.QUEUE.

  Si ***reduce*** la longitud máxima del mensaje en una cola existente, los mensajes existentes no se ven afectados. Sin embargo, todos los mensajes nuevos deben cumplir los nuevos criterios.

Vaciar una cola local

Puede utilizar el mandato CLEAR para vaciar una cola local.

Para suprimir todos los mensajes de una cola local, llamada MAGENTA.QUEUE, utilice el siguiente mandato:

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

Nota: No hay ningún mensaje de solicitud que le permita cambiar de opinión, por lo tanto, cuando pulse la tecla Intro se perderán los mensajes.

Una cola no se puede vaciar si:

- Hay mensajes no confirmados que se han transferido a la cola bajo punto de sincronización.
- Hay una aplicación que tiene abierta actualmente la cola.

Suprimir una cola local

Puede utilizar el mandato MQSC DELETE QLOCAL para suprimir una cola local.

Una cola no puede suprimirse si contiene mensajes sin confirmar. No obstante, si la cola tiene uno o más mensajes confirmados y ningún mensaje sin confirmar, se puede suprimir sólo si se especifica la opción PURGE. Por ejemplo:

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

Especificar NOPURGE en lugar de PURGE asegura que la cola no se suprime si contiene algún mensaje confirmado.

Examinar colas

WebSphere MQ proporciona un examinado de colas de ejemplo que puede utilizar para examinar el contenido de los mensajes de una cola. El examinador se suministra en los formatos fuente y ejecutable.

MQ_INSTALLATION_PATH representa el directorio de alto nivel en el que está instalado WebSphere MQ.

En WebSphere MQ para Windows, los nombres de archivo y las vías de acceso del navegador de colas de ejemplo son los siguientes:

Fuente
MQ_INSTALLATION_PATH\tools\c\samples\

Ejecutable
MQ_INSTALLATION_PATH\tools\c\samples\bin\amqsbcg.exe

En WebSphere MQ para UNIX and Linux, los nombres de archivo y las vías de acceso son los siguientes:

Fuente
MQ_INSTALLATION_PATH/samp/amqsbcg0.c

Ejecutable
MQ_INSTALLATION_PATH/samp/bin/amqsbcg

El ejemplo requiere dos parámetros de entrada, el nombre de la cola y el nombre del gestor de colas. Por ejemplo:

```
amqsbcg SYSTEM.ADMIN.QMGREVENT.tpp01 saturn.queue.manager
```

Los resultados habituales de este mandato pueden verse en la Figura 16 en la página 89.

```
AMQSBCG0 - starts here
**********************

 MQOPEN - 'SYSTEM.ADMIN.QMGR.EVENT'


 MQGET of message number 1
****Message descriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 850
  Format : 'MQEVENT '
  Priority : 0  Persistence : 0
  MsgId : X'414D512073617475726E2E71756575650005D30033563DB8'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'saturn.queue.manager                          '
  ** Identity Context
  UserIdentifier : '            '
  AccountingToken :
   X'0000000000000000000000000000000000000000000000000000000000000000'
  ApplIdentityData : '                                        '
  ** Origin Context
  PutApplType    : '7'
  PutApplName    : 'saturn.queue.manager        '
  PutDate  : '19970417'    PutTime  : '15115208'
  ApplOriginData : '    '

  GroupId : X'000000000000000000000000000000000000000000000000'
  MsgSeqNumber   : '1'
  Offset         : '0'
  MsgFlags       : '0'
  OriginalLength : '104'

****   Message      ****

 length - 104 bytes

00000000:  0700 0000 2400 0000 0100 0000 2C00 0000 '....→.......,...'
00000010:  0100 0000 0100 0000 0100 0000 AE08 0000 '................'
00000020:  0100 0000 0400 0000 4400 0000 DF07 0000 '........D.......'
00000030:  0000 0000 3000 0000 7361 7475 726E 2E71 '....0...saturn.q'
00000040:  7565 7565 2E6D 616E 6167 6572 2020 2020 'ueue.manager    '
00000050:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060:  2020 2020 2020 2020                     '                '

 No more messages
 MQCLOSE
 MQDISC
```

Figura 16. Resultados habituales del examinador de colas

Habilitar colas grandes

IBM WebSphere MQ da soporte a colas de más de 2 GB.

En sistemas Windows, el soporte para archivos grandes está disponible sin ninguna habilitación adicional. En sistemas AIX, HP-UX, Linux y Solaris, deberá habilitar de forma explícita el soporte de archivos grandes para poder crear archivos de colas que tengan más de 2 GB. Consulte la documentación del sistema operativo para obtener información sobre cómo llevar a cabo esta operación.

Algunos programas de utilidad, por ejemplo tar, no pueden trabajar con archivos de más de 2 GB. Antes de habilitar el soporte de archivos grandes, consulte la documentación del sistema operativo para ver información sobre las restricciones en los programas de utilidad que utilice.

Para obtener información sobre la planificación de la cantidad de almacenamiento necesaria para las colas, visite el sitio web IBM WebSphere MQ para ver informes de rendimiento específicos de cada plataforma:

```
https://www.ibm.com/software/integration/ts/mqseries/
```

Trabajar con colas alias

Puede definir una cola alias para hacer referencia indirectamente a otra cola o tema.

V 7.5.0.8

Atención: Las listas de distribución no admiten el uso de cola alias que apuntan a objetos de tema. A partir de Version 7.5.0, Fix Pack 8, si una cola alias apunta a un objeto de tema en una lista de distribución, IBM WebSphere MQ devuelve MQRC_ALIAS_BASE_Q_TYPE_ERROR.

La cola a la que una cola alias hace referencia puede ser cualquiera de las siguientes:

- Una cola local (consulte el apartado “Definición de una cola local” en la página 85).
- Una definición local de una cola remota (consulte el apartado “Creación de una definición local de una cola remota” en la página 114).
- Un tema.

Una cola alias no es una cola real, sino una definición que se resuelve en una cola real (o de destino) en tiempo de ejecución. La definición de la cola alias especifica la cola de destino. Cuando una aplicación efectúa una llamada MQOPEN a una cola alias, el gestor de colas resuelve el alias en el nombre de la cola de destino.

Una cola alias no se puede resolver en otra cola alias definida localmente. Sin embargo, una cola alias puede resolverse como colas alias definidas en otro lugar de los clústeres de los cuales forme parte el gestor de colas local. En Resolución de nombres hallará más información.

Las colas alias son útiles para:

- Proporcionar a las aplicaciones niveles diferentes de autorizaciones de acceso a la cola de destino.
- Permitir que diferentes aplicaciones trabajen con la misma cola de diferentes modos. (Quizá desee asignar prioridades predeterminadas diferentes o valores de persistencia predeterminados diferentes).
- Simplificar el mantenimiento, la migración y el equilibrio de carga. (Quizá desee cambiar el nombre de la cola de destino sin tener que cambiar la aplicación, que sigue utilizando el alias).

Por ejemplo, suponga que se ha desarrollado una aplicación para colocar mensajes en una cola denominada MY.ALIAS.QUEUE. La aplicación especifica el nombre de esta cola cuando realiza una solicitud MQOPEN e, indirectamente, si coloca un mensaje en esta cola. La aplicación no sabe que la cola es una cola alias. Para cada llamada MQI que utilice este alias, el gestor de colas determina el nombre de cola real, que podría ser una cola local o una cola remota definida en este gestor de colas.

Al cambiar el valor del atributo TARGET, puede redirigir las llamadas MQI a otra cola, posiblemente en otro gestor de colas. Esto resulta de gran utilidad para el mantenimiento, la migración y el equilibrio de la carga.

Definición de una cola alias

El mandato siguiente crea una cola alias:

```
DEFINE QALIAS (MY.ALIAS.QUEUE) TARGET (YELLOW.QUEUE)
```

Este mandato redirige las llamadas MQI que especifican MY.ALIAS.QUEUE a la cola YELLOW.QUEUE. Este mandato no crea la cola de destino; las llamadas MQI fracasan si la cola YELLOW.QUEUE no existe en tiempo de ejecución.

Si cambia la definición del alias, puede redirigir las llamadas MQI a otra cola. Por ejemplo:

```
ALTER QALIAS (MY.ALIAS.QUEUE) TARGET (MAGENTA.QUEUE)
```

Este mandato redirige las llamadas MQI a otra cola, MAGENTA.QUEUE.

También puede utilizar colas alias para que hacer que una sola cola (la cola destino) parezca tener atributos distintos para aplicaciones distintas. Esto se hace definiendo dos alias, uno para cada aplicación. Suponga que tiene dos aplicaciones:

- La aplicación ALPHA puede transferir mensajes a YELLOW.QUEUE, pero no tiene autorización para obtener mensajes de dicha cola.
- La aplicación BETA puede obtener mensajes de YELLOW.QUEUE, pero no tiene autorización para colocar mensajes en ella.

El mandato siguiente define un alias que está habilitado para operaciones de transferencia e inhabilitado para operaciones de obtención para la aplicación ALPHA:

```
DEFINE QALIAS (ALPHAS.ALIAS.QUEUE) +
       TARGET (YELLOW.QUEUE) +
       PUT (ENABLED) +
       GET (DISABLED)
```

El mandato siguiente define un alias que está inhabilitado para operaciones de transferencia y habilitado para operaciones de obtención para la aplicación BETA:

```
DEFINE QALIAS (BETAS.ALIAS.QUEUE) +
       TARGET (YELLOW.QUEUE) +
       PUT (DISABLED) +
       GET (ENABLED)
```

ALPHA utiliza el nombre de cola ALPHAS.ALIAS.QUEUE en sus llamadas MQI; BETA utiliza el nombre de cola BETAS.ALIAS.QUEUE. Ambas aplicaciones acceden a la misma cola, pero de forma diferente.

Puede utilizar los atributos LIKE y REPLACE cuando defina alias de colas, del mismo modo que los utiliza con colas locales.

Utilización de otros mandatos con colas alias

Puede utilizar los mandatos MQSC adecuados para visualizar o alterar atributos de la cola alias o para suprimir el objeto de cola alias. Por ejemplo:

Utilice el mandato siguiente para visualizar los atributos de la cola alias:

```
DISPLAY QALIAS (ALPHAS.ALIAS.QUEUE)
```

Utilice el mandato siguiente para alterar el nombre de cola base, en el que se resuelve el alias, donde la opción `force` fuerza el cambio incluso si la cola está abierta:

```
ALTER QALIAS (ALPHAS.ALIAS.QUEUE) TARGET(ORANGE.LOCAL.QUEUE) FORCE
```

Utilice el mandato siguiente para suprimir este alias de cola:

```
DELETE QALIAS (ALPHAS.ALIAS.QUEUE)
```

No se puede suprimir una cola alias si una aplicación tiene actualmente abierta la cola. Consulte Referencia de MQSC para obtener más información sobre esto y otros mandatos de cola alias.

Trabajar con colas modelo

Un gestor de colas crea una *cola dinámica* si recibe una llamada MQI de una aplicación que especifica un nombre de cola que se ha definido como cola modelo. El nombre de la nueva cola dinámica lo genera el gestor de colas cuando se crea la cola. Una *cola modelo* es una plantilla que especifica los atributos de las colas dinámicas creadas a partir de ella. Las colas modelo proporcionan un método práctico para que las aplicaciones puedan crear colas cuando se necesitan.

Definición de una cola modelo

Las colas modelo se definen con un conjunto de atributos del mismo modo en que se define una cola local. Las colas modelo y las colas locales tienen el mismo conjunto de atributos, excepto que en las colas modelo se puede especificar si las colas dinámicas creadas son temporales o persistentes. (Las colas persistentes se conservan tras reiniciar el gestor de colas, las colas temporales no). Por ejemplo:

```
DEFINE QMODEL (GREEN.MODEL.QUEUE) +
       DESCR('Queue for messages from application X') +
       PUT (DISABLED) +
       GET (ENABLED) +
       NOTRIGGER +
       MSGDLVSQ (FIFO) +
       MAXDEPTH (1000) +
       MAXMSGL (2000) +
       USAGE (NORMAL) +
       DEFTYPE (PERMDYN)
```

Este mandato crea una definición de cola modelo. Desde el atributo DEFTYPE, puede ver que las colas reales creadas con esta plantilla son colas dinámicas permanentes. Todos los atributos no especificados se copian automáticamente de la cola predeterminada SYSTEM.DEFAULT.MODEL.QUEUE.

Puede utilizar los atributos LIKE y REPLACE cuando defina colas modelo, del mismo modo que los utiliza con colas locales.

Utilización de otros mandatos con colas modelo

Puede utilizar los mandatos MQSC adecuados para visualizar o alterar los atributos de una cola modelo o para suprimir el objeto de cola modelo. Por ejemplo:

Utilice el mandato siguiente para visualizar los atributos de la cola modelo:

```
DISPLAY QUEUE (GREEN.MODEL.QUEUE)
```

Utilice el mandato siguiente para alterar el modelo a fin de habilitar operaciones de transferencia en cualquier cola dinámica creada a partir de este modelo:

```
ALTER QMODEL (BLUE.MODEL.QUEUE) PUT(ENABLED)
```

Utilice el mandato siguiente para suprimir esta cola modelo:

```
DELETE QMODEL (RED.MODEL.QUEUE)
```

Trabajar con temas administrativos

Utilice los mandatos MQSC para gestionar temas administrativos.

Consulte Referencia de MQSC si desea información detallada sobre estos mandatos.

Conceptos relacionados

Objetos de tema administrativo

“Definición de un tema administrativo” en la página 93

Utilice el mandato MQSC **DEFINE TOPIC** para crear un tema administrativo. Al definir un tema administrativo, puede establecer cada atributo de tema.

"Visualizar atributos de objeto de tema administrativo" en la página 93
Utilice el mandato MQSC **DISPLAY TOPIC** para visualizar un objeto de tema administrativo.

"Cambiar atributos de tema administrativo" en la página 94
Puede cambiar los atributos de tema de dos maneras, utilizando el mandato **ALTER TOPIC** o el mandato **DEFINE TOPIC** con el atributo **REPLACE** .

"Copiar una definición de tema administrativo" en la página 94
Puede copiar una definición de tema utilizando el atributo LIKE en el mandato **DEFINE**.

"Supresión de una definición de tema administrativo" en la página 95
Puede utilizar el mandato MQSC **DELETE TOPIC** para suprimir un tema administrativo.

Definición de un tema administrativo

Utilice el mandato MQSC **DEFINE TOPIC** para crear un tema administrativo. Al definir un tema administrativo, puede establecer cada atributo de tema.

Los atributos de tema que no se definen explícitamente se heredan del tema administrativo predeterminado, SYSTEM.DEFAULT.TOPIC, creado al realizar la instalación del sistema.

Por ejemplo, el mandato **DEFINE TOPIC** que sigue, define un tema denominado **ORANGE.TOPIC** con estas características:

- Se resuelve en la serie de tema ORANGE. Para obtener información sobre cómo se pueden utilizar las series de temas, consulte Combinación de series de temas.
- Los atributos establecidos en ASPARENT utilizan el atributo tal como está definido en el tema padre de este tema. Esta acción se repite ene árbol de temas hasta que se encuentra el tema raíz SYSTEM.BASE.TOPIC. Si desea más información sobre los árboles de temas, consulte Árboles de temas.

```
DEFINE TOPIC (ORANGE.TOPIC) +
       TOPICSTR (ORANGE) +
       DEFPRTY(ASPARENT) +
       NPMSGDLV(ASPARENT)
```

Nota:

- Con la excepción del valor de la serie de tema, todos los valores de atributo que se muestran son los valores predeterminados. Se muestran aquí sólo como ejemplo ilustrativo.Puede omitirlos si está seguro de que los valores predeterminados son los que desea o no han sido modificados. Consulte también el apartado "Visualizar atributos de objeto de tema administrativo" en la página 93.
- Si ya tiene un tema administrativo con el nombre ORANGE.TOPIC en el mismo gestor de colas, este mandato falla.Utilice el atributo REPLACE si desea sobrescribir la definición existente de un tema, pero consulte también "Cambiar atributos de tema administrativo" en la página 94

Visualizar atributos de objeto de tema administrativo

Utilice el mandato MQSC **DISPLAY TOPIC** para visualizar un objeto de tema administrativo.

Para visualizar todos los temas, utilice:

```
DISPLAY TOPIC(ORANGE.TOPIC)
```

Puede visualizar de forma selectiva los atributos especificándolos individualmente.Por ejemplo:

```
DISPLAY TOPIC(ORANGE.TOPIC) +
        TOPICSTR +
        DEFPRTY +
        NPMSGDLV
```

Este mandato muestra los tres atributos especificados de la siguiente manera:

```
AMQ8633: Display topic details.
   TOPIC(ORANGE.TOPIC)                     TYPE(LOCAL)
   TOPICSTR(ORANGE)                        DEFPRTY(ASPARENT)
   NPMSGDLV(ASPARENT)
```

Para visualizar los valores ASPARENT del tema tal como se utilizan en tiempo de ejecución, utilice DISPLAY TPSTATUS. Por ejemplo, utilice:

```
DISPLAY TPSTATUS(ORANGE) DEFPRTY NPMSGDLV
```

El mandato muestra los siguientes detalles:

```
AMQ8754: Display topic status details.
   TOPICSTR(ORANGE)                        DEFPRTY(0)
   NPMSGDLV(ALLAVAIL)
```

Al definir un tema administrativo, éste toma todos los atributos que no especifique explícitamente del tema administrativo predeterminado, que se llama SYSTEM.DEFAULT.TOPIC. Para saber cuáles son estos atributos predeterminados, utilice el siguiente mandato:

```
DISPLAY TOPIC (SYSTEM.DEFAULT.TOPIC)
```

Cambiar atributos de tema administrativo

Puede cambiar los atributos de tema de dos maneras, utilizando el mandato **ALTER TOPIC** o el mandato **DEFINE TOPIC** con el atributo **REPLACE** .

Si, por ejemplo, desea cambiar la prioridad predeterminada de los mensajes entregados a un tema denominado ORANGE.TOPIC, para que sea 5, utilice uno de estos mandatos:

- Utilizando el mandato **ALTER**:

  ```
  ALTER TOPIC(ORANGE.TOPIC) DEFPRTY(5)
  ```

  Este mandato cambia un único atributo, el de la prioridad predeterminada del mensaje entregado a este tema a 5; todos los demás atributos permanecen intactos.

- Utilizando el mandato **DEFINE**:

  ```
  DEFINE TOPIC(ORANGE.TOPIC) DEFPRTY(5) REPLACE
  ```

  Este mandato cambia la prioridad predeterminada de los mensajes entregados a este tema. Todos los demás atributos reciben sus valores predeterminados.

Si modifica la prioridad de los mensajes enviados a este tema, los mensajes existentes no se ven afectados. Los mensajes nuevos, sin embargo, utilizan la prioridad especificada si no la proporciona la aplicación de publicación.

Copiar una definición de tema administrativo

Puede copiar una definición de tema utilizando el atributo LIKE en el mandato **DEFINE**.

Por ejemplo:

```
DEFINE TOPIC (MAGENTA.TOPIC) +
       LIKE (ORANGE.TOPIC)
```

Este mandato crea un tema, MAGENTA.TOPIC, con los mismos atributos que el tema original, ORANGE.TOPIC, en lugar de los atributos del tema administrativo predeterminado del sistema. Especifique el nombre del tema que debe copiarse exactamente cómo lo especificó al crear el tema. Si el nombre contiene caracteres en minúscula, especifique el nombre entre comillas simples.

También puede utilizar esta forma del mandato **DEFINE** para copiar una definición de tema, pero realice los cambios en losa tributos del original. Por ejemplo:

```
DEFINE TOPIC(BLUE.TOPIC) +
       TOPICSTR(BLUE) +
       LIKE(ORANGE.TOPIC)
```

También puede copiar los atributos del tema BLUE.TOPIC en el tema GREEN.TOPIC y especificar que cuando las aplicaciones no puedan entregarse a la cola de suscriptor correcta no se coloquen en la cola de mensajes no entregados. Por ejemplo:

```
DEFINE TOPIC(GREEN.TOPIC) +
       TOPICSTR(GREEN) +
       LIKE(BLUE.TOPIC) +
       USEDLQ(NO)
```

Supresión de una definición de tema administrativo

Puede utilizar el mandato MQSC **DELETE TOPIC** para suprimir un tema administrativo.

```
DELETE TOPIC(ORANGE.TOPIC)
```

Las aplicaciones ya no podrán abrir el tema para su publicación o realizar nuevas suscripciones utilizando el nombre del objeto, ORANGE.TOPIC.Las aplicaciones de publicación que tienen el tema abierto pueden continuar publicando la serie de tema resuelta.Las suscripciones ya realizadas en este tema continúan recibiendo publicaciones después de que se haya suprimido el tema.

Las aplicaciones que no hacen referencia a este objeto de tema, pero que utilizan la serie de tema resuelta representada por este objeto de tema, 'NARANJA' en este ejemplo, siguen funcionando. En este caso heredan las propiedades de un objeto de tema superior en el árbol de temas. Si desea más información sobre los árboles de temas, consulte Árboles de temas.

Trabajar con suscripciones

Utilice los mandatos MQSC para gestionar suscripciones.

Las suscripciones pueden ser de tres tipos, definidos en el atributo **SUBTYPE**:

ADMIN
Definida administrativamente por un usuario.

PROXY
Suscripción creada internamente para direccionar publicaciones entre gestores de colas.

API
Creada mediante programación, por ejemplo, con la llamada MQI MQSUB.

En el manual Referencia de MQSC encontrará información detallada acerca de estos mandatos.

Conceptos relacionados

“Definir una suscripción administrativa” en la página 96
Utilice el mandato MQSC **DEFINE SUB** para crear una suscripción administrativa. También puede utilizar el valor predeterminado definido en la definición de suscripción local predeterminada. O bien puede modificar las características de suscripción a partir de las de la suscripción local predeterminada SYSTEM.DEFAULT.SUB creada al instalar el sistema.

“Visualización de atributos de suscripciones” en la página 96
Puede utilizar el mandato **DISPLAY SUB** para visualizar los atributos configurados de cualquier suscripción conocida por el gestor de colas.

“Cambiar atributos de suscripciones locales” en la página 97
Puede cambiar los atributos de suscripción de dos formas, mediante el mandato **ALTER SUB** o el mandato **DEFINE SUB** con el atributo **REPLACE**.

“Copiar un definición de suscripción local” en la página 98

Puede copiar una definición de suscripción utilizando el atributo **LIKE** en el mandato **DEFINE**.

"Supresión de una suscripción" en la página 98
Puede utilizar el mandato MQSC **DELETE SUB** para suprimir una suscripción local.

Definir una suscripción administrativa

Utilice el mandato MQSC **DEFINE SUB** para crear una suscripción administrativa. También puede utilizar el valor predeterminado definido en la definición de suscripción local predeterminada. O bien puede modificar las características de suscripción a partir de las de la suscripción local predeterminada SYSTEM.DEFAULT.SUB creada al instalar el sistema.

Por ejemplo, el mandato **DEFINE SUB** que sigue define una suscripción denominada ORANGE con estas características:

- Suscripción duradera, lo que significa que persiste tras el reinicio del gestor de colas, con una caducidad ilimitada.
- Recibe las publicaciones creadas en la serie de tema ORANGE, con las prioridades de mensaje tal como las establecen las aplicaciones de publicación.
- Las publicaciones entregadas para esta suscripción se envían a la cola local SUBQ, que debe definirse antes de definir la suscripción.

```
DEFINE SUB (ORANGE) +
       TOPICSTR (ORANGE) +
       DESTCLAS (PROVIDED) +
       DEST (SUBQ) +
       EXPIRY (UNLIMITED) +
       PUBPRTY (ASPUB)
```

Nota:

- La suscripción y el nombre de la serie de tema no es tienen que coincidir.
- Salvo los valores de la descripción y la serie de tema, todos los valores de atributo que se muestran son los valores predeterminados.Se muestran aquí sólo como ejemplo ilustrativo.Puede omitirlos si está seguro de que los valores predeterminados son los que desea o no han sido modificados. Consulte también el apartado "Visualización de atributos de suscripciones" en la página 96.
- Si ya tiene una suscripción local con el nombre TEST en el mismo gestor de colas, este mandato falla.Utilice el atributo **REPLACE** si desea sobrescribir la definición existente de una cola, pero consulte también "Cambiar atributos de suscripciones locales" en la página 97.
- Si la cola SUBQ no existe, este mandato falla.

Visualización de atributos de suscripciones

Puede utilizar el mandato **DISPLAY SUB** para visualizar los atributos configurados de cualquier suscripción conocida por el gestor de colas.

Por ejemplo, utilice:

```
DISPLAY SUB (ORANGE)
```

Puede visualizar de forma selectiva los atributos especificándolos individualmente. Por ejemplo:

```
DISPLAY SUB (ORANGE) +
        SUBID +
        TOPICSTR +
        DURABLE
```

Este mandato muestra los tres atributos especificados de la siguiente manera:

```
AMQ8096: WebSphere MQ subscription inquired.
   SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   SUB(ORANGE)                                TOPICSTR(ORANGE)
   DURABLE(YES)
```

TOPICSTR es la serie de tema resuelta en la que opera este suscriptor.Cuando se define una suscripción para utilizar un objeto de tema, la serie de tema de dicho objeto se utiliza como prefijo para la serie de tema proporcionada al realizar la suscripción. SUBID es un identificador exclusivo asignado por el gestor de colas cuando se crea una suscripción.Se trata de un atributo útil que se debe visualizar para mostrar porque algunos nombres de suscripciones pueden ser largos o estar en un juego de caracteres diferentes para los que podría resultar poco práctico.

Un método alternativo para visualizar las suscripciones es utilizar el SUBID:

```
DISPLAY SUB +
        SUBID(414D5120414141202020202020202020EE921E4E20002A03) +
        TOPICSTR +
        DURABLE
```

Este mandato ofrece la misma salida que antes:

```
AMQ8096: WebSphere MQ subscription inquired.
 SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   SUB(ORANGE)                                TOPICSTR(ORANGE)
   DURABLE(YES)
```

Las suscripciones de proxy en un gestor de colas no se visualizan de forma predeterminada. Para visualizarlas, especifique el valor PROXY o ALL para **SUBTYPE**.

Puede utilizar el mandato DISPLAY SBSTATUS para ver los atributos de tiempo de ejecución. Por ejemplo, utilice el mandato:

```
DISPLAY SBSTATUS(ORANGE) NUMMSGS
```

Se visualiza la siguiente salida:

```
AMQ8099: WebSphere MQ subscription status inquired.
   SUB(ORANGE)
   SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   NUMMSGS(0)
```

Al definir una suscripción administrativa, ésta toma los atributos que no se especifican explícitamente de la suscripción predeterminada, que se denomina SYSTEM.DEFAULT.SUB. Para saber cuáles son estos atributos predeterminados, utilice el siguiente mandato:

```
DISPLAY SUB (SYSTEM.DEFAULT.SUB)
```

Cambiar atributos de suscripciones locales

Puede cambiar los atributos de suscripción de dos formas, mediante el mandato **ALTER SUB** o el mandato **DEFINE SUB** con el atributo **REPLACE**.

Si, por ejemplo, desea cambiar a 5 la prioridad de los mensajes entregados a una suscripción denominada ORANGE, utilice uno de estos mandatos:

- Utilice el mandato ALTER:

  ```
  ALTER SUB(ORANGE) PUBPRTY(5)
  ```

  Este mandato cambia un solo atributo, el de la prioridad de los mensajes entregados a esta suscripción a 5; todos los demás atributos permanecen intactos.
- Utilice el mandato DEFINE:

  ```
  DEFINE SUB (ORANGE) PUBPRTY(5) REPLACE
  ```

  Este mandato cambia la prioridad de los mensajes entregados a esta suscripción y todos los demás atributos que tienen a los que se asignan los valores predeterminados.

Si modifica la prioridad de los mensajes enviados a esta suscripción, los mensajes existentes no se verán afectados.Sin embargo, los nuevos mensajes tienen la prioridad especificada.

Copiar un definición de suscripción local

Puede copiar una definición de suscripción utilizando el atributo **LIKE** en el mandato **DEFINE**.

Por ejemplo:

```
DEFINE SUB (BLUE) +
       LIKE (ORANGE)
```

También puede copiar los atributos de la suscripción REAL a la suscripción THIRD.SUB, y especificar el ID de correlación de las publicaciones entregadas como THIRD, en lugar de el ID de correlación de los publicadores. Por ejemplo:

```
DEFINE SUB(THIRD.SUB) +
       LIKE(BLUE) +
       DESTCORL(ORANGE)
```

Supresión de una suscripción

Puede utilizar el mandato MQSC **DELETE SUB** para suprimir una suscripción local.

```
DELETE SUB(ORANGE)
```

También puede suprimir una suscripción utilizando el SUBID:

```
DELETE SUB SUBID(414D5120414141202020202020202020EE921E4E20002A03)
```

Comprobación de mensajes en una suscripción

Acerca de esta tarea

Cuando se define una suscripción, ésta se asocia a una cola. Los mensajes publicados que coinciden con la suscripción se colocan en esta cola.

Tenga en cuenta que los mandatos **runmqsc** siguientes muestran sólo las suscripciones que han recibido mensajes.

Para comprobar los mensajes que actualmente se encuentran en la cola para una suscripción, siga estos pasos:

Procedimiento

1. Para comprobar si hay mensajes en cola para un tipo de suscripción **DISPLAY SBSTATUS(<sub_name>) NUMMSGS**, consulte “Visualización de atributos de suscripciones” en la página 96.
2. Si el valor de **NUMMSGS** es mayor que cero, identifique la cola asociada con la suscripción escribiendo **DISPLAY SUB(<sub_name>)DEST**.
3. Utilizando el nombre de la cola devuelta puede ver los mensajes siguiendo la técnica que se describe en “Examinar colas” en la página 88.

Trabajar con servicios

Los objetos de servicio son una forma de gestionar los procesos adicionales como parte de un gestor de colas. Con los servicios, puede definir programas que se inician y se detienen cuando se inicia o se detiene el gestor de colas. Los servicios de IBM WebSphere MQ siempre se inician bajo el ID de usuario del usuario que ha iniciado el gestor de colas.

Los objetos de servicio pueden ser de los tipos siguientes:

Servidor
Un servidor es un objeto de servicio que tiene el parámetro SERVTYPE establecido en SERVER. Un objeto de servicio de servidor es la definición de un programa que se ejecuta cuando se inicia un gestor de colas especificado. Los objetos de servicio de servidor definen programas que normalmente se ejecutan durante un largo periodo de tiempo. Por ejemplo, un objeto de servicio de servidor se puede utilizar para ejecutar un proceso de supervisor desencadenante, como `runmqtrm`.

Sólo se puede ejecutar una instancia de un objeto de servicio de servidor al mismo tiempo. El estado de los objetos de servicio de servidor que están en ejecución se puede supervisar con el mandato MQSC, DISPLAY SVSTATUS.

Mandato
Un mandato es un objeto de servicio que tiene el parámetro SERVTYPE establecido en COMMAND. Los objetos de servicio de mandato son similares a los objetos de servicio de servidor, aunque se pueden ejecutar varias instancias de un objeto de servicio de mandato simultáneamente y su estado no se puede supervisar con el mandato MQSC DISPLAY SVSTATUS.

Si se ejecuta el mandato MQSC, STOP SERVICE, no se realiza ninguna comprobación para determinar si el programa que inició el mandato MQSC, START SERVICE, sigue activo antes de ejecutar el programa de detención.

Definición de un objeto de servicio

Puede definir un objeto de servicio con diversos atributos.

Los atributos son los siguientes:

SERVTYPE
Define el tipo del objeto de servicio. Los valores posibles son los siguientes:

Servidor
Objeto de servicio del servidor.

Sólo se puede ejecutar una instancia de un objeto de servicio de servidor al mismo tiempo. El estado de los objetos de servicio de servidor se puede supervisar con el mandato MQSC, DISPLAY SVSTATUS.

Mandato
Objeto de servicio del mandato.

Se pueden ejecutar varias instancias de un objeto de servicio de mandato al mismo tiempo. El estado de un objeto de servicio de mandato no se pueden supervisar.

STARTCMD
El programa que se ejecuta para iniciar el servicio. Debe especificarse una vía de acceso completa al programa.

STARTARG
Argumentos que se han pasado al programa de inicio.

STDERR
Especifica la vía de acceso a un archivo al que se debería redirigir la salida de error estándar (stderr) del programa de servicio.

STDOUT
Especifica la vía de acceso a un archivo al que se debería redirigir la salida estándar (stdout) del programa de servicio.

STOPCMD
El programa que se ejecuta para detener el servicio. Debe especificarse una vía de acceso completa al programa.

STOPARG
Argumentos que se han pasado al programa de detención.

CONTROL
Especifica cómo se debe iniciar y detener el servicio.

MANUAL
El servicio no se debe iniciar ni detener de forma automática. está controlado por la utilización de los mandatos START SERVICE y STOP SERVICE. Este es el valor predeterminado.

QMGR
El servicio que se define se debe iniciar y detener al mismo tiempo que se inicia y se detiene el gestor de colas.

STARTONLY
El servicio debe iniciarse al mismo tiempo que se inicia el gestor de colas, pero no tiene que detenerse cuando se detiene el gestor de colas.

Conceptos relacionados

"Gestión de servicios" en la página 100
Mediante la utilización del parámetro CONTROL, una instancia de un objeto de servicio se puede iniciar o detener automáticamente con el gestor de colas, o se puede iniciar y detener utilizando los mandatos MQSC START SERVICE y STOP SERVICE.

Gestión de servicios

Mediante la utilización del parámetro CONTROL, una instancia de un objeto de servicio se puede iniciar o detener automáticamente con el gestor de colas, o se puede iniciar y detener utilizando los mandatos MQSC START SERVICE y STOP SERVICE.

Cuando se inicia una instancia de un objeto de servicio, se graba un mensaje en las anotaciones de error del gestor de colas, que contiene el nombre del objeto de servicio y el ID de proceso del proceso iniciado. A continuación se muestra una entrada de anotaciones de ejemplo para un objeto de servicio de servidor que se está iniciando:

```
  02/15/2005 11:54:24 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5028: The Server 'S1' has started. ProcessId(13031).

  EXPLANATION:
  The Server process has started.
  ACTION:
  None.
```

A continuación se muestra una entrada de anotaciones de ejemplo para un objeto de servicio de mandato que se está iniciando:

```
  02/15/2005 11:53:55 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5030: The Command 'C1' has started. ProcessId(13030).

  EXPLANATION:
  The Command has started.
  ACTION:
  None.
```

Cuando se detiene una instancia de un servicio de servidor, se graba un mensaje en las anotaciones de error del gestor de colas, que contiene el nombre del servicio y el ID de proceso del proceso que finaliza. A continuación se muestra una entrada de anotaciones de ejemplo para un objeto de servicio de servidor que se está deteniendo:

```
  02/15/2005 11:54:54 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5029: The Server 'S1' has ended. ProcessId(13031).

  EXPLANATION:
  The Server process has ended.
  ACTION:
  None.
```

Referencia relacionada

"Variables de entorno adicionales" en la página 101
Cuando se inicia un servicio, el entorno en el que se inicia el proceso de servicio se hereda del entorno del gestor de colas. Es posible definir variables de entorno adicionales que se deben establecer en el entorno del proceso de servicio añadiendo las variables que desea definir en uno de los archivos de alteración temporal del entorno `service.env`.

Variables de entorno adicionales

Cuando se inicia un servicio, el entorno en el que se inicia el proceso de servicio se hereda del entorno del gestor de colas. Es posible definir variables de entorno adicionales que se deben establecer en el entorno del proceso de servicio añadiendo las variables que desea definir en uno de los archivos de alteración temporal del entorno `service.env`.

Nota:

Puede añadir las variables de entorno a dos archivos posibles:

- El archivo `service.env` del ámbito de máquina, que se encuentra en `/var/mqm` en sistemas UNIX and Linux o en el directorio de datos seleccionado durante la instalación en sistemas Windows.
- El archivo `service.env` del ámbito de gestor de colas, que se encuentra en el directorio de datos del gestor de colas. Por ejemplo, la ubicación del archivo de alteración temporal de entorno para un gestor de colas llamado QMNAME es:
  - En sistemas UNIX and Linux, `/var/mqm/qmgrs/QMNAME/service.env`
  - En sistemas Windows, `C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\service.env`

Los dos archivos se procesan, si están disponibles, y las definiciones del archivo del ámbito de gestor de colas tienen preferencia sobre las del archivo del ámbito de máquina.

Cualquier variable de entorno se puede especificar en `service.env`. Por ejemplo, si el servicio IBM WebSphere MQ ejecuta varios mandatos, puede ser útil establecer la variable de usuario PATH en el archivo `service.env`. Los valores que establece la variable no pueden ser variables de entorno; por ejemplo CLASSPATH=*%CLASSPATH%* es incorrecto. Asimismo, en Linux PATH=*$PATH*`:/opt/mqm/bin` dará resultados imprevistos.

CLASSPATH debe estar en mayúsculas y la sentencia de vía de acceso de clase sólo puede contener literales. Algunos servicios (telemetría por ejemplo) establecen su propia vía de acceso de clase. La CLASSPATH definida en `service.env` se añade a la misma.

El formato de las variables definidas en el archivo, `service.env`, es una lista de parejas de variables nombre/valor. Cada variable debe definirse en una línea nueva, y cada variable se toma tal como se define explícitamente, incluidos los espacios en blanco. A continuación, se muestra un ejemplo del archivo `service.env`:

```
#********************************************************************#
#*                                                                  *#
#* <N_OCO_COPYRIGHT>                                                *#
#* Licensed Materials - Property of IBM                             *#
#*                                                                  *#
#* 63H9336                                                          *#
#* (C) Copyright IBM Corporation 2005, 2024.                        *#
#*                                                                  *#
#* <NOC_COPYRIGHT>                                                  *#
#*                                                                  *#
#********************************************************************#
#********************************************************************#
#* Module Name: service.env                                         *#
#* Type       : WebSphere MQ service environment file               *#
#* Function   : Define additional environment variables to be set   *#
#*              for SERVICE programs.                               *#
#* Usage      : <VARIABLE>=<VALUE>                                  *#
#*                                                                  *#
#********************************************************************#
MYLOC=/opt/myloc/bin
MYTMP=/tmp
```

```
TRACEDIR=/tmp/trace
MYINITQ=ACCOUNTS.INITIATION.QUEUE
```

Referencia relacionada

“Inserciones reemplazables en definición de servicio” en la página 102
En la definición de un objeto de servicio, es posible sustituir las señales. Los tokens que se sustituyan se reemplazan automáticamente por su texto ampliado cuando se ejecute el programa de servicio. Los tokens de sustitución se pueden obtener de la lista siguiente de señales comunes o de las variables definidas en el archivo service.env.

Inserciones reemplazables en definición de servicio

En la definición de un objeto de servicio, es posible sustituir las señales. Los tokens que se sustituyan se reemplazan automáticamente por su texto ampliado cuando se ejecute el programa de servicio. Los tokens de sustitución se pueden obtener de la lista siguiente de señales comunes o de las variables definidas en el archivo service.env.

A continuación se muestran tokens comunes que se pueden utilizar para sustituir tokens en la definición de un objeto de servicio:

MQ_INSTALL_PATH
La ubicación donde se ha instalado WebSphere MQ.

MQ_DATA_PATH
La ubicación del directorio de datos de WebSphere MQ:

- En sistemas UNIX and Linux, la ubicación del directorio de datos de WebSphere MQ es `/var/mqm/`.
- En sistemas Windows, la ubicación del directorio de datos de WebSphere MQ es el directorio de datos seleccionado durante la instalación de WebSphere MQ

QMNAME
El nombre del gestor de colas actual.

MQ_SERVICE_NAME
Nombre del servicio.

MQ_SERVER_PID
Esta señal sólo la pueden utilizar los argumentos STOPARG y STOPCMD.

Para los objetos de servicio de servidor, esta señal se reemplaza por el ID de proceso del proceso iniciado por los argumentos STARTCMD y STARTARG. De lo contrario, esta señal se reemplaza por 0.

MQ_Q_MGR_DATA_PATH
La ubicación del directorio de datos del gestor de colas.

MQ_Q_MGR_DATA_NAME
El nombre transformado del gestor de colas. Para obtener más información sobre la transformación de nombres, consulte Descripción de los nombre de archivo de WebSphere MQ.

Para utilizar inserciones reemplazables, inserte la señal entre caracteres + en cualquiera de las series de STARTCMD, STARTARG, STOPCMD, STOPARG, STDOUT o STDERR. Si desea obtener ejemplos, consulte el apartado “Ejemplos de utilización de objetos de servicio” en la página 102.

Ejemplos de utilización de objetos de servicio

Los servicios que aparecen en este apartado se escriben con caracteres separadores de vías de acceso de estilo UNIX, excepto cuando se indique lo contrario.

Utilización de un objeto de servicio de servidor

En este ejemplo se muestra cómo definir, utilizar y modificar un objeto de servicio de servidor para iniciar un supervisor desencadenante.

1. Se define un objeto de servicio de servidor utilizando el siguiente mandato MQSC:

```
DEFINE SERVICE(S1) +
```

```
        CONTROL(QMGR) +
        SERVTYPE(SERVER) +
        STARTCMD('+MQ_INSTALL_PATH+bin/runmqtrm') +
        STARTARG('-m +QMNAME+ -q ACCOUNTS.INITIATION.QUEUE') +
        STOPCMD('+MQ_INSTALL_PATH+bin/amqsstop') +
        STOPARG('-m +QMNAME+ -p +MQ_SERVER_PID+')
```

Donde:

+MQ_INSTALL_PATH+ es una señal que representa el directorio de instalación.

+QMNAME+ es una señal que representa el nombre del gestor de colas.

ACCOUNTS.INITIATION.QUEUE es la cola de inicio.

amqsstop es un programa de ejemplo que se proporciona con WebSphere MQ y que solicita al gestor de colas que interrumpa todas las conexiones para el ID de proceso. amqsstop genera mandatos PCF, por lo que el servidor de mandatos debe estar en ejecución.

+MQ_SERVER_PID+ es una señal que representa el ID de proceso que se pasa al programa de detención.

Consulte "Inserciones reemplazables en definición de servicio" en la página 102 para ver una lista de las señales habituales.

2. La próxima vez que se inicie el gestor de colas, se ejecutará una instancia del objeto de servicio del servidor. Sin embargo, iniciaremos una instancia del objeto de servicio de servidor inmediatamente con el mandato MQSC siguiente:

```
START SERVICE(S1)
```

3. El estado del proceso de servicio de servidor se visualiza utilizando el mandato MQSC siguiente:

```
DISPLAY SVSTATUS(S1)
```

4. Este ejemplo ahora muestra cómo modificar el objeto de servicio de servidor y cómo conseguir que las actualizaciones se identifiquen y reciban reiniciando manualmente el proceso de servicio de servidor. El objeto de servicio de servidor se modifica para que la cola de inicio se especifique como JUPITER.INITIATION.QUEUE. Se utiliza el siguiente mandato MQSC:

```
ALTER SERVICE(S1) +
      STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

Nota: Un servicio en ejecución no identificará ninguna actualización en la definición de servicio hasta que se reinicie.

5. El proceso de servicio de servidor se reinicia para que la modificación sea identificada, utilizando los mandatos MQSC siguientes:

```
STOP SERVICE(S1)
```

Seguidos de:

```
START SERVICE(S1)
```

El proceso de servicio de servidor se reinicia e identifica las modificaciones realizadas en el paso "4" en la página 103.

Nota: El mandato MQSC, STOP SERVICE, sólo se puede utilizar si se especifica un argumento STOPCMD en la definición de servicio.

Utilización de un objeto de servicio de mandato

En este ejemplo se muestra cómo definir un objeto de servicio de mandato para iniciar un programa que graba entradas en el registro del sistema del sistema operativo cuando se inicia o se detiene un gestor de colas.

1. El objeto de servicio de mandato se define utilizando el mandato MQSC siguiente:

```
DEFINE SERVICE(S2) +
       CONTROL(QMGR) +
       SERVTYPE(COMMAND) +
       STARTCMD('/usr/bin/logger') +
       STARTARG('Queue manager +QMNAME+ starting') +
       STOPCMD('/usr/bin/logger') +
       STOPARG('Queue manager +QMNAME+ stopping')
```

Donde:

`logger` es el mandato que proporciona el sistema UNIX and Linux para grabar en registro del sistema.

+QMNAME+ es una señal que representa el nombre del gestor de colas.

Utilización de un objeto de servicio de mandato cuando un gestor de colas sólo finaliza

En este ejemplo se muestra cómo definir un objeto de servicio de mandato para iniciar un programa que graba entradas en el registro del sistema del sistema operativo cuando un gestor de colas sólo se detiene.

1. El objeto de servicio de mandato se define utilizando el mandato MQSC siguiente:

```
DEFINE SERVICE(S3) +
       CONTROL(QMGR) +
       SERVTYPE(COMMAND) +
       STOPCMD('/usr/bin/logger') +
       STOPARG('Queue manager +QMNAME+ stopping')
```

Donde:

`logger` es un programa de ejemplo que se proporciona con WebSphere MQ que puede grabar entradas en el registro del sistema del sistema operativo.

+QMNAME+ es una señal que representa el nombre del gestor de colas.

Más información sobre el paso de argumentos

En este ejemplo se muestra cómo definir un objeto de servicio de servidor para iniciar un programa llamado `runserv` cuando se inicie un gestor de colas.

Este ejemplo se ha escrito con caracteres separadores de vías de acceso de estilo Windows.

Uno de los argumentos que se va a pasar al programa que se está iniciando es una serie que contiene un espacio. Este argumento debe pasarse como una sola serie de caracteres; Para ello, se utilizan comillas dobles, tal como se muestra en el mandato siguiente, para definir el objeto de servicio de mandato:

1. El objeto de servicio de servidor se define utilizando el mandato MQSC siguiente:

```
   DEFINE SERVICE(S1) SERVTYPE(SERVER) CONTROL(QMGR) +
     STARTCMD('C:\Program Files\Tools\runserv.exe') +
     STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
     STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')


DEFINE SERVICE(S4) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('C:\Program Files\Tools\runserv.exe') +
       STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
       STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')
```

Donde:

+QMNAME+ es una señal que representa el nombre del gestor de colas.

`"C:\Program Files\Tools\"` es una serie que contiene un espacio, que se pasará como una única serie.

Inicio automático de un servicio

En este ejemplo se muestra cómo definir un objeto de servicio del servidor para que inicie automáticamente el supervisor de activaciones cuando se inicie el gestor de colas.

1. El objeto de servicio de servidor se define utilizando el mandato MQSC siguiente:

```
DEFINE SERVICE(TRIG_MON_START) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('runmqtrm') +
       STARTARG('-m +QMNAME+ -q +IQNAME+')
```

   Donde:

   +QMNAME+ es una señal que representa el nombre del gestor de colas.

   +IQNAME+ es una variable de entorno que define el usuario en uno de los archivos service.env y que representa el nombre de la cola de inicio.

Gestión de objetos para desencadenamiento

WebSphere MQ le permite iniciar una aplicación automáticamente cuando se cumplen determinadas condiciones en una cola. Por ejemplo, es posible que desee iniciar una aplicación cuando el número de mensaje de una cola alcanza un número especificado. Este recurso se denomina *desencadenamiento*. Deberá definir los objetos que dan soporte al desencadenamiento.

El desencadenamiento se describe con detalle en la sección Iniciar aplicaciones WebSphere MQ utilizando desencadenantes.

Definición de una cola de aplicación para el desencadenamiento

Una cola de aplicación es una cola local que las aplicaciones utilizan para el envío de mensajes, mediante la MQI. El desencadenamiento requiere definir varios atributos de cola en la cola de aplicación.

El desencadenamiento en sí se habilita mediante el atributo *Trigger* (TRIGGER en los mandatos MQSC). En este ejemplo, se va a generar un suceso desencadenante cuando haya 100 mensajes con una prioridad 5 o superior en la cola local MOTOR.INSURANCE.QUEUE, como se indica a continuación:

```
DEFINE QLOCAL (MOTOR.INSURANCE.QUEUE) +
       PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
       MAXMSGL (2000) +
       DEFPSIST (YES) +
       INITQ (MOTOR.INS.INIT.QUEUE) +
       TRIGGER +
       TRIGTYPE (DEPTH) +
       TRIGDPTH (100)+
       TRIGMPRI (5)
```

donde:

QLOCAL (MOTOR.INSURANCE.QUEUE)
Es el nombre de la cola de aplicación que se define.

PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
Es el nombre de la definición de proceso que describe la aplicación que se va a iniciar mediante un programa supervisor desencadenante.

MAXMSGL (2000)
Especifica la longitud máxima de los mensajes de la cola.

DEFPSIST (YES)
Especifica que los mensajes de esta cola son persistentes de forma predeterminada.

INITQ (MOTOR.INS.INIT.QUEUE)
Es el nombre de la cola de inicio a la que el gestor de colas va a transferir el mensaje desencadenante.

TRIGGER
Es el valor del atributo de desencadenamiento.

TRIGTYPE (DEPTH)
Especifica que se genera un suceso desencadenante cuando el número de mensajes de la prioridad obligatoria (TRIGMPRI) alcanza el número especificado en TRIGDPTH.

TRIGDPTH (100)
Especifica el número de mensajes necesarios para generar un suceso desencadenante.

TRIGMPRI (5)
Es la prioridad de los mensajes que el gestor de colas va a incluir en el recuento para decidir si va a generar un suceso desencadenante. Sólo se incluyen en el recuento los mensajes con prioridad 5 o superior.

Definición de una cola de inicio

Cuando se produce un suceso desencadenante, el gestor de colas coloca un mensaje desencadenante en la cola de inicio especificada en la definición de la cola de aplicación. Las colas de inicio no tienen valores especiales, pero puede utilizar como guía la siguiente definición de la cola local MOTOR.INS.INIT.QUEUE:

```
DEFINE QLOCAL(MOTOR.INS.INIT.QUEUE) +
       GET (ENABLED) +
       NOSHARE +
       NOTRIGGER +
       MAXMSGL (2000) +
       MAXDEPTH (1000)
```

Definición de un proceso

Utilice el mandato DEFINE PROCESS para crear una definición de proceso. Una definición de proceso define la aplicación que se va a utilizar para procesar los mensajes de la cola de aplicación. La definición de cola de aplicación nombra el proceso que se va a utilizar y, de este modo, asocia la cola de aplicación con la aplicación que se va a utilizar para procesar sus mensajes. Esto se hace mediante el atributo PROCESS en la cola de aplicación MOTOR.INSURANCE.QUEUE. El siguiente mandato MQSC define el proceso necesario, MOTOR.INSURANCE.QUOTE.PROCESS, identificado en este ejemplo:

```
DEFINE PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
               DESCR ('Insurance request message processing') +
               APPLTYPE (UNIX) +
               APPLICID ('/u/admin/test/IRMP01') +
               USERDATA ('open, close, 235')
```

Donde:

MOTOR.INSURANCE.QUOTE.PROCESS
Es el nombre de la definición de proceso.

DESCR ('Insurance request message processing')
Describe el programa de aplicación relacionado con esta definición. Este texto Aparece cuando utiliza el mandato DISPLAY PROCESS y sirve para ayudarle a identificar lo que hace el proceso. Si utiliza espacios en la serie, debe colocarla entre comillas simples.

APPLTYPE (UNIX)
Es el tipo de aplicación que debe iniciarse.

APPLICID ('/u/admin/test/IRMP01')
Es el nombre del archivo ejecutable de la aplicación, especificado como nombre de archivo totalmente calificado. En sistemas Windows, un valor típico de `APPLICID` sería `c:\appl\test\irmp01.exe`.

USERDATA ('open, close, 235')
Son datos definidos por el usuario, que la aplicación puede utilizar.

Visualización de los atributos de una definición de proceso

Utilice el mandato DISPLAY PROCESS para examinar los resultados de la definición. Por ejemplo:

```
DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)


    24 : DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) ALL
AMQ8407: Display Process details.
    DESCR ('Insurance request message processing')
    APPLICID ('/u/admin/test/IRMP01')
    USERDATA (open, close, 235)
    PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
    APPLTYPE (UNIX)
```

También puede utilizar el mandato MQSC ALTER PROCESS para modificar una definición de proceso existente, y el mandato DELETE PROCESS para suprimir una definición de proceso.

Administración de objetos de IBM WebSphere MQ remotos

En esta sección se explica cómo administrar objetos de IBM WebSphere MQ en un gestor de colas remoto utilizando mandatos MQSC, y cómo utilizar objetos de cola remota para controlar el destino de los mensajes y los mensajes de respuesta.

Este apartado describe:

Canales, clústeres y gestión de colas remotas

Un gestor de colas se comunica con otro gestor de colas enviando un mensaje y, si es necesario, recibiendo una respuesta. El gestor de colas receptor puede estar:

- En la misma máquina
- En otra máquina en la misma ubicación o incluso en el otro lado del mundo
- Ejecutándose en la misma plataforma que el gestor de colas local
- Ejecutándose en otra plataforma soportada por WebSphere MQ

Estos mensajes pueden proceder de:

- Programas de aplicación escritos por el usuario que transfieren datos de un nodo a otro.
- Aplicaciones de administración escritas por el usuario que utilizan mandatos PCF o MQAI
- IBM WebSphere MQ Explorer.
- Gestores de colas que envíen:
  - Mensajes de sucesos de instrumentación a otro gestor de colas
  - Mandatos MQSC emitidos desde un mandato `runmqsc` en modalidad indirecta (donde los mandatos se ejecutan en otro gestor de colas)

Para que un mensaje pueda enviarse a un gestor de colas remoto, el gestor de colas local necesita un mecanismo para detectar la llegada de mensajes y poder transportarlos. Debe estar formado por:

- Un canal como mínimo
- Una cola de transmisión
- Un iniciador de canal

Para que un gestor de colas remoto reciba un mensaje, es necesario un escucha.

Un canal es un enlace de comunicaciones unidireccional entre dos gestores de colas que puede transportar mensajes destinados a cualquier número de colas del gestor de colas remoto.

Cada extremo del canal tiene una definición distinta. Por ejemplo, si un extremo es un emisor o un servidor, el otro extremo debe ser un receptor o un peticionario. Un solo canal consta de una *definición de canal emisor* en el extremo del gestor de colas local y de una *definición de canal receptor* en el extremo del gestor de colas remoto. Las dos definiciones deben tener el mismo nombre y, juntas, formar un solo canal de mensajes.

Si desea que el gestor de colas remoto responda a mensajes enviados por el gestor de colas local, deberá configurar un segundo canal para devolver las respuestas al gestor de colas local.

Utilice el mandato MQSC DEFINE CHANNEL para definir canales. En esta sección, los ejemplos relacionados con canales utilizan los atributos de canal predeterminados a menos que se indique lo contrario.

En cada extremo de un canal hay un agente de canal de mensajes (MCA) que controla el envío y la recepción de mensajes. El MCA extrae los mensajes de la cola de transmisión y los transfiere al enlace de comunicaciones entre gestores de colas.

Una cola de transmisión es una cola local especializada que contiene temporalmente mensajes antes de que el MCA los extraiga y los envíe al gestor de colas remoto. El nombre de la cola de transmisión se especifica en una *definición de cola remota*.

Puede permitir que MCA transfiera los mensajes utilizando varias hebras. Este proceso se conoce como *canalización*. El proceso de canalización permite que MCA transfiera los mensajes de forma más eficaz y mejora el rendimiento de los canales. Consulte Atributos de canales para obtener más detalles sobre cómo configurar un canal para utilizar el proceso de canalización.

En el apartado “Preparación de canales y colas de transmisión para la administración remota” en la página 110 se explica cómo utilizar dichas definiciones para configurar la administración remota.

Para obtener más información sobre la configuración de la gestión de colas distribuidas en general, consulte Componentes de colas distribuidas.

Administración remota utilizando clústeres

En una red WebSphere MQ que utilice la gestión de colas distribuidas, cada gestor de colas es independiente. Si un gestor de colas necesita enviar mensajes a otro gestor de colas, éste debe definir una cola de transmisión, un canal para el gestor de colas remoto y una definición de cola remota para cada cola a la que desea enviar mensajes.

Un *clúster* es un grupo de gestores de colas configurados de forma que puedan comunicarse directamente entre ellos a través de una sola red, sin definiciones complejas de colas de transmisión, canales y colas. Los clústeres pueden configurarse con facilidad y, normalmente, contienen gestores de colas que están relacionados lógicamente de algún modo y necesitan compartir datos o aplicaciones. Incluso los clústeres más pequeños reducen los costes de administración del sistema.

El establecimiento de una red de gestores de colas en un clúster requiere menos definiciones que el establecimiento de un entorno de gestión de colas distribuidas tradicional. Como es necesario efectuar menos definiciones, la red puede configurarse o modificarse más rápida y fácilmente y el riesgo de cometer errores en las definiciones disminuye.

Para configurar un clúster, se necesita una definición de clúster emisor (CLUSSDR) y una definición de clúster receptor (CLUSRCVR) para cada gestor de colas. No se necesita ninguna definición de cola de transmisión ni de cola remota. Los principios de la administración remota son los mismos cuando se utilizan dentro de un clúster, pero las definiciones propiamente dichas son significativamente más sencillas.

Para obtener más información sobre clústeres, sus atributos y cómo configurarlos, consulte Clústeres de gestores de colas.

Administración remota desde un gestor de colas local

En este apartado se explica cómo administrar un gestor de colas remoto desde un gestor de colas local utilizando mandatos MQSC y PCF.

La preparación de las colas y de los canales es esencialmente igual para mandatos MQSC y para mandatos PCF. En este apartado, los ejemplos muestran mandatos MQSC porque son más fáciles de entender. Para obtener más información sobre cómo escribir programas de administración utilizando mandatos PCF, consulte “Utilización de formatos de mandato programables” en la página 10.

Los mandatos MQSC se pueden enviar a un gestor de colas remoto interactivamente o desde un archivo de texto que contenga los mandatos. El gestor de colas remoto puede estar en la misma máquina o en una máquina distinta, que suele ser lo normal. Puede administrar de forma remota los gestores de colas en otros entornos de WebSphere MQ, entre ellos los sistemas UNIX and Linux, sistemas Windows, IBM i y z/OS.

Para implementar la administración remota debe crear objetos específicos. A menos que tenga requisitos especializados, los valores predeterminados (por ejemplo, para la longitud máxima de mensaje) son suficientes.

Preparación de los gestores de colas para la administración remota

Cómo utilizar los mandatos MQSC para preparar los gestores de colas para la administración remota.

La Figura 17 en la página 109 muestra la configuración de los gestores de colas y canales necesarios para la administración remota mediante el mandato **runmqsc**. El objeto `source.queue.manager` es el gestor de colas de origen desde el que puede emitir mandatos MQSC y al que se devuelven los resultados de estos mandatos (mensajes de operador). El objeto `target.queue.manager` es el nombre del gestor de colas de destino, que procesa los mandatos y genera todos los mensajes de operador.

Nota: Si utiliza **runmqsc** con la opción -w, `source.queue.manager` ***debe*** ser el gestor de colas predeterminado. Para obtener más información sobre la creación de un gestor de colas, consulte crtmqm.

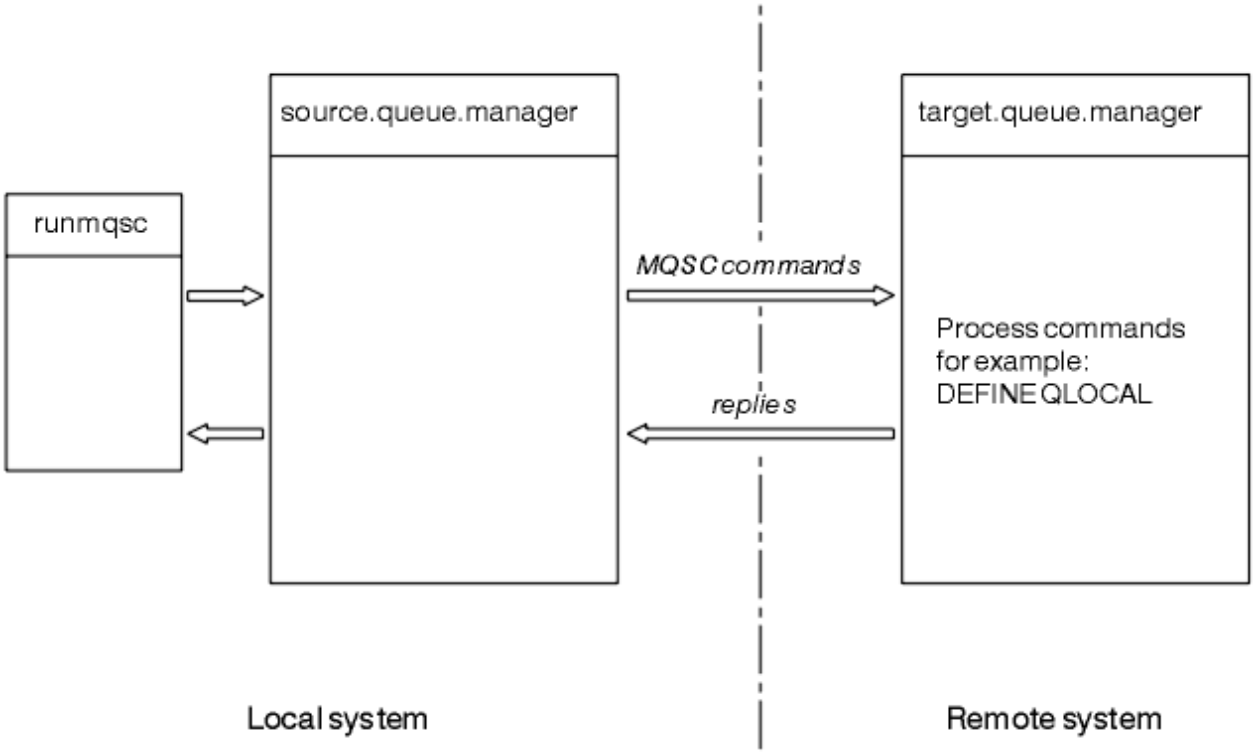


Figura 17. Administración remota utilizando mandatos MQSC

En ambos sistemas, si todavía no lo ha hecho:

- Cree el gestor de colas y los objetos predeterminados, utilizando el mandato `crtmqm`.
- Inicie el gestor de colas, utilizando el mandato `strmqm`.

En el gestor de colas de destino:

- La cola de mandatos, SYSTEM.ADMIN.COMMAND.QUEUE, debe estar presente. Esta cola se crea de forma predeterminada cuando se crea un gestor de colas.

Tiene que ejecutar estos mandatos localmente o a través de un recurso de red, por ejemplo Telnet.

Preparación de canales y colas de transmisión para la administración remota

Cómo utilizar los mandatos MQSC para preparar los canales y colas de transmisión para la administración remota.

Para ejecutar mandatos MQSC de forma remota, configure dos canales, uno para cada dirección, y sus colas de transmisión asociadas. En este ejemplo se presupone que el tipo de transporte utilizado es TCP/IP y que se conoce la dirección TCP/IP relacionada.

El canal `source.to.target` es para enviar mandatos MQSC del gestor de colas de origen al gestor de colas de destino. Su emisor está en `source.queue.manager` y su receptor está en `target.queue.manager`. El canal `target.to.source` es para devolver la salida de los mandatos y cualquier mensaje de operador que se genere para el gestor de colas origen. También debe definir una cola de transmisión para cada canal. Esta cola es una cola local que recibe el nombre del gestor de colas receptor. El nombre de XMITQ debe coincidir con el nombre del gestor de colas remoto para que la administración remota funcione, a menos que esté utilizando un alias de gestor de colas.Figura 18 en la página 110 resume esta configuración.

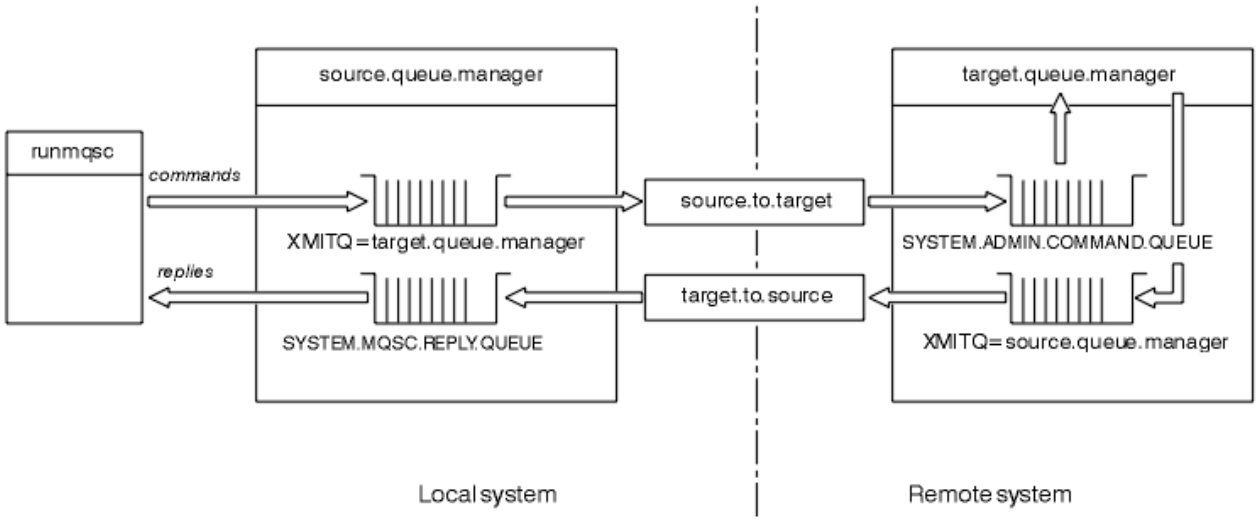


Figura 18. Preparación de canales y colas para la administración remota

Consulte Conexión de aplicaciones utilizando colas distribuidas para obtener más información sobre la configuración de canales.

Definición de canales, escuchas y colas de transmisión

En el gestor de colas de origen (`source.queue.manager`), emita los siguientes mandatos MQSC para definir los canales, el escucha y la cola de transmisión:

1. Defina el canal emisor en el gestor de colas de origen:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(SDR) +
       CONNAME (RHX5498) +
       XMITQ ('target.queue.manager') +
       TRPTYPE(TCP)
```

2. Defina el canal receptor en el gestor de colas de origen:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Defina el escucha en el gestor de colas de origen:

```
DEFINE LISTENER ('source.queue.manager') +
       TRPTYPE (TCP)
```

4. Defina la cola de transmisión en el gestor de colas de origen:

```
DEFINE QLOCAL ('target.queue.manager') +
       USAGE (XMITQ)
```

Emita los mandatos siguientes en el gestor de colas de destino (`target.queue.manager`), para crear los canales, el escucha y la cola de transmisión:

1. Defina el canal emisor en el gestor de colas de destino:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(SDR) +
       CONNAME (RHX7721) +
       XMITQ ('source.queue.manager') +
       TRPTYPE(TCP)
```

2. Defina el canal receptor en el gestor de colas de destino:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Defina el escucha en el gestor de colas de destino:

```
DEFINE LISTENER ('target.queue.manager') +
       TRPTYPE (TCP)
```

4. Defina la cola de transmisión en el gestor de colas de destino:

```
DEFINE QLOCAL ('source.queue.manager') +
       USAGE (XMITQ)
```

Nota: Los nombres de conexión TCP/IP especificados para el atributo CONNAME en las definiciones de canal emisor son meramente ilustrativos. Se trata del nombre de red de la máquina que se encuentra en el *otro* extremo de la conexión. Utilice los valores adecuados para su red.

Inicio de los escuchas y los canales

Cómo utilizar los mandatos MQSC para iniciar los escuchas y los canales.

Inicie los dos escuchas utilizando los siguientes mandatos MQSC:

1. Inicie el escucha en el gestor de colas de origen, `source.queue.manager`, emitiendo el siguiente mandato MQSC:

```
START LISTENER ('source.queue.manager')
```

2. Inicie el escucha en el gestor de colas de destino, `target.queue.manager`, emitiendo el siguiente mandato MQSC:

```
START LISTENER ('target.queue.manager')
```

Inicie los dos canales emisores utilizando los siguientes mandatos MQSC:

1. Inicie el canal emisor en el gestor de colas de origen, `source.queue.manager`, emitiendo el siguiente mandato MQSC:

```
START CHANNEL ('source.to.target')
```

2. Inicie el canal emisor en el gestor de colas de destino, `target.queue.manager`, emitiendo el siguiente mandato MQSC:

```
START CHANNEL ('target.to.source')
```

Definición automática de canales
Puede habilitar la definición automática de definiciones de conexión con el servidor y con el receptor actualizando el objeto gestor de colas, mediante el mandato MQSC, ALTER QMGR (o el mandato PCF Modificar gestor de colas).

Si WebSphere MQ recibe una petición de conexión de entrada y no puede encontrar un canal receptor o de conexión de servidor apropiado, crea un canal automáticamente. Las definiciones automáticas se basan en dos definiciones predeterminadas proporcionadas con WebSphere MQ: SYSTEM.AUTO.RECEIVER y SYSTEM.AUTO.SVRCONN.

Si desea más información sobre cómo crear definiciones de canal automáticamente, consulte Preparación de canales. Si desea más información sobre cómo definir automáticamente canales para clústeres, consulte Definición automáticamente de canales de clúster.

Gestión del servidor de mandatos para la administración remota

Cómo iniciar, detener y visualizar el estado del servidor de mandatos. Es indispensable tener un servidor de mandatos para toda la administración que implique mandatos PCF, la MQAI y también para la administración remota.

Cada gestor de colas puede tener un servidor de mandatos asociado. El servidor de mandatos procesa todos los mandatos entrantes procedentes de gestores de colas remotos o los mandatos PCF procedentes de aplicaciones. Presenta los mandatos al gestor de colas para que los procese y devuelve un código de terminación o un mensaje de operador, dependiendo del origen del mandato.

Nota: En la administración remota, debe asegurarse de que el gestor de colas de destino está en ejecución. De lo contrario, los mensajes que contienen mandatos no pueden salir del gestor de colas desde el que se han emitido. En vez de ello, estos mensajes se transfieren a la cola de transmisión local que sirve al gestor de colas remoto. Evite esta situación.

Existen mandatos de control individuales para iniciar y detener el servidor de mandatos. Siempre que el servidor de mandatos se esté ejecutando, los usuarios de WebSphere MQ para Windows o WebSphere MQ para Linux (plataformasx86 y x86-64) pueden realizar las operaciones descritas en las secciones siguientes utilizando IBM WebSphere MQ Explorer. Para obtener más información, consulte “Administración utilizando IBM WebSphere MQ Explorer” en la página 57.

Iniciar el servidor de mandatos

En función del valor del atributo del gestor de colas, *SCMDSERV*, el servidor de mandatos se inicia automáticamente cuando se inicia el gestor de colas o debe iniciarse manualmente. El valor del atributo de gestor de colas se puede modificar utilizando el mandato MQSC `ALTER QMGR` especificando el parámetro SCMDSERV. Por omisión, el servidor de mandatos se inicia automáticamente.

Si *SCMDSERV* está establecido en MANUAL, inicie el servidor de mandatos utilizando el mandato:

```
strmqcsv saturn.queue.manager
```

donde `saturn.queue.manager` es el gestor de colas para el que se inicia el servidor de mandatos.

Visualización del estado del servidor de mandatos

Para la administración remota, debe asegurarse de que el servidor de mandatos del gestor de colas de destino está ejecutándose. Si no se está ejecutando, no se podrá procesar ningún mandato remoto. Todos los mensajes que contienen mandatos se ponen en la cola de mandatos del gestor de colas de destino.

Para visualizar el estado del servidor de mandatos de un gestor de colas, emita el siguiente mandato MQSC:

```
DISPLAY QMSTATUS CMDSERV
```

Detención de un servidor de mandatos

Para detener el servidor de mandatos que se inició en el ejemplo anterior, utilice el mandato siguiente:

```
endmqcsv saturn.queue.manager
```

Hay dos formas de detener el servidor de mandatos:

- Para una conclusión controlada, utilice el mandato `endmqcsv` con el indicador -c, que es el valor predeterminado.
- Para una detención inmediata, utilice el mandato `endmqcsv` con el indicador -i.

Nota: Al detener un gestor de colas también se detiene el servidor de mandatos asociado al mismo.

Emisión de mandatos MQSC en un gestor de colas remoto

Puede utilizar un formulario concreto del mandato **`runmqsc`** para ejecutar los mandatos MQSC en un gestor de colas remoto.

El servidor de mandatos ***debe*** estar en ejecución en el gestor de colas de destino, si va a procesar mandatos MQSC de forma remota. (Esto no es necesario en el gestor de colas de origen). Para obtener más información sobre cómo iniciar el servidor de mandatos en un gestor de colas, consulte el apartado “Gestión del servidor de mandatos para la administración remota” en la página 112.

A continuación, en el gestor de colas de origen, puede ejecutar mandatos MQSC interactivamente en modalidad indirecta, escribiendo:

```
runmqsc -w 30 target.queue.manager
```

Esta forma del mandato `runmqsc`, con el indicador -w, ejecuta los mandatos MQSC en modalidad indirecta, en la que los mandatos se colocan (modificados) en la cola de entrada del servidor de mandatos y se ejecutan en orden.

Cuando teclea un mandato MQSC, éste se redirige al gestor de colas remoto, en este caso `target.queue.manager`. El tiempo de espera está establecido en 30 segundos; si al cabo de 30 segundos no se ha recibido una respuesta, se genera el siguiente mensaje en el gestor de colas local (de origen):

```
AMQ8416: MQSC timed out waiting for a response from the command server.
```

Cuando acabe de emitir mandatos MQSC, el gestor de colas local mostrará cualquier respuesta con tiempo de espera excedido que haya llegado y descartará todas las respuestas posteriores.

El gestor de colas origen predeterminado es el gestor de colas local predeterminado. Si especifica la opción `-m` *NombreGestColasLocal* en el mandato **runmqsc**, puede dirigir la emisión de mandatos a través de cualquier gestor local de colas.

En modalidad indirecta, también puede ejecutar un archivo de mandatos MQSC en un gestor de colas remoto. Por ejemplo:

```
runmqsc -w 60 target.queue.manager < mycomds.in > report.out
```

donde `mycomds.in` es un archivo que contiene mandatos MQSC y `report.out` es el archivo de informe.

Método recomendado para emitir mandatos de forma remota

Cuando emita mandatos en un gestor de colas remoto, considere la posibilidad de utilizar el enfoque siguiente:

1. Coloque los mandatos MQSC que van a ejecutarse en el sistema remoto en un archivo de mandatos.
2. Verifique los mandatos MQSC localmente, especificando el indicador -v en el mandato `runmqsc`.

   No puede utilizar `runmqsc` para verificar mandatos MQSC en otro gestor de colas.
3. Compruebe que el archivo de mandatos se ejecuta localmente sin errores.
4. Ejecute el archivo de mandatos en el sistema remoto.

Si tiene problemas al usar mandatos MQSC de forma remota

Si tiene dificultades para ejecutar mandatos MQSC de forma remota, asegúrese de que ha:

- Iniciado el servidor de mandatos en el gestor de colas de destino.
- Definido una cola de transmisión válida.
- Definido los dos extremos de los canales de mensajes para:
  - El canal por el que se están enviando los mandatos.
  - El canal por el que se van a devolver las respuestas.
- Especificado el nombre de conexión (CONNAME) correcto en la definición de canal.
- Iniciado los escuchas antes de iniciar los canales de mensajes.
- Comprobado que el intervalo de desconexión no ha expirado, por ejemplo, si un canal se ha iniciado pero se ha cerrado al cabo de un tiempo. Esto es especialmente importante si inicia manualmente los canales.
- Enviado solicitudes desde un gestor de colas de origen que no tengan sentido para el gestor de colas de destino (por ejemplo, solicitudes que incluyan parámetros que no están soportados en el gestor de colas remoto).

Consulte también “Resolución de problemas con mandatos MQSC” en la página 82.

Creación de una definición local de una cola remota

Una definición local de una cola remota es una definición en un gestor de colas local que hace referencia a una cola en un gestor de colas remoto.

No es necesario que defina una cola remota desde una posición local, pero la ventaja de hacerlo es que las aplicaciones pueden hacer referencia a la cola remota por su nombre definido localmente, en lugar de tener que especificar un nombre que esté calificado con el ID del gestor de colas en el que está situado la cola remota.

Descripción del modo de trabajar de definiciones locales de colas remotas

Una aplicación se conecta a un gestor de colas local y luego emite una llamada MQOPEN. En la llamada de apertura, el nombre de cola especificado es el de una definición de cola remota existente en el gestor de colas local. La definición de cola remota suministra los nombres de la cola de destino, del gestor de colas de destino y, opcionalmente, de una cola de transmisión. Para colocar un mensaje en la cola remota, la aplicación emite una llamada MQPUT , especificando el descriptor de contexto devuelto por la llamada MQOPEN . El gestor de colas utiliza el nombre de la cola remota y el del gestor de colas remoto en una cabecera de transmisión que se añade al principio del mensaje. Esta información se utiliza para dirigir el mensaje a su destino correcto en la red.

Como administrador, puede controlar el destino del mensaje alterando la definición de cola remota.

En el ejemplo siguiente se muestra cómo una aplicación transfiere un mensaje en una cola que es propiedad de un gestor de colas remoto. La aplicación se conecta a un gestor de colas, por ejemplo, `saturn.queue.manager`. La cola de destino es propiedad de otro gestor de colas.

En la llamada MQOPEN, la aplicación especifica estos campos:

Valor de campo	Descripción
ObjectName CYAN.REMOTE.QUEUE	Especifica el nombre local del objeto de cola remota. Esto define la cola de destino y el gestor de colas de destino.
ObjectType (Cola)	Identifica este objeto como una cola.
ObjectQmgrName En blanco o `saturn.queue.manager`	Este campo es opcional. Si está en blanco, se presupone el nombre del gestor de colas local. (Es el gestor de colas donde se encuentra la definición de cola remota).

A continuación, la aplicación emite una llamada MQPUT para colocar un mensaje en esta cola.

En el gestor de colas local, puede crear una definición local de una cola remota utilizando los siguientes mandatos MQSC:

```
DEFINE QREMOTE (CYAN.REMOTE.QUEUE) +
       DESCR ('Queue for auto insurance requests from the branches') +
       RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE) +
       RQMNAME (jupiter.queue.manager) +
       XMITQ (INQUOTE.XMIT.QUEUE)
```

donde:

QREMOTE (CYAN.REMOTE.QUEUE)
Especifica el nombre local del objeto de cola remota. Éste es el nombre que las aplicaciones conectadas a este gestor de colas deben especificar en la llamada MQOPEN para abrir la cola AUTOMOBILE.INSURANCE.QUOTE.QUEUE del gestor de colas remoto `jupiter.queue.manager`.

DESCR ('Queue for auto insurance requests from the branches')
Proporciona un texto adicional que describe el uso de la cola.

RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)
Especifica el nombre de la cola de destino del gestor de colas remoto. Ésta es la cola de destino real para los mensajes enviados por aplicaciones que especifiquen el nombre de cola CYAN.REMOTE.QUEUE. La cola AUTOMOBILE.INSURANCE.QUOTE.QUEUE debe definirse como una cola local en el gestor de colas remoto.

RQMNAME (jupiter.queue.manager)
Especifica el nombre del gestor de colas remoto al que pertenece la cola de destino AUTOMOBILE.INSURANCE.QUOTE.QUEUE.

XMITQ (INQUOTE.XMIT.QUEUE)
Especifica el nombre de la cola de transmisión. Es opcional; si no se especifica el nombre de una cola de transmisión, se utiliza una cola con el mismo nombre que el gestor de colas remoto.

En cualquier caso, la cola de transmisión adecuada debe definirse como una cola local con un atributo *Usage* que especifique que es una cola de transmisión (USAGE(XMITQ) en los mandatos MQSC).

Forma alternativa de colocar mensajes en una cola remota

Utilizar una definición local de una cola remota no es la única manera de colocar mensajes en una cola remota. Las aplicaciones pueden especificar el nombre de cola completo, incluido el nombre del gestor de colas remoto, como parte de la llamada MQOPEN. En este caso, no se requiere una definición local de una cola remota. No obstante, esto significa que las aplicaciones deben conocer o tener acceso al nombre del gestor de colas remoto durante la ejecución.

Utilización de otros mandatos con colas remotas

Puede utilizar mandatos MQSC para visualizar o alterar los atributos de un objeto cola remota, o puede suprimir el objeto de cola remota. Por ejemplo:

- Para visualizar los atributos de la cola remota:

```
DISPLAY QUEUE (CYAN.REMOTE.QUEUE)
```

- Para cambiar la cola remota para habilitar las transferencias. Esto no afecta a la cola de destino, sólo a las aplicaciones que especifiquen esta cola remota:

```
ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)
```

- Para suprimir esta cola remota. Esto no afecta a la cola de destino, sólo a su definición local:

```
  DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

Nota: Cuando se suprime una cola remota, sólo se suprime la representación local de la cola remota. No se suprime la cola remota en sí ni ningún mensaje que se encuentre en ella.

Definición de una cola de transmisión

Una cola de transmisión es una cola local que se utiliza cuando un gestor de colas envía mensajes a un gestor de colas remoto a través de un canal de mensajes.

El canal proporciona un enlace unidireccional con el gestor de colas remoto. Los mensajes se ponen en la cola de transmisión en espera de que el canal pueda aceptarlos. Cuando se define un canal, debe especificarse un nombre de cola de transmisión en el extremo emisor del canal de mensajes.

El atributo USAGE de los mandatos MQSC define si una cola es una cola de transmisión o una cola normal.

Colas de transmisión predeterminadas

Cuando un gestor de colas envía mensajes a un gestor de colas remoto, identifica la cola de transmisión mediante la secuencia siguiente:

1. La cola de transmisión nombrada en el atributo XMITQ de la definición local de una cola remota.
2. Una cola de transmisión con el mismo nombre que el gestor de colas de destino. (Este valor es el valor predeterminado en XMITQ de la definición local de una cola remota.)
3. La cola de transmisión nombrada en el atributo DEFXMITQ del gestor de colas local.

Por ejemplo, el siguiente mandato MQSC crea una cola de transmisión predeterminada en `source.queue.manager` para los mensajes que van a `target.queue.manager`:

```
DEFINE QLOCAL ('target.queue.manager') +
       DESCR ('Default transmission queue for target qm') +
       USAGE (XMITQ)
```

Las aplicaciones pueden transferir mensajes directamente a una cola de transmisión aunque también pueden transferirse indirectamente a través de una definición de cola remota. Consulte también el apartado “Creación de una definición local de una cola remota” en la página 114.

Utilización de definiciones de colas remotas como alias

Además de para localizar una cola de otro gestor de colas, una definición local de una cola remota también puede utilizarse para los alias de gestor de colas y los alias de cola de respuesta. Ambos tipos de alias se resuelven mediante la definición local de una cola remota. Debe definir los canales adecuados para que el mensaje llegue a su destino.

Alias de gestor de colas

Un alias es el proceso por el cual el nombre del gestor de colas de destino, tal como se especifica en un mensaje, es modificado por un gestor de colas en la ruta del mensaje.Los alias de gestor de colas son importantes porque se pueden utilizar para controlar el destino de los mensajes dentro de una red de gestores de colas.

Para hacer esto debe alterar la definición de cola remota del gestor de colas en el punto de control. La aplicación emisora no sabe que el nombre de gestor de colas especificado es un alias.

Para obtener más información sobre los alias de gestor de colas, consulte ¿Qué son los alias?.

Alias de cola de respuesta

De forma opcional, una aplicación puede especificar el nombre de una cola de respuesta cuando coloca un *mensaje de solicitud* en una cola.

Si la aplicación que procesa el mensaje extrae el nombre de la cola de respuesta, sabe dónde enviar el *mensaje de respuesta*, si fuera necesario.

Un alias de cola de respuesta es el proceso por el cual una cola de respuesta, tal como se especifica en un mensaje de solicitud, es modificada por un gestor de colas en la ruta del mensaje. La aplicación emisora no sabe que el nombre de cola de respuesta especificado es un alias.

Un alias de cola de respuesta le permite modificar el nombre de la cola de respuesta y, opcionalmente, de su gestor de colas. Esto, a su vez, le permite controlar qué ruta se utiliza para los mensajes de respuesta.

Para obtener más información sobre mensajes de solicitud, mensajes de respuesta y colas de respuesta, consulte las secciones Tipos de mensajes y Cola de respuesta y gestor de colas.

Para obtener más información sobre los alias de cola de respuesta, consulte Alias de cola de respuesta y clústeres .

Conversión de datos entre juegos de caracteres codificados

El gestor de colas puede convertir los datos de mensajes en formatos definidos por WebSphere MQ (también conocidos como *formatos incorporados*) de un juego de caracteres codificado a otro, siempre y cuando ambos juegos de caracteres codificados se refieran a un solo idioma o a un grupo de idiomas similares.

Por ejemplo, está soportada la conversión entre los juegos de caracteres codificados con los identificadores (CCSID) 850 y 500, porque ambos se aplican a idiomas de Europa Occidental.

Para las conversiones de caracteres de (NL) de nueva línea de EBCDIC a ASCII, consulte Todos los gestores de colas.

Las conversiones soportadas están definidas en Conversión de datos.

Cuando un gestor de colas no puede convertir mensajes en formatos incorporados

El gestor de colas no puede convertir automáticamente mensajes en formatos incorporados si sus CCSID respectivos representan grupos de idiomas nacionales distintos. Por ejemplo, la conversión entre el CCSID 850 y el CCSID 1025 (que es un juego de caracteres codificado EBCDIC para idiomas que utilizan el alfabeto cirílico) no está soportada porque muchos de los caracteres de uno de los juegos de caracteres codificados no pueden representarse en el otro. Si tiene una red de gestores de colas que trabajan en distintos idiomas nacionales y la conversión de datos entre algunos de los juegos de caracteres codificados no está soportada, puede habilitar una conversión predeterminada. La conversión de datos predeterminada está descrita en el apartado “Conversión de datos predeterminada” en la página 118.

Archivo ccsid.tbl

El archivo ccsid.tbl se utiliza con las siguientes finalidades:

- En WebSphere MQ para Windows, registra todos los conjuntos de códigos soportados.
- En las plataformas AIX y HP-UX, el sistema operativo mantiene los conjuntos de códigos soportados de forma interna.
- Para todas las demás plataformas UNIX and Linux, las páginas de códigos soportadas se conservan en las tablas de conversión proporcionadas por WebSphere MQ.
- Especifica cualquier conjunto de códigos adicional. Para especificar conjuntos de códigos adicionales, debe editar ccsid.tbl (las instrucciones para su edición se proporcionan en el archivo).
- Especifica cualquier conversión de datos predeterminada.

Puede actualizar la información registrada en ccsid.tbl. Esto puede resultar útil si, por ejemplo, un release posterior del sistema operativo tiene soporte para más juegos de caracteres codificados.

En WebSphere MQ para Windows, ccsid.tbl se encuentra en el directorio `C:\Program Files\IBM\WebSphere MQ\conv\table` de forma predeterminada.

En los sistemas WebSphere MQ para UNIX and Linux, ccsid.tbl se encuentra en el directorio `/var/mqm/conv/table`.

Conversión de datos predeterminada

Si configura canales entre dos máquinas en las que no se da soporte normalmente a la conversión de datos, deberá habilitar la conversión de datos predeterminada para que funcionen los canales.

Para habilitar la conversión de datos predeterminada, edite el archivo ccsid.tbl para especificar un CCSID EBCDIC predeterminado y un CCSID ASCII predeterminado. Las instrucciones correspondientes están incluidas en el archivo. Debe hacer esto en todas las máquinas que se conectarán utilizando los canales. Reinicie el gestor de colas para que el cambio surta efecto.

El proceso de conversión de datos predeterminada es el siguiente:

- Si la conversión entre los CCSID de origen y de destino no está soportada, pero los CCSID de los entornos de origen y de destino son ambos EBCDIC o son ambos ASCII, los datos de tipo carácter se pasan a la aplicación de destino sin realizar ninguna conversión.
- Si un CCSID representa un juego de caracteres codificado ASCII, y el otro representa un juego de caracteres codificado EBCDIC, WebSphere MQ convierte los datos utilizando los CCSID de conversión de datos predeterminados definidos en ccsid.tbl.

Nota: Intente limitar los caracteres que se convierten a aquellos que tengan los mismos valores de código en el juego de caracteres codificado especificado para el mensaje y en el juego de caracteres codificado predeterminado. Si sólo utiliza el juego de caracteres que es válido para los nombres de objeto de WebSphere MQ (tal como se define en Denominación de objetos de IBM WebSphere MQ) podrá cumplir, en general, este requisito. Se producen excepciones con los CCSID EBCDIC 290, 930, 1279 y

5026 utilizados en Japón, ya que los caracteres en minúsculas tienen códigos distintos de los utilizados en otros CCSID EBCDIC.

Conversión de mensajes en formatos definidos por el usuario

El gestor de colas no puede convertir mensajes en formatos definidos por el usuario de un juego de caracteres codificado a otro. Si necesita convertir datos en un formato definido por el usuario, debe facilitar una salida de conversión de datos para cada formato de este tipo. No utilice identificadores CCSID predeterminados para convertir datos de tipo carácter en formatos definidos por el usuario. Si desea más información sobre cómo convertir los datos en formatos definidos por el usuario y sobre cómo escribir las salidas de la conversión de datos, consulte Escritura de salidas de conversión de datos.

Cambio del CCSID del gestor de colas

Cuando haya utilizado el atributo CCSID del mandato ALTER QMGR para cambiar el CCSID del gestor de colas, detenga y reinicie el gestor de colas para asegurarse de que todas las aplicaciones que estén en ejecución, incluyendo el servidor de mandatos y los programas de canal, se detengan y se reinicien.

Esto es necesario porque cualquier aplicación que esté en ejecución cuando se cambia el CCSID del gestor de colas sigue utilizando el CCSID existente.

Administración de IBM WebSphere MQ Telemetry

IBM WebSphere MQ Telemetry se administra mediante IBM WebSphere MQ Explorer u una línea de mandatos. Utilice el explorador para configurar canales de telemetría, controlar el servicio de telemetría y supervisar los clientes MQTT que están conectados a IBM WebSphere MQ. Configure la seguridad de IBM WebSphere MQ Telemetry utilizando JAAS, SSL y el gestor de autorizaciones sobre objetos IBM WebSphere MQ .

Administración utilizando IBM WebSphere MQ Explorer

Utilice el explorador para configurar canales de telemetría, controlar el servicio de telemetría y supervisar los clientes MQTT que están conectados a IBM WebSphere MQ. Configure la seguridad de IBM WebSphere MQ Telemetry utilizando JAAS, SSL y el gestor de autorizaciones sobre objetos IBM WebSphere MQ .

Administración utilizando la línea de mandatos

IBM WebSphere MQ Telemetry se puede administrar completamente en la línea de mandatos mediante los mandatos IBM WebSphere MQ MQSC.

La documentación de IBM WebSphere MQ Telemetry también tiene scripts de ejemplo que muestran el uso básico de la aplicación cliente de MQ Telemetry Transport v3 .

Lea y comprenda los ejemplos en Programas de ejemplo de IBM WebSphere MQ Telemetry en la sección Desarrollo de aplicaciones IBM WebSphere MQ Telemetry antes de utilizarlos.

Conceptos relacionados

WebSphere MQ Telemetry

“Configurar la cola distribuida para enviar mensajes a clientes MQTT” en la página 123
Las aplicaciones de IBM WebSphere MQ pueden enviar mensajes de clientes MQTT v3 publicando en una suscripción que haya creado un cliente o enviando un mensaje directamente. Sea cual sea el método utilizado, el mensaje se coloca en `SYSTEM.MQTT.TRANSMIT.QUEUE`y el servicio de telemetría (MQXR) lo envía al cliente. Hay varias formas de colocar un mensaje en `SYSTEM.MQTT.TRANSMIT.QUEUE`.

“Identificación, autorización y autenticación de clientes MQTT” en la página 126

“Autenticación de canal de telemetría mediante SSL” en la página 133

“Privacidad de las publicaciones en los canales de telemetría” en la página 135

“Configuración SSL de clientes MQTT y canales de telemetría” en la página 136

“Configuración JAAS del canal de telemetría” en la página 141
Configure JAAS para autenticar el valor de `Username` que envía el cliente.

“Daemon de IBM WebSphere MQ Telemetry para conceptos de dispositivos” en la página 143
El daemon de IBM WebSphere MQ Telemetry para dispositivos es una aplicación cliente de MQTT V3 avanzada. Utilícelo para almacenar y reenviar mensajes desde otros clientes MQTT. Se conecta a IBM WebSphere MQ como un cliente MQTT, pero también puede conectarle otros clientes MQTT.

Tareas relacionadas

“Configuración de un gestor de colas para la telemetría en Linux y AIX” en la página 120
Siga estos pasos manuales para configurar un gestor de colas que ejecute IBM WebSphere MQ Telemetry. Puede ejecutar un procedimiento automático para definir una configuración más simple utilizando el soporte de IBM WebSphere MQ Telemetry para IBM WebSphere MQ Explorer.

“Configuración de un gestor de colas para la telemetría en Windows” en la página 122
Siga estos pasos manuales para configurar un gestor de colas que ejecute IBM WebSphere MQ Telemetry. Puede ejecutar un procedimiento automático para definir una configuración más simple utilizando el soporte de IBM WebSphere MQ Telemetry para IBM WebSphere MQ Explorer.

Referencia relacionada

Propiedades de MQXR

Configuración de un gestor de colas para la telemetría en Linux y AIX

Siga estos pasos manuales para configurar un gestor de colas que ejecute IBM WebSphere MQ Telemetry. Puede ejecutar un procedimiento automático para definir una configuración más simple utilizando el soporte de IBM WebSphere MQ Telemetry para IBM WebSphere MQ Explorer.

Antes de empezar

1. Consulte Instalación de IBM WebSphere MQ Telemetry para obtener información sobre cómo instalar IBM WebSphere MQ y la característica IBM WebSphere MQ Telemetry.
2. Cree e inicie un gestor de colas. El gestor de colas se conoce como *qMgr* en esta tarea.
3. Como parte de esta tarea, debe configurar el servicio de telemetría (MQXR). Los valores de propiedad MQXR se almacenan en un archivo de propiedades específico de la plataforma: `mqxr_unix.properties`. Normalmente no necesita editar directamente el archivo de propiedades MQXR, porque casi todos los valores se pueden configurar a través de los mandatos de administración de MQSC o MQ Explorer. Si decide editar directamente el archivo, detenga el gestor de colas antes de realizar los cambios. Consulte Propiedades MQXR.

Acerca de esta tarea

El soporte de IBM WebSphere MQ Telemetry para IBM WebSphere MQ Explorer incluye un asistente y un procedimiento de mandato de ejemplo `sampleMQM`. Los usuarios establecen una configuración inicial utilizando el ID de usuario invitado; consulte Verificación de la instalación de IBM WebSphere MQ Telemetry utilizando IBM WebSphere MQ Explorer y los programas de ejemplo de IBM WebSphere MQ Telemetry.

Siga los pasos de esta tarea para configurar IBM WebSphere MQ Telemetry manualmente utilizando diferentes esquemas de autorización.

Procedimiento

1. Abra una ventana de mandatos en el directorio de ejemplos de telemetría.

   El directorio de ejemplos de telemetría es `/opt/mqm/mqxr/samples`.
2. Cree la cola de transmisión de telemetría.

   ```
   echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc
   qMgr
   ```

   Cuando el servicio de telemetría (MQXR) se inicia por primera vez, se crea `SYSTEM.MQTT.TRANSMIT.QUEUE`.

En esta tarea se crea manualmente, porque SYSTEM.MQTT.TRANSMIT.QUEUE debe existir antes de que se inicie el servicio de telemetría (MQXR) para autorizar el acceso a la misma.

3. Establecer la cola de transmisión predeterminada

   Cuando se inicia por primera vez el servicio de telemetría (MQXR), este no altera el gestor de colas para que SYSTEM.MQTT.TRANSMIT.QUEUE sea cola de transmisión predeterminada.

   Para hacer que SYSTEM.MQTT.TRANSMIT.QUEUE sea la cola de transmisión predeterminada, debe alterar la propiedad de la cola de transmisión predeterminada. Modifique la propiedad utilizando IBM WebSphere MQ Explorer o con el mandato en el ejemplo siguiente:

   ```
   echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
   ```

   Al modificar la cola de transmisión predeterminada, se puede modificar también la configuración existente. La razón por la que se cambia la cola de transmisión predeterminada por SYSTEM.MQTT.TRANSMIT.QUEUE es para poder enviar mensajes directamente a los clientes MQTT de forma más sencilla. Sin modificar la cola de transmisión predeterminada, debe añadir una definición de cola remota para cada cliente que reciba mensajes de IBM WebSphere MQ; consulte "Envío de un mensaje a un cliente directamente" en la página 125.

4. Siga el procedimiento indicado en "Autorización a clientes MQTT a acceder a objetos de WebSphere MQ" en la página 127 para crear uno o varios ID de usuario. Los ID de usuario tienen autorización para publicar, suscribirse y enviar publicaciones a clientes MQTT.

5. Instale el servicio de telemetría (MQXR)

   ```
   cat /opt/<install_dir>/mqxr/samples/installMQXRService_unix.mqsc | runmqsc qMgr
   ```

   Consulte también el código de ejemplo en Figura 19 en la página 122.

6. Inicie el servicio.

   ```
   echo "START SERVICE(SYSTEM.MQXR.SERVICE)" | runmqsc qMgr
   ```

   El servicio de telemetría (MQXR) se inicia automáticamente cuando se inicia el gestor de colas.

   En esta tarea se inicia manualmente, porque el gestor de colas ya se está ejecutando.

7. Utilizando IBM WebSphere MQ Explorer, configure los canales de telemetría para que acepten las conexiones de clientes MQTT.

   Los canales de telemetría deben estar configurados de tal forma que sus identidades sean uno de los ID definidos en el paso 4.

   Consulte también DEFINE CHANNEL (MQTT).

8. Verifique la configuración ejecutando el cliente de ejemplo.

   Para que el cliente de ejemplo funcione con el canal de telemetría, el canal debe autorizar al cliente a publicar, suscribirse y recibir publicaciones. El cliente de ejemplo se conecta al canal de telemetría en el puerto 1883 de forma predeterminada. Consulte también Programas de ejemplo de IBM WebSphere MQ Telemetry.

Ejemplo

La Figura 19 en la página 122 muestra el mandato **runmqsc** para crear el SYSTEM.MQXR.SERVICE manualmente en Linux.

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
    CONTROL(QMGR) +
    DESCR('Manages clients using MQXR protocols such as MQTT') +
    SERVTYPE(SERVER) +
    STARTCMD('+MQ_INSTALL_PATH+/mqxr/bin/runMQXRService.sh') +
    STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+" -g "+MQ_DATA_PATH+"') +
    STOPCMD('+MQ_INSTALL_PATH+/mqxr/bin/endMQXRService.sh') +
    STOPARG('-m +QMNAME+') +
    STDOUT('+MQ_Q_MGR_DATA_PATH+/mqxr.stdout') +
    STDERR('+MQ_Q_MGR_DATA_PATH+/mqxr.stderr')
```

Figura 19. installMQXRService_unix.mqsc

Configuración de un gestor de colas para la telemetría en Windows

Siga estos pasos manuales para configurar un gestor de colas que ejecute IBM WebSphere MQ Telemetry. Puede ejecutar un procedimiento automático para definir una configuración más simple utilizando el soporte de IBM WebSphere MQ Telemetry para IBM WebSphere MQ Explorer.

Antes de empezar

1. Consulte Instalación de IBM WebSphere MQ Telemetry para obtener información sobre cómo instalar IBM WebSphere MQ y la característica IBM WebSphere MQ Telemetry.
2. Cree e inicie un gestor de colas. El gestor de colas se conoce como *qMgr* en esta tarea.
3. Como parte de esta tarea, debe configurar el servicio de telemetría (MQXR). Los valores de propiedad MQXR se almacenan en un archivo de propiedades específico de la plataforma: `mqxr_win.properties`. Normalmente no necesita editar directamente el archivo de propiedades MQXR, porque casi todos los valores se pueden configurar a través de los mandatos de administración de MQSC o MQ Explorer. Si decide editar directamente el archivo, detenga el gestor de colas antes de realizar los cambios. Consulte Propiedades MQXR.

Acerca de esta tarea

El soporte de IBM WebSphere MQ Telemetry para IBM WebSphere MQ Explorer incluye un asistente y un procedimiento de mandato de ejemplo `sampleMQM`. Los usuarios establecen una configuración inicial utilizando el ID de usuario invitado; consulte Verificación de la instalación de IBM WebSphere MQ Telemetry utilizando IBM WebSphere MQ Explorer y los programas de ejemplo de IBM WebSphere MQ Telemetry.

Siga los pasos de esta tarea para configurar IBM WebSphere MQ Telemetry manualmente utilizando diferentes esquemas de autorización.

Procedimiento

1. Abra una ventana de mandatos en el directorio de ejemplos de telemetría.

   El directorio de ejemplos de telemetría es *WMQ program installation directory*`\mqxr\samples`.
2. Cree la cola de transmisión de telemetría.

   ```
   echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
   ```

   Cuando el servicio de telemetría (MQXR) se inicia por primera vez, se crea `SYSTEM.MQTT.TRANSMIT.QUEUE`.

   En esta tarea se crea manualmente, porque `SYSTEM.MQTT.TRANSMIT.QUEUE` debe existir antes de que se inicie el servicio de telemetría (MQXR) para autorizar el acceso a la misma.
3. Establecer la cola de transmisión predeterminada

   ```
   echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
   ```

   Figura 20. Establezca la cola de transmisión predeterminada

Cuando se inicia por primera vez el servicio de telemetría (MQXR), este no altera el gestor de colas para que SYSTEM.MQTT.TRANSMIT.QUEUE sea cola de transmisión predeterminada.

Para hacer que SYSTEM.MQTT.TRANSMIT.QUEUE sea la cola de transmisión predeterminada, debe alterar la propiedad de la cola de transmisión predeterminada. Modifique la propiedad mediante IBM WebSphere MQ Explorer o bien con el mandato que aparece en Figura 20 en la página 122.

Al modificar la cola de transmisión predeterminada, se puede modificar también la configuración existente. La razón por la que se cambia la cola de transmisión predeterminada por SYSTEM.MQTT.TRANSMIT.QUEUE es para poder enviar mensajes directamente a los clientes MQTT de forma más sencilla. Sin modificar la cola de transmisión predeterminada, debe añadir una definición de cola remota para cada cliente que reciba mensajes de IBM WebSphere MQ; consulte "Envío de un mensaje a un cliente directamente" en la página 125.

4. Siga el procedimiento indicado en "Autorización a clientes MQTT a acceder a objetos de WebSphere MQ" en la página 127 para crear uno o varios ID de usuario. Los ID de usuario tienen autorización para publicar, suscribirse y enviar publicaciones a clientes MQTT.
5. Instale el servicio de telemetría (MQXR)

   ```
   type
   installMQXRService_win.mqsc | runmqsc qMgr
   ```

6. Inicie el servicio.

   ```
   echo START SERVICE(SYSTEM.MQXR.SERVICE) | runmqsc qMgr
   ```

   El servicio de telemetría (MQXR) se inicia automáticamente cuando se inicia el gestor de colas.

   En esta tarea se inicia manualmente, porque el gestor de colas ya se está ejecutando.
7. Utilizando IBM WebSphere MQ Explorer, configure los canales de telemetría para que acepten las conexiones de clientes MQTT.

   Los canales de telemetría deben estar configurados de tal forma que sus identidades sean uno de los ID definidos en el paso 4.

   Consulte también DEFINE CHANNEL (MQTT).
8. Verifique la configuración ejecutando el cliente de ejemplo.

   Para que el cliente de ejemplo funcione con el canal de telemetría, el canal debe autorizar al cliente a publicar, suscribirse y recibir publicaciones. El cliente de ejemplo se conecta al canal de telemetría en el puerto 1883 de forma predeterminada. Consulte también Programas de ejemplo de IBM WebSphere MQ Telemetry.

Creación manual de SYSTEM.MQXR.SERVICE

La Figura 21 en la página 123 muestra el mandato **runmqsc** para crear el SYSTEM.MQXR.SERVICE manualmente en Windows.

```
DEF    SERVICE(SYSTEM.MQXR.SERVICE) +
    CONTROL(QMGR) +
    DESCR('Manages clients using MQXR protocols such as MQTT') +
    SERVTYPE(SERVER) +
    STARTCMD('+MQ_INSTALL_PATH+\mqxr\bin\runMQXRService.bat') +
    STARTARG('-m +QMNAME+  -d "+MQ_Q_MGR_DATA_PATH+\." -g "+MQ_DATA_PATH+\."') +
    STOPCMD('+MQ_INSTALL_PATH+\mqxr\bin\endMQXRService.bat') +
    STOPARG('-m +QMNAME+') +
    STDOUT('+MQ_Q_MGR_DATA_PATH+\mqxr.stdout') +
    STDERR('+MQ_Q_MGR_DATA_PATH+\mqxr.stderr')
```

Figura 21. installMQXRService_win.mqsc

Configurar la cola distribuida para enviar mensajes a clientes MQTT

Las aplicaciones de IBM WebSphere MQ pueden enviar mensajes de clientes MQTT v3 publicando en una suscripción que haya creado un cliente o enviando un mensaje directamente. Sea cual sea el método

utilizado, el mensaje se coloca en SYSTEM.MQTT.TRANSMIT.QUEUEy el servicio de telemetría (MQXR) lo envía al cliente. Hay varias formas de colocar un mensaje en SYSTEM.MQTT.TRANSMIT.QUEUE.

Publicación de un mensaje en respuesta a una suscripción de cliente MQTT

El servicio de telemetría (MQXR) crea una suscripción en nombre del cliente MQTT. El cliente es el destino de las publicaciones que coincidan con la suscripción que envíe el cliente. Los servicios de telemetría reenvían las publicaciones coincidentes al cliente.

Un cliente MQTT se conecta a WebSphere MQ como un gestor de colas, con su nombre de gestor de colas establecido en su >ClientIdentifier. El destino de las publicaciones que se envían al cliente es una cola de transmisión, SYSTEM.MQTT.TRANSMIT.QUEUE. El servicio de telemetría reenvía mensajes en SYSTEM.MQTT.TRANSMIT.QUEUE a clientes MQTT, utilizando el nombre del gestor de colas de destino como clave para un cliente específico.

El servicio de telemetría (MQXR) abre la cola de transmisión utilizando IdentificadorCliente como el nombre del gestor de colas. El servicio de telemetría (MQXR) pasa el manejador de objeto de la cola a la llamada MQSUB, para reenviar las publicaciones que coincidan con la suscripción del cliente. En la resolución del nombre de objeto, se crea el ClientIdentifier como el nombre del gestor de colas remoto, y la cola de transmisión debe resolverse en SYSTEM.MQTT.TRANSMIT.QUEUE. Utilizando la resolución de nombres de objeto estándar de WebSphere MQ , *ClientIdentifier* se resuelve de la forma siguiente; consulte Tabla 6 en la página 124.

1. *ClientIdentifier* no coincide con nada.

   ClientIdentifier es un nombre de gestor de colas remoto. No coincide con el nombre del gestor de colas local, con un alias de gestor de colas, ni con un nombre de cola de transmisión.
   El nombre de cola no se ha definido. Actualmente, el servicio de telemetría (MQXR) establece SYSTEM.MQTT.PUBLICATION.QUEUE como el nombre de la cola. Un cliente MQTT v3 no da soporte a las colas, por lo que el cliente ignora el nombre de cola que se ha resuelto.
   La propiedad del gestor de colas local, Cola de transmisión predeterminada, debe establecerse en SYSTEM.MQTT.TRANSMIT.QUEUE, de modo que la publicación se coloque en SYSTEM.MQTT.TRANSMIT.QUEUE para enviarse al cliente.
2. *ClientIdentifier* coincide con un alias de gestor de colas denominado *ClientIdentifier*.

   ClientIdentifier es un nombre de gestor de colas remoto. Coincide con el nombre de un alias de gestor de colas.
   El alias del gestor de colas debe definirse con *ClientIdentifier* como el nombre del gestor de colas remoto.
   Al establecer el nombre de cola de transmisión en la definición de alias de gestor de colas, no es necesario que la transmisión predeterminada se establezca en SYSTEM.MQTT.TRANSMIT.QUEUE.

Tabla 6. Resolución de nombres de un alias de gestor de colas MQTT

	Entrada		**Salida**		
ClientIdentifier	**Nombre del gestor de colas**	**Nombre de cola**	**Nombre del gestor de colas**	**Nombre de cola**	**Cola de transmisión**
No coincide con nada	*ClientIdentifier*	*sin definir*	*ClientIdentifier*	*sin definir*	Cola de transmisión predeterminada. SYSTEM.MQTT.TRANSMIT.QUEUE

Tabla 6. Resolución de nombres de un alias de gestor de colas MQTT (continuación)					
	Entrada		**Salida**		
ClientIdentifier	**Nombre del gestor de colas**	**Nombre de cola**	**Nombre del gestor de colas**	**Nombre de cola**	**Cola de transmisión**
Coincide con un alias de gestor de colas denominado *ClientIdentifier*	*ClientIdentifier*	*sin definir*	*ClientIdentifier*	*sin definir*	SYSTEM.MQTT.TRANSMIT.QUEUE

Para obtener más información sobre la resolución de nombres, consulte Resolución de nombres.

Cualquier programa de WebSphere MQ puede publicar en el mismo tema. La publicación se envía a sus suscriptores, incluidos aquellos clientes MQTT v3 que tengan una suscripción al tema.

Si se crea un tema administrativo en un clúster, con el atributo CLUSTER(*clusterName*), cualquier aplicación del clúster puede publicar en el cliente; por ejemplo:

```
echo DEFINE TOPIC('MQTTExamples') TOPICSTR('MQTT Examples') CLUSTER(MQTT) REPLACE | runmqsc qMgr
```

Figura 22. Definición de un tema de clúster en Windows

Nota: No asigne a SYSTEM.MQTT.TRANSMIT.QUEUE un atributo de clúster.

Los suscriptores y publicadores de cliente MQTT pueden conectarse a gestores de colas distintos. Los suscriptores y publicadores pueden formar parte del mismo clúster, o pueden conectarse mediante una jerarquía de publicación/suscripción. La publicación se entrega del publicador al suscriptor, mediante WebSphere MQ.

Envío de un mensaje a un cliente directamente

Una alternativa a que un cliente cree una suscripción y reciba una publicación que coincida con la suscripción de tema, es enviar un mensaje a un cliente MQTT v3 directamente. Las aplicaciones de cliente MQTT V3 no pueden enviar mensajes directamente, pero otras aplicaciones como, por ejemplo, las aplicaciones de WebSphere MQ, sí pueden.

La aplicación de WebSphere MQ debe conocer el ClientIdentifier del cliente MQTT v3. Como los clientes MQTT v3 no tienen colas, el nombre de cola de destino se pasa al método v3 cliente de aplicaciones messageArrived de MQTT como nombre de tema. Por ejemplo, en un programa MQI, cree un descriptor de objetos con el cliente como ObjectQmgrName:

```
MQOD.ObjectQmgrName = ClientIdentifier;
MQOD.ObjectName = name;
```

Figura 23. El descriptor de objetos MQI envía un mensaje a un destino de cliente MQTT v3

Si la aplicación se escribe utilizando JMS, cree un destino de punto a punto; por ejemplo:

```
javax.jms.Destination jmsDestination =
                      (javax.jms.Destination)jmsFactory.createQueue
                      ("queue://ClientIdentifier/name");
```

Figura 24. Un destino JSM envía un mensaje a un cliente MQTT v3

Para enviar un mensaje no solicitado a un cliente MQTT, utilice una definición de cola remota. El nombre del gestor de colas remoto debe resolverse en el `ClientIdentifier` del cliente. La cola de transmisión debe resolverse en `SYSTEM.MQTT.TRANSMIT.QUEUE`; consulte Tabla 7 en la página 126. El nombre de la cola remota puede ser cualquier cosa. El cliente lo recibe como una serie de tema.

Tabla 7. Resolución de nombres una definición de cola remota de cliente MQTT

Entrada		**Salida**		
Nombre de cola	**Nombre del gestor de colas**	**Nombre de cola**	**Nombre del gestor de colas**	**Cola de transmisión**
Nombre de la definición de la cola remota	En blanco o nombre de gestor de colas local	Nombre de cola remota utilizado como una serie de tema	*ClientIdentifier*	`SYSTEM.MQTT.TRANSMIT.QUEUE`

Si el cliente está conectado, el mensaje se envía directamente al cliente MQTT, que llama al método `messageArrived`; consulte el método `messageArrived`.

Si el cliente se ha desconectado con una sesión persistente, el mensaje se almacena en `SYSTEM.MQTT.TRANSMIT.QUEUE`; consulte Sesiones sin estado y con estado MQTT . Se reenvía al cliente cuando éste se vuelve a conectar a la sesión.

Si envía un mensaje no persistente, éste se envía al cliente con una calidad de servicio de "como máximo una vez", `QoS=0`. Si envía un mensaje persistente directamente a un cliente, de forma predeterminada, se envía con la calidad de servicio "exactamente una vez", `QoS=2`. Como es posible que el cliente no tenga un mecanismo de persistencia, el cliente puede reducir la calidad de servicio que acepta para los mensajes enviados directamente. Para bajar la calidad de servicio para los mensajes enviados directamente a un cliente, suscríbase al tema `DEFAULT.QoS`. Especifique la calidad de servicio máxima a la que el cliente puede dar soporte.

Identificación, autorización y autenticación de clientes MQTT

El servicio de telemetría (MQXR) publica temas de WebSphere MQ, o se suscribe a ellos, en nombre de los clientes MQTT, a través de los canales MQTT. El administrador de WebSphere MQ configura la identidad de canal MQTT que se utiliza para la autorización de WebSphere MQ. El administrador debe definir una identidad común para el canal o utilizar el `Username` o `ClientIdentifier` de un cliente conectado al canal.

El servicio de telemetría (MQXR) puede autenticar el cliente utilizando el valor `Username` que proporciona el cliente o bien utilizando un certificado de cliente. El valor de `Username` se autentica utilizando una contraseña que proporciona el cliente.

En resumen: la identificación de cliente es la selección de la identidad de cliente. En función del contexto, el cliente se identifica con el valor `ClientIdentifier`, el valor `Username`, una identidad común de cliente que crea el administrador o un certificado de cliente. El identificador de cliente utilizado para la comprobación de la autenticidad no tiene que ser el mismo identificador que se utiliza para la autorización.

Los programas de cliente MQTT establecen el valor de `Username` y de `Password` que se envían al servidor utilizando un canal MQTT. También pueden establecer las propiedades SSL que se necesitan para cifrar y autenticar la conexión. El administrador decide si autenticar el canal MQTT y cómo autenticarlo.

Para autorizar a un cliente MQTT acceder a los objetos de WebSphere MQ, autorice al valor `ClientIdentifier`, o al valor `Username` del cliente, o autorice una identidad común del cliente. Para permitir que un cliente se conecte a WebSphere MQ, autentique el valor de `Username`, o utilice un certificado de cliente. Configurar JAAS para autenticar el valor de `Username`, y configure SSL para autenticar un certificado de cliente.

Si define un valor `Password` en el cliente, cifre la conexión utilizando VPN, o configure el canal MQTT para que utilice SSL y mantenga la contraseña privada.

Resulta difícil gestionar certificados de clientes. Por este motivo, si los riesgos que conlleva la autenticación por contraseña resultan aceptables, ésta se utiliza a menudo para autenticar a los clientes.

Si existe una manera segura de gestionar y almacenar el certificado de cliente, se puede confiar en la autenticación con certificados. Sin embargo los certificados no suelen gestionarse de forma segura en los entornos en los que se utiliza la telemetría. En su lugar la autenticación de dispositivos que utilizan certificados de cliente se complementa con la autenticación de las contraseñas de cliente en el servidor. Debido a la complejidad adicional, la utilización de certificados de cliente se limita a aplicaciones altamente confidenciales. El uso de formas de autenticación se conoce como autenticación por dos factores. Debe conocer uno de los factores, por ejemplo, una contraseña, y tener otro, por ejemplo, un certificado.

En una aplicación en la que se necesite mucha confidencialidad como, por ejemplo, los dispositivos que utilicen la tecnología "chip and pin", el dispositivo se bloquea durante la fabricación para evitar intrusiones no autorizadas en el hardware y software internos. En el dispositivo se copia un certificado de cliente de confianza y duración limitada. El dispositivo se despliega en la ubicación en la que va a utilizarse. Cada vez que se utiliza el dispositivo, se realiza más autenticación, usando una contraseña u otro certificado de una tarjeta inteligente.

Identidad y autorización de cliente MQTT

Utilice el valor `ClientIdentifier`, `Username`, o una identidad de cliente común para autorizar a acceder a objetos de WebSphere MQ.

El administrador de WebSphere MQ tiene tres opciones para seleccionar la identidad del canal MQTT. El administrador elige cuándo definir o modificar el canal MQTT que utiliza el cliente. La identidad se utiliza para autorizar el acceso a temas de WebSphere MQ. Las opciones son las siguientes:

1. El identificador de cliente.
2. Una identidad que el administrador proporciona al canal.
3. El valor de `Username` que se ha pasado del cliente MQTT.

`claseUsername` es un atributo de la clase MqttConnectOptions. Debe establecerse antes de que el cliente se conecte al servicio. Su valor predeterminado es null.

Utilice el mandato **`setmqaut`** de WebSphere MQ para seleccionar qué objetos y qué acciones, están autorizados para que los utilice la identidad asociada al canal MQTT. Por ejemplo, para autorizar a una identidad de canal, `MQTTClient`, que proporciona el administrador del gestor de colas, `QM1`:

```
setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

Autorización a clientes MQTT a acceder a objetos de WebSphere MQ

Siga estos pasos para autorizar a los clientes MQTT a publicar y suscribirse a objetos de WebSphere MQ. Los pasos siguen cuatro patrones de control de acceso alternativos.

Antes de empezar

Los clientes MQTT están autorizados a acceder a objetos en WebSphere MQ cuando se les asigna una identidad al conectarse a un canal de telemetría. El administrador de WebSphere MQ configura el canal de telemetría utilizando WebSphere MQ Explorer para dar a un cliente uno de los tres tipos de identidad siguientes:

1. `ClientIdentifier`
2. `Nombre de usuario`
3. Un nombre que el administrador asigna al canal.

Sea cual sea el tipo que utilice, la identidad debe estar definida en WebSphere MQ como principal por el servicio de autorización instalado. El servicio de autorización predeterminado en Windows o Linux se denomina Gestor de autorizaciones sobre objetos (OAM). Si va a utilizar el OAM, la identidad debe definirse como un ID de usuario.

Utilice la identidad para dar a un cliente, o grupo de clientes, permiso para publicar o suscribirse en temas definidos por WebSphere MQ. Si un cliente MQTT se ha suscrito a un tema, utilice la identidad para darle permiso para recibir las publicaciones que resultantes.

Resulta difícil gestionar un sistema con decenas de miles de clientes MQTT, ya que cada uno necesita permiso de acceso individual. Una solución es definir identidades comunes y asociar clientes MQTT individuales a identidades comunes. Defina tantas identidades comunes como sea necesario para definir diferentes combinaciones de permisos. Otra solución es escribir un servicio de autorización propio que maneje los miles de usuarios de forma más sencilla a como lo hace el sistema operativo.

Puede combinar clientes MQTT en identidades comunes de dos formas, mediante el OAM:

1. Defina varios canales de telemetría, cada uno con un ID de usuario diferente que el administrador asigna mediante WebSphere MQ Explorer. Los clientes que se conectan utilizando diferentes números de puertos TCP/IP se asocian a diferentes canales de telemetría y se les asignan diferentes identidades.
2. Defina un único canal de telemetría, pero haga que cada cliente seleccione un valor de `Username` entre un conjunto pequeño de ID de usuario. El administrador configura el canal de telemetría para seleccionar el valor de `Username` del cliente como su identidad.

En esta tarea, la identidad del canal de telemetría se denomina *`mqttUser`*, independientemente de cómo se establezca. Si las colecciones de clientes utilizan identidades diferentes, utilice varios *`mqttUsers`*, uno para cada colección de clientes. Puesto que la tarea utiliza OAM, cada *`mqttUser`* debe ser un ID de usuario.

Acerca de esta tarea

En esta tarea puede elegir entre cuatro patrones de control de acceso, que puede adaptar a requisitos específicos. Los patrones se diferencian en su granularidad del control de acceso.

- “Sin control de acceso” en la página 128
- “Control de acceso de granularidad gruesa” en la página 128
- “Control de acceso de granularidad media” en la página 129
- “Control de acceso de granularidad precisa” en la página 129

El resultado de los modelos es asignar conjuntos de permisos de *`mqttUsers`* para publicar y suscribirse a WebSphere MQ y recibir publicaciones de WebSphere MQ.

Sin control de acceso
A los clientes MQTT se les proporciona autorización administrativa de WebSphere MQ, y pueden realizar cualquier acción en cualquier objeto.

Procedimiento

1. Cree un ID de usuario *`mqttUser`* para que actúe como la identidad de todos los clientes MQTT.
2. Añada *`mqttUser`* al grupo `mqm` ; consulte Adición de un usuario a un grupo en Windowso Adición de un usuario a un grupo en Linux

Control de acceso de granularidad gruesa
Los clientes MQTT tienen autorización para publicar y suscribirse, y enviar mensajes a clientes MQTT. No tienen autorización para realizar otras acciones, ni para acceder a otros objetos.

Procedimiento

1. Cree un ID de usuario *mqttUser* para que actúe como la identidad de todos los clientes MQTT.
2. Autorice a *mqttUser* a publicar y suscribirse a todos los temas, y a enviar publicaciones a clientes MQTT.

```
setmqaut -m qMgr -t topic -n SYSTEM.BASE.TOPIC -p mqttUser -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqttUser -all +put
```

Control de acceso de granularidad media
Los clientes MQTT se dividen en diferentes grupos para publicar y suscribirse a diferentes conjuntos de temas, y para enviar mensajes a clientes MQTT.

Procedimiento

1. Cree varios ID de usuario *mqttUsers*, y varios temas administrativos en el árbol de temas de publicación/suscripción.
2. Autorice a diferentes *mqttUsers* para temas diferentes.

```
setmqaut -m qMgr -t topic -n topic1 -p mqttUserA -all +pub +sub
setmqaut -m qMgr -t topic -n topic2 -p mqttUserB -all +pub +sub
```

3. Cree un grupo *mqtt*, y añada todos los *mqttUsers* al grupo.
4. Autorice a *mqtt* a enviar temas a clientes MQTT.

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Control de acceso de granularidad precisa
Los clientes MQTT se incorporan en un sistema de control de acceso existente que autoriza a los grupos a realizar acciones sobre los objetos.

Acerca de esta tarea

Se asigna un ID de usuario a uno o más grupos de sistema operativo en función de las autorizaciones que necesite. Si las aplicaciones de WebSphere MQ publican o se suscriben al mismo espacio de temas que los clientes MQTT, utilice este modelo. Se hace referencia a los grupos como Publish*X*, Subscribe*Y*y mqtt.

Publish*X*
Los miembros de los grupos Publish*X* pueden publicar en *topicX*.

Subscribe*Y*
Los miembros de grupos de Subscribe*Y* pueden suscribirse a *topicY*.

mqtt
Los miembros del grupo *mqtt* puede enviar publicaciones a clientes MQTT.

Procedimiento

1. Cree varios grupos, Publish*X* y Subscribe*Y* a los que se les asignen varios temas administrativos en el árbol de temas de publicación/suscripción.
2. Cree un grupo mqtt.
3. Cree varios ID de usuario *mqttUsers* y agregue los usuarios a cualquiera de los grupos, dependiendo qué estén autorizados a hacer.
4. Autorice a diferentes grupos Publish*X* y Subscribe*X* a diferentes temas y autorice al grupo *mqtt* a enviar mensajes a clientes MQTT.

```
setmqaut -m qMgr -t topic -n topic1 -p PublishX -all +pub
setmqaut -m qMgr -t topic -n topic1 -p SubscribeX -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Autenticación de cliente MQTT mediante contraseña

Autentique el valor de `Username` utilizando la contraseña del cliente. Puede autenticar el cliente utilizando una identidad diferente a la utilizada para autorizar al cliente a publicar temas y a suscribirse a los mismos.

El servicio de telemetría (MQXR) utiliza JAAS para autenticar el valor `Username` del cliente. JAAS utiliza el valor `Password` que proporciona el cliente MQTT.

El administrador de WebSphere MQ decide si se debe autenticar el valor de `Username`, o no autenticar nada, configurando el canal MQTT al que se conecta un cliente. Los clientes pueden asignarse a diferentes canales y cada canal puede configurarse para que autentique sus clientes de formas diferentes. Mediante JAAS, puede configurar qué métodos deben autenticar el cliente y cuáles pueden hacerlo de forma opcional.

La opción que elija para autenticar la identidad no afecta a la opción que elija para autorizar la identidad. Es posible que desee configurar una identidad común para la autorización para facilitar las tareas administrativas, pero que se autentique cada usuario para que utilice dicha identidad. En el procedimiento siguiente se describen los pasos para autenticar los usuarios individuales para que utilice una identidad común:

1. El administrador de WebSphere MQ establece la identidad del canal MQTT en un nombre cualquiera como, por ejemplo, `MQTTClientUser`, mediante WebSphere MQ Explorer.
2. El administrador de WebSphere MQ autoriza a `MQTTClient` a publicar y a suscribirse a cualquier tema:

   ```
   setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
   setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
   ```

3. El desarrollador de aplicaciones cliente MQTT crea un objeto `MqttConnectOptions` y establece `Username` y `Password` antes de conectarse al servidor.
4. El desarrollador de seguridad crea un `LoginModule` JAAS para autenticar el valor `Username` con el valor `Password` y lo incluye en el archivo de configuración JAAS.
5. El administrador de WebSphere MQ configura el canal MQTT para autenticar el valor `UserName` del cliente utilizando JAAS.

Autenticación de cliente MQTT mediante SSL

Las conexiones entre el cliente MQTT y el gestor de colas las inicia siempre el cliente MQTT. El cliente MQTT es siempre el cliente SSL. Tanto la autenticación de cliente del servidor como la autenticación de servidor del cliente MQTT son opcionales.

Al proporcionar al cliente un certificado digital firmado privado, puede autenticar el cliente MQTT en IBM WebSphere MQ. El administrador de IBM WebSphere MQ MQ puede obligar a los clientes MQTT a autenticarse ellos mismos en el gestor de colas mediante SSL. Sólo puede solicitar la autenticación de cliente como parte de una autenticación mutua.

Algunos tipos de redes privadas virtuales (VPN) como, por ejemplo, IPsec, autentican los puntos finales de una conexión TCP/IP, como alternativa a utilizar SSL. VPN cifra cada paquete IP que se transmite por la red. Una vez establecida la conexión VPN, se ha establecido la red de confianza. Puede conectar clientes MQTT a los canales de telemetría utilizando TCP/IP en la red VPN.

La autenticación de cliente utilizando SSL depende de que el cliente tenga un dato secreto. Este dato secreto es la clave privada del cliente en el caso de un certificado autofirmado o una clave que proporciona una entidad emisora de certificados. La clave se utiliza para firmar el certificado digital del cliente. Todas las personas que tengan la correspondiente clave pública pueden verificar el certificado digital. Los certificados pueden ser de confianza o, si están en cadena, se les puede realizar un seguimiento a través de una cadena de certificados a un certificado raíz de confianza. La verificación de clientes envía todos los certificados de la cadena de certificados que proporciona el cliente al servidor. El servidor comprueba la cadena de certificados hasta que encuentra un certificado en el que confía. El certificado de confianza puede ser un certificado público generado a partir de un certificado

autofirmado, o un certificado raíz que normalmente ha emitido una entidad emisora de certificados. Un último paso, que es opcional, es la comparación del certificado de confianza con una lista de revocación de certificados "activa".

El certificado de confianza puede haberlo emitido una entidad emisora de certificados y puede estar incluido en el almacén de certificados JRE. Puede ser un certificado autofirmado o cualquiera que se haya añadido al almacén de claves del canal de telemetría como un certificado de confianza.

Nota: El canal de telemetría tiene un almacén de claves/almacén de confianza combinado que contiene tanto las claves privadas de uno o varios canales de telemetría, y todos los certificados públicos necesarios para autenticar clientes. Puesto que un canal SSL debe tener un almacén de claves, y es el mismo archivo que el almacén de confianza, nunca se hace referencia al almacén de certificados JRE. La implicación es que si la autenticación de un cliente requiere un certificado raíz de CA, debe colocar el certificado raíz en el almacén de claves del canal, aunque el certificado raíz de CA ya esté en el almacén de certificados JRE. Nunca se hace referencia alguna al almacén de certificados JRE.

Considere las amenazas a las que la autenticación del cliente pretende hacer frente y qué papel juega el cliente y el servidor en esa situación. La autenticación del certificado de cliente sola no es suficiente para evitar accesos no autorizados al sistema. Si otra persona ha tomado posesión del dispositivo del cliente, éste no tiene necesariamente que estar actuando con la autoridad del titular del certificado. No confíe nunca en una única defensa frente a ataques no deseados. Utilice al menos dos factores para la autenticación, además de la posesión complementaria de un certificado con conocimiento de información privada. Por ejemplo, utilice JAAS y autentique el cliente utilizando una contraseña que emita el servidor.

La principal amenaza a la que se enfrenta un certificado de cliente es que caiga en las manos equivocadas. El certificado se encuentra en un almacén de claves protegido con contraseña en el cliente. ¿Cómo se coloca en el almacén de claves? ¿Cómo consigue el cliente MQTT la contraseña para el almacén de claves? ¿Qué nivel de seguridad tiene la protección con contraseña? Los dispositivos de telemetría a menudo son fáciles de eliminar y pueden ser pirateados en privado. ¿Debe el hardware del dispositivo estar protegido contra manipulación no autorizada? La distribución y protección de certificados del lado del cliente se reconoce como difícil; se denomina el problema de la gestión de claves.

Una amenaza secundaria es que el dispositivo se utilice de forma incorrecta para acceder a servidores de forma imprevista. Por ejemplo, si la aplicación MQTT está amenazada, es posible utilizar algún punto débil de la configuración del servidor mediante la identidad de cliente autenticada.

Para autenticar un cliente MQTT mediante SSL, configure el canal de telemetría y el cliente.

-
-

Configuración de canal de telemetría para la autenticación de cliente MQTT mediante SSL

El administrador de IBM WebSphere MQ configura los canales de telemetría en el servidor. Cada canal se configura para aceptar una conexión TCP/IP en un número de puerto diferente. Los canales SSL se configuran con un acceso protegido con frase de contraseña a los archivos de claves. Si un canal SSL se define sin frase de contraseña o archivo de claves, el canal no acepta conexiones SSL.

Establezca la propiedad `com.ibm.mq.MQTT.ClientAuth` de un canal de telemetría SSL en `REQUIRED` para obligar a todos los clientes que se conecten en dicho canal a que proporcionen pruebas de que han verificado los certificados digitales. Los certificados de cliente se autentican utilizando certificados de las entidades emisoras de certificados, dirigidos a un certificado raíz de confianza. Si el certificado de cliente es autofirmado o está firmado por un certificado que proviene de una entidad emisora de certificados, los certificados firmados públicamente del cliente, o autoridad de certificados, deben almacenarse de forma segura en el servidor.

Coloque el certificado de cliente firmado públicamente o el certificado de la entidad emisora de certificados en el almacén de claves del canal de telemetría. En el servidor, los certificados firmados públicamente se almacenan en el mismo archivo de claves que los certificados firmados en privado, en vez de en un almacén de confianza aparte.

El servidor verifica la firma de los certificados de cliente que se envía utilizando todos los certificados públicos y paquetes de cifrado que tenga. El servidor verifica la cadena de claves. El gestor de colas puede configurarse para probar el certificado respecto a la lista de revocación de certificados. La propiedad de lista de nombres de revocación de gestor de colas es SSLCRLNL.

Si alguno de los certificados que un cliente envía lo verifica un certificado del almacén de claves del servidor, el cliente se autentica.

El administrador de WebSphere MQ puede configurar el mismo canal de telemetría para utilizar JAAS en la comprobación del valor Username o ClientIdentifier del cliente con el valor Password del cliente.

Puede utilizar el mismo almacén de claves para varios canales de telemetría.

La verificación de al menos un certificado digital en el almacén de claves de cliente protegido con contraseña en el dispositivo autentica al cliente en el servidor. El certificado digital sólo se utiliza para la autenticación mediante WebSphere MQ. No se utiliza para verificar la dirección TCP/IP del cliente, o establecer la identidad del cliente para la autorización o la contabilidad. La identidad del cliente adoptado por el servidor es Username o ClientIdentifier del cliente, o una identidad creada por el administrador de WebSphere MQ.

También puede utilizar las suites de cifrado SSL para la autenticación de cliente.A continuación se muestra una lista alfabética de las suites de cifrado SSL que se admiten actualmente:

- SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
- SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- SSL_DH_anon_WITH_AES_128_CBC_SHA
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_DSS_WITH_AES_128_CBC_SHA
- SSL_DHE_DSS_WITH_DES_CBC_SHA
- SSL_DHE_DSS_WITH_RC4_128_SHA
- SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA

- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V7.5.0.2 Si piensa utilizar suites de cifrado de SHA-2 , consulte Requisitos del sistema para utilizar suites de cifrado de SHA-2 con canales MQTT.

Conceptos relacionados

"Configuración del canal de telemetría para la autenticación de canal mediante SSL" en la página 134
El administrador de IBM WebSphere MQ configura los canales de telemetría en el servidor. Cada canal se configura para aceptar una conexión TCP/IP en un número de puerto diferente. Los canales SSL se configuran con un acceso protegido con frase de contraseña a los archivos de claves. Si un canal SSL se define sin frase de contraseña o archivo de claves, el canal no acepta conexiones SSL.

CipherSpecs y CipherSuites

Referencia relacionada

DEFINE CHANNEL (MQTT)
ALTER CHANNEL (MQTT)

Autenticación de canal de telemetría mediante SSL

Las conexiones entre el cliente MQTT y el gestor de colas las inicia siempre el cliente MQTT. El cliente MQTT es siempre el cliente SSL. Tanto la autenticación de cliente del servidor como la autenticación de servidor del cliente MQTT son opcionales.

El cliente intenta siempre autenticar el servidor, a menos que esté configurado para utilizar una CipherSpec que dé soporte a la conexión anónima. Si la autenticación falla, la conexión no se establece.

Algunos tipos de redes privadas virtuales (VPN) como, por ejemplo, IPsec, autentican los puntos finales de una conexión TCP/IP, como alternativa a utilizar SSL. VPN cifra cada paquete IP que se transmite por la red. Una vez establecida la conexión VPN, se ha establecido la red de confianza. Puede conectar clientes MQTT a los canales de telemetría utilizando TCP/IP en la red VPN.

La autenticación de servidor mediante SSL autentica el servidor al que va a enviar información confidencial. El cliente realiza las comprobaciones que coinciden con los certificados enviados desde el servidor, con los certificados colocados en su almacén de confianza o en su almacén `cacerts` de JRE.

El almacén de certificados JRE es un archivo JKS, `cacerts`. Se encuentra en JRE `InstallPath\lib\security\`. Se instala con la contraseña predeterminada `changeit`. Puede almacenar los certificados en que confíe en el almacén de certificados de JRE, o en el almacén de confianza del cliente. No puede utilizar ambos almacenes. Utilice el almacén de confianza del cliente si desea mantener los certificados públicos en los que el cliente confía separados de los certificados que utilicen otras aplicaciones Java. Utilice el almacén de certificados JRE si desea utilizar un almacén de certificados común para todas las aplicaciones Java que ejecuten en el cliente. Si decide utilizar el

almacén de certificados de JRE, revise los certificados que contenga, para asegurarse de que confía en ellos.

Puede modificar la configuración JSSE indicando un proveedor de confianza diferente. Puede personalizar un proveedor de confianza para que realice diferentes comprobaciones en un certificado. En algunos entornos OGSi que han utilizado el cliente MQTT, el entorno proporciona un proveedor de confianza diferente.

Para autenticar el canal de telemetría utilizando SSL, configure el servidor y el cliente.

•
•

Configuración del canal de telemetría para la autenticación de canal mediante SSL

El administrador de IBM WebSphere MQ configura los canales de telemetría en el servidor. Cada canal se configura para aceptar una conexión TCP/IP en un número de puerto diferente. Los canales SSL se configuran con un acceso protegido con frase de contraseña a los archivos de claves. Si un canal SSL se define sin frase de contraseña o archivo de claves, el canal no acepta conexiones SSL.

Almacene el certificado digital en el servidor, firmado con su clave privada, en el almacén de claves que el canal de telemetría vaya a utilizar en el servidor. Almacene cualquier certificado de su cadena de claves en el almacén de claves, si desea transmitir la cadena de claves al cliente. Configure el canal de telemetría mediante WebSphere MQ Explorer para utilizar SSL. Proporcione la vía de acceso al almacén de claves y la frase de contraseña para acceder al mismo. Si no establece el número de puerto TCP/IP del canal, el número de puerto del canal de telemetría SSL toma, como valor predeterminado, el 8883.

También puede utilizar las suites de cifrado SSL para la autenticación de canal.A continuación se muestra una lista alfabética de las suites de cifrado SSL que se admiten actualmente:

- SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
- SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
- SSL_DH_anon_WITH_AES_128_CBC_SHA
- SSL_DH_anon_WITH_DES_CBC_SHA
- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_DSS_WITH_AES_128_CBC_SHA
- SSL_DHE_DSS_WITH_DES_CBC_SHA
- SSL_DHE_DSS_WITH_RC4_128_SHA
- SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5

- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V 7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V 7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V 7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V 7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V 7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V 7.5.0.2 Si piensa utilizar suites de cifrado de SHA-2 , consulte Requisitos del sistema para utilizar suites de cifrado de SHA-2 con canales MQTT.

Conceptos relacionados

“Configuración de canal de telemetría para la autenticación de cliente MQTT mediante SSL” en la página 131

El administrador de IBM WebSphere MQ configura los canales de telemetría en el servidor. Cada canal se configura para aceptar una conexión TCP/IP en un número de puerto diferente. Los canales SSL se configuran con un acceso protegido con frase de contraseña a los archivos de claves. Si un canal SSL se define sin frase de contraseña o archivo de claves, el canal no acepta conexiones SSL.

CipherSpecs y CipherSuites

Referencia relacionada

DEFINE CHANNEL (MQTT)

ALTER CHANNEL (MQTT)

Privacidad de las publicaciones en los canales de telemetría

La privacidad de las publicaciones MQTT enviadas en cualquier dirección mediante los canales de telemetría está protegida mediante SSL, para cifrar las transmisiones a través de la conexión.

Los clientes MQTT que se conecten a los canales de telemetría utilizan SSL para proteger la privacidad de las publicaciones transmitidas en el canal, mediante el cifrado de claves simétricas. Puesto que los puntos finales no se autentican, no se puede confiar en un canal de cifrado solo. Combine la protección de privacidad con la autenticación del servidor o mutua.

Algunos tipos de redes privadas virtuales (VPN) como, por ejemplo, IPsec, autentican los puntos finales de una conexión TCP/IP, como alternativa a utilizar SSL. VPN cifra cada paquete IP que se transmite por la red. Una vez establecida la conexión VPN, se ha establecido la red de confianza. Puede conectar clientes MQTT a los canales de telemetría utilizando TCP/IP en la red VPN.

Para obtener información sobre una configuración típica, que cifre el canal y autentique el servidor, consulte “Autenticación de canal de telemetría mediante SSL” en la página 133.

El cifrado de las conexiones SSL sin autenticar el servidor expone la conexión frente a ataques de terceros. Aunque la información que se intercambia está protegida contra escuchas no autorizadas, no sabe con quién se está intercambiando. A menos que controle la red, estará expuesto a que alguien intercepte las transmisiones IP, haciéndose pasar por el punto final.

Puede crear una conexión SSL cifrada, sin autenticar el servidor, mediante un intercambio de claves Diffie-Hellman CipherSpec que dé soporte a SSL anónima. El secreto maestro, compartido entre el cliente y el servidor, y que se utiliza para cifrar transmisiones SSL, se establece sin intercambiar ningún certificado de servidor firmado en privado.

Puesto que las conexiones anónimas son inseguras, la mayoría de las implementaciones SSL no toman como valor predeterminado la utilización de las CipherSpecs anónimas. Si un canal de telemetría acepta una conexión SSL que solicita un cliente, el canal deberá tener un almacén de claves protegido mediante una frase de contraseña. De forma predeterminada, puesto que las implementaciones SSL no utilizan las CipherSpecs anónimas, el almacén de claves debe contener un certificado de firma privada que demuestre que el cliente puede autenticarse.

Si utiliza las CipherSpecs anónima, el almacén de claves del servidor debe existir, pero no es necesario que contenga ningún certificado firmado en privado.

Otra forma de establecer una conexión cifrada es sustituir el proveedor de confianza en el cliente por su propia implementación. El proveedor de confianza no ha podría autenticar el certificado de servidor, pero la conexión se cifraría.

Configuración SSL de clientes MQTT y canales de telemetría

Los clientes MQTT y el servicio WebSphere MQ Telemetry (MQXR) utilizan JSSE (Java Secure Socket Extension) para conectar canales de telemetría utilizando SSL. El daemon de IBM WebSphere MQ Telemetry para dispositivos no admite SSL.

Configure SSL para autenticar el canal de telemetría, el cliente MQTT y cifrar la transferencia de mensajes entre los clientes y el canal de telemetría.

Algunos tipos de redes privadas virtuales (VPN) como, por ejemplo, IPsec, autentican los puntos finales de una conexión TCP/IP, como alternativa a utilizar SSL. VPN cifra cada paquete IP que se transmite por la red. Una vez establecida la conexión VPN, se ha establecido la red de confianza. Puede conectar clientes MQTT a los canales de telemetría utilizando TCP/IP en la red VPN.

Puede configurar la conexión entre un cliente MQTT de Java y un canal de telemetría para utilizar el protocolo SSL sobre TCP/IP. Lo que se asegura depende de cómo configure SSL para que utilice JSSE. Empezando con la configuración más segura, puede configurar tres niveles de seguridad diferentes:

1. Permita que sólo se conecten los clientes MQTT de confianza. Conecte un cliente MQTT sólo a un canal de telemetría de confianza. Cifre los mensajes entre el cliente y el gestor de colas; consulte “Autenticación de cliente MQTT mediante SSL” en la página 130.
2. Conecte un cliente MQTT sólo a un canal de telemetría de confianza. Cifre los mensajes entre el cliente y el gestor de colas; consulte “Autenticación de canal de telemetría mediante SSL” en la página 133.
3. Cifre los mensajes entre el cliente y el gestor de colas; consulte “Privacidad de las publicaciones en los canales de telemetría” en la página 135.

Parámetros de configuración JSSE

Modifique los parámetros JSSE para cambiar la forma en la que se configura una conexión SSL. Los parámetros de configuración de JSSE se organizan en tres conjuntos:

1. Canal de IBM WebSphere MQ Telemetry
2. Cliente Java MQTT
3. JRE

Configure los parámetros del canal de telemetría mediante IBM WebSphere MQ Explorer. Establezca los parámetros del cliente Java MQTT en el atributo `MqttConnectionOptions.SSLProperties` . Modifique los parámetros de seguridad JRE editando los archivos en el directorio de seguridad JRE en el cliente y en el servidor.

Canal de IBM WebSphere MQ Telemetry

Establezca todos los parámetros SSL del canal de telemetría utilizando WebSphere MQ Explorer.

ChannelName

`ChannelName` es un parámetro necesario en todos los canales.

El nombre del canal identifica el canal asociado a un número de puerto particular. Ponga nombre a los canales; le servirá de ayuda para administrar los conjuntos de clientes MQTT.

PortNumber

`PortNumber` es un parámetro opcional en todos los canales. El valor predeterminado es 1883 para canales TCP, y 8883 para los canales SSL.

El número de puerto TCP/IP asociado a este canal. Los clientes MQTT se conectan a un canal especificando el puerto definido para el mismo. Si el canal tiene propiedades SSL, el cliente debe conectarse utilizando el protocolo SSL; por ejemplo:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

KeyFileName

`KeyFileName` es un parámetro necesario para los canales SSL. Con los canales TCP, debe omitirse.

`KeyFileNombre` es la vía de acceso al almacén de claves Java que contiene los certificados digitales que proporciona. Utilice JKS, JCEKS o PKCS12 como el tipo de almacén de claves en el servidor.

Identifique el tipo de almacén de claves utilizando una de las extensiones de archivo siguientes:

```
.jks
.jceks
.p12
.pkcs12
```

Un almacén de claves que tenga cualquier otra extensión de archivo se supone que es un almacén de claves JKS.

Puede combinar un tipo de almacén de claves en el servidor con otros tipos de almacén de claves en el cliente.

Coloque el certificado privado del servidor en el almacén de claves. El certificado se conoce como el certificado del servidor. El certificado puede estar autofirmado o formar parte de una cadena de certificados firmado por una autoridad de firmas.

Si está utilizando una cadena de certificados, coloque los certificados asociados en el almacén de claves del servidor.

El certificado del servidor y cualquier certificado de su cadena de certificados se envían a los clientes para autenticar la identidad del servidor.

Si ha establecido `ClientAuth` en `Required`, el almacén de claves debe contener los certificados necesarios para autenticar el cliente. El cliente envía un certificado autofirmado, o una cadena de certificados, y el cliente se autentica mediante la primera verificación de este material contra un certificado del almacén de claves. Con el uso de una cadena de certificados, un certificado puede verificar muchos clientes, incluso si se emiten con certificados de cliente distintos.

PassPhrase

PassPhrase es un parámetro necesario para los canales SSL. Con los canales TCP, debe omitirse.

La frase de contraseña se utiliza para proteger el almacén de claves.

ClientAuth

ClientAuth es un parámetro SSL opcional. Toma como valor predeterminado que no se autentique ningún cliente. Con los canales TCP, debe omitirse.

Establezca ClientAuth si desea que el servicio de telemetría (MQXR) autentique el cliente, antes de permitir que el cliente se conecte al canal de telemetría.

Si establece ClientAuth, el cliente debe conectarse al servidor utilizando SSL y autenticar el servidor. Como respuesta a la definición de ClientAuth, el cliente envía su certificado digital al servidor y cualquier otro certificado de su almacén de claves. El certificado digital se conoce como el certificado de cliente. Estos certificados se autentican con los certificados contenidos en el almacén de claves del canal, y en el almacén cacerts de JRE.

CipherSuite

CipherSuite es un parámetro SSL opcional. El valor predeterminado es intentar todas las CipherSpecs habilitadas. Con los canales TCP, debe omitirse.

Si desea utilizar una CipherSpec particular, establezca CipherSuite en el mismo nombre que la CipherSpec que debe utilizarse para establecer una conexión SSL.

El servicio de telemetría y el cliente MQTT negocian una CipherSpec común a partir de todas las CipherSpecs que están habilitadas en cada extremo. Si se especifica una CipherSpec determinada en uno de los extremos de la conexión, o en ambos, debe coincidir con la CipherSpec del otro extremo.

Instale otros cifrados añadiendo otros proveedores adicionales a JSSE.

Federal Information Processing Standards (FIPS)

FIPS es un parámetro opcional. De forma predeterminada, no está definido.

Bien en el panel de propiedades del gestor de colas, o bien mediante **runmqsc**, establezca SSLFIPS. SSLFIPS especifica si sólo se deben utilizar algoritmos certificados por FIPS.

Revocation namelist

Revocation namelist es un valor opcional. De forma predeterminada, no está definido.

Bien en el panel de propiedades del gestor de colas, o bien mediante **runmqsc**, establezca SSLCRLNL. SSLCRLNL especifica una lista de nombres de objetos de información de autenticación que se utilizan para proporcionar ubicaciones de revocación de certificados.

No se utiliza ningún otro parámetro de gestor de colas que defina propiedades SSL.

Cliente Java MQTT

Establezca las propiedades SSL para el cliente Java en MqttConnectionOptions.SSLProperties; por ejemplo:

```
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.keyStoreType", "JKS");
com.ibm.micro.client.mqttv3.MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setSSLProperties(sslClientProperties);
```

Los nombres y valores de propiedades específicas se describen en la clase MqttConnectOptions.Para obtener enlaces a la documentación de la API de cliente para las bibliotecas de cliente MQTT, consulte Referencia de programación de cliente MQTT.

Protocol

Protocol es opcional.

El protocolo se selecciona bajo negociación con el servidor de telemetría. Si necesita un protocolo específico, puede seleccionar uno. Si el servidor de telemetría no da soporte al protocolo, la conexión falla.

ContextProvider

`ContextProvider` es opcional.

KeyStore

`KeyStore` es opcional. Configúrelo si ha establecido `ClientAuth` en el servidor para forzar la autenticación del cliente.

Coloque el certificado digital del cliente, firmado mediante su clave privada, en el almacén de claves. Especifique la vía de acceso del almacén de claves y la contraseña. El tipo y el proveedor son opcionales. JKS es el tipo predeterminado, e IBMJCE el proveedor predeterminado.

Especifique un proveedor de almacén de claves diferente que haga referencia a una clase que añada un nuevo proveedor de almacén de claves. Pase el nombre del algoritmo utilizado por el proveedor del almacén de claves para crear una instancia de `KeyManagerFactory` estableciendo el nombre del gestor de claves.

TrustStore

`TrustStore` es opcional. Puede colocar todos los certificados de confianza en el almacén `cacerts` de JRE.

Configure el almacén de confianza si desea tener almacén de confianza diferente para el cliente. Es posible que no pueda configurar el almacén de confianza si el servidor utiliza un certificado emitido por una CA bien conocida que ya tenga su certificado raíz almacenado en `cacerts`.

Añada el certificado firmado públicamente del servidor o el certificado raíz al almacén de confianza, y especifique la vía de acceso de almacén de confianza y la contraseña. JKS es el tipo predeterminado, e IBMJCE el proveedor predeterminado.

Especifique un proveedor de almacén de confianza diferente que haga referencia a una clase que añada un nuevo proveedor de almacén de confianza. Pase el nombre del algoritmo utilizado por el proveedor del almacén de confianza para crear una instancia de `TrustManagerFactory` estableciendo el nombre del gestor de confianza.

JRE

En el JRE se configuran otros aspectos sobre la seguridad Java que afectan al comportamiento del SSL en el cliente y en el servidor. Los archivos de configuración en Windows se encuentran en *Java Installation Directory*`\jre\lib\security`. Si va a utilizar el JRE que se proporciona con IBM WebSphere MQ la vía de acceso es la que se indica en la tabla siguiente:

Tabla 8. Vías de acceso de archivo por plataforma para archivos de configuración JRE SSL

Plataforma	**Vía de acceso de archivo**
Windows	*WMQ Installation Directory*`\java\jre\lib\security`
Otras plataformas UNIX and Linux	*WMQ Installation Directory*`/java/jre64/jre/lib/security`

Entidades emisoras de certificados conocidas

El archivo `cacerts` contiene los certificados raíz de las entidades emisoras de certificados conocidas. El archivo `cacerts` se utiliza de forma predeterminada, a menos que especifique un almacén de confianza. Si utiliza el almacén `cacerts`, o no proporciona ningún almacén de confianza, debe revisar y editar la lista de firmantes que aparece en `cacerts` para que se cumplan sus requisitos de seguridad.

Puede abrir `cacerts` utilizando el mandato `strmqikm`de WebSphere MQ que ejecuta el programa de utilidad IBM Key Management. Abra `cacerts` como un archivo JKS, utilizando la contraseña `changeit`. Modifique la contraseña para proteger el archivo.

Configuración de clases de seguridad

Utilice el archivo `java.security` para registrar proveedores de seguridad adicionales y otras propiedades de seguridad predeterminadas.

Permisos

Utilice el archivo `java.policy` para modificar los permisos otorgados a los recursos. `javaws.policy` otorga permisos a `javaws.jar`

Fuerza de cifrado

Algunos JRE se entregan con poca fuerza de cifrado. Si no puede importar claves a los almacenes de claves, la poca fuerza de cifrado puede ser la causa. Intente iniciar **ikeyman** con el mandato **strmqikm**, o descargue archivos de jurisdicción fuerte, pero limitada, de IBM developer kits, Security information.

Importante: Es posible que su país tenga restricciones sobre la importación, posesión, utilización y nueva exportación a otro país de software cifrado. Antes de descargar o utilizar archivos de políticas sin restricciones, debe comprobar las leyes existentes en su país. Compruebe sus regulaciones y políticas sobre importación, posesión, utilización y nueva exportación de software cifrado para ver si están permitidas estas acciones.

Modificación del proveedor de confianza para permitir al cliente para conectarse a cualquier servidor

En el siguiente ejemplo se muestra cómo añadir un proveedor de confianza y cómo hacer referencia al mismo desde el código de cliente MQTT. En el ejemplo no se autentican el cliente ni el servidor. La conexión SSL resultante se cifra sin que se haya autenticado.

El fragmento de código que aparece en la Figura 25 en la página 140 establece el proveedor de confianza `AcceptAllProviders` y el gestor de confianza del cliente MQTT.

```
java.security.Security.addProvider(new AcceptAllProvider());
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.trustManager","TrustAllCertificates");
sslClientProperties.setProperty("com.ibm.ssl.trustStoreProvider","AcceptAllProvider");
conOptions.setSSLProperties(sslClientProperties);
```

Figura 25. Fragmento de código de cliente MQTT

```
package com.ibm.mq.id;
public class AcceptAllProvider extends java.security.Provider {
    private static final long serialVersionUID = 1L;
    public AcceptAllProvider() {
        super("AcceptAllProvider", 1.0, "Trust all X509 certificates");
        put("TrustManagerFactory.TrustAllCertificates",
                AcceptAllTrustManagerFactory.class.getName());
    }
```

Figura 26. AcceptAllProvider.java

```
    protected static class AcceptAllTrustManagerFactory extends
            javax.net.ssl.TrustManagerFactorySpi {
        public AcceptAllTrustManagerFactory() {}
        protected void engineInit(java.security.KeyStore keystore) {}
        protected void engineInit(
                javax.net.ssl.ManagerFactoryParameters parameters) {}
        protected javax.net.ssl.TrustManager[] engineGetTrustManagers() {
            return new javax.net.ssl.TrustManager[] { new AcceptAllX509TrustManager() };
        }
```

Figura 27. AcceptAllTrustManagerFactory.java

```
protected static class AcceptAllX509TrustManager implements
        javax.net.ssl.X509TrustManager {
    public void checkClientTrusted(
            java.security.cert.X509Certificate[] certificateChain,
            String authType) throws java.security.cert.CertificateException {
        report("Client authtype=" + authType);
        for (java.security.cert.X509Certificate certificate : certificateChain) {
            report("Accepting:" + certificate);
        }
    }
    public void checkServerTrusted(
            java.security.cert.X509Certificate[] certificateChain,
            String authType) throws java.security.cert.CertificateException {
        report("Server authtype=" + authType);
        for (java.security.cert.X509Certificate certificate : certificateChain) {
            report("Accepting:" + certificate);
        }
    }
    public java.security.cert.X509Certificate[] getAcceptedIssuers() {
        return new java.security.cert.X509Certificate[0];
    }
private static void report(String string) {
       System.out.println(string);
  }
}
```

Figura 28. AcceptAllX509TrustManager.java

Configuración JAAS del canal de telemetría

Configure JAAS para autenticar el valor de `Username` que envía el cliente.

El administrador de WebSphere MQ configura qué canales MQTT que necesitan autenticación de cliente mediante JAAS. Especifique el nombre de una configuración JAAS para cada canal que vaya a realizar la autenticación JAAS. Los canales pueden utilizar todos la misma configuración JAAS o utilizar configuraciones JAAS diferentes. Las configuraciones se definen en *WMQData directory*`\qmgrs\`*qMgrName*`\mqxr\jaas.config`.

El archivo `jaas.config` está organizado por nombre de configuración JAAS. Bajo cada nombre de configuración aparece una lista de las configuraciones de inicio de sesión; consulte la Figura 29 en la página 142.

JAAS proporciona cuatro módulos de inicio de sesión estándar. Los módulos de inicio de sesión estándar de NT y UNIX son de valor limitado.

JndiLoginModule
Autentica un servicio de directorio configurado con JNDI (Java Naming and Directory Interface).

Krb5LoginModule
Autentica utilizando protocolos Kerberos.

NTLoginModule
Autentica utilizando la información de seguridad de NT para el usuario actual.

UnixLoginModule
Autentica utilizando la información de seguridad UNIX del usuario actual.

El problema al utilizar `NTLoginModule` o `UnixLoginModule` es que el servicio de telemetría (MQXR) se ejecuta con la identidad `mqm` y no con la identidad del canal MQTT. `mqm` es la identidad que se pasa a `NTLoginModule` o a `UnixLoginModule` para la autenticación, y no la identidad del cliente.

Para solucionar este problema, escriba un módulo de inicio de sesión propio o utilice otros módulos de inicio de sesión estándar. Con WebSphere MQ Telemetry se proporciona un archivo `JAASLoginModule.java` de ejemplo. Es una implementación de la interfaz de `javax.security.auth.spi.LoginModule`. Utilícelo para desarrollar un método de autenticación propio.

Las clases nuevas de LoginModule que proporcione deben encontrarse en la vía de acceso de clase del servicio de telemetría (MQXR). No coloque sus clases en los directorios de WebSphere MQ que estén en

la vía de acceso de clases. Cree sus propios directorios y defina la vía de acceso de clase completa del servicio de telemetría (MQXR).

Puede aumentar la vía de acceso de clase que utiliza el servicio de telemetría (MQXR) estableciendo la vía de acceso de clase en el archivo `service.env`. CLASSPATH debe estar en mayúsculas y la sentencia de vía de acceso de clase sólo puede contener literales. No puede utilizar variables en CLASSPATH; por ejemplo CLASSPATH=%CLASSPATH% no es correcto. El servicio de telemetría (MQXR) establece su propia vía de acceso de clases. Se añade el valor CLASSPATH definido en el archivo `service.env`.

El servicio de telemetría (MQXR) proporciona dos devoluciones de llamada que devuelven el valor de `Username` y de `Password` del cliente conectado al canal MQTT. El `Nombre de usuario` y la `Contraseña` se establecen en el objeto `MqttConnectOptions` . Consulte Figura 30 en la página 142 para obtener un ejemplo de cómo acceder al `Nombre de usuario` y la `Contraseña`.

Ejemplos

Ejemplo de un archivo de configuración JAAS con una configuración denominada `MQXRConfig`.

```
MQXRConfig {
  samples.JAASLoginModule required debug=true;
  //com.ibm.security.auth.module.NTLoginModule required;
  //com.ibm.security.auth.module.Krb5LoginModule required
  //                 principal=principal@your_realm
  //                 useDefaultCcache=TRUE
  //                 renewTGT=true;
  //com.sun.security.auth.module.NTLoginModule required;
  //com.sun.security.auth.module.UnixLoginModule required;
  //com.sun.security.auth.module.Krb5LoginModule required
  //                 useTicketCache="true"
  //                 ticketCache="${user.home}${/}tickets";
};
```

Figura 29. Archivo `jaas.config` de ejemplo

Ejemplo de un módulo de inicio de sesión JAAS codificado para recibir el valor de `Username` y de `Password` que proporciona un cliente MQTT.

```
  public boolean login()
      throws javax.security.auth.login.LoginException {
    javax.security.auth.callback.Callback[] callbacks =
      new javax.security.auth.callback.Callback[2];
    callbacks[0] = new javax.security.auth.callback.NameCallback("NameCallback");
    callbacks[1] = new javax.security.auth.callback.PasswordCallback(
        "PasswordCallback", false);
    try {
      callbackHandler.handle(callbacks);
      String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
          .getName();
      char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
          .getPassword();
      // Accept everything.
      if (true) {
        loggedIn = true;
      } else
        throw new javax.security.auth.login.FailedLoginException("Login failed");

      principal= new JAASPrincipal(username);

    } catch (java.io.IOException exception) {
      throw new javax.security.auth.login.LoginException(exception.toString());
    } catch (javax.security.auth.callback.UnsupportedCallbackException exception) {
      throw new javax.security.auth.login.LoginException(exception.toString());
    }

    return loggedIn;
  }
```

Figura 30. Método `JAASLoginModule.Login()` de ejemplo

Daemon de IBM WebSphere MQ Telemetry para conceptos de dispositivos

El daemon de IBM WebSphere MQ Telemetry para dispositivos es una aplicación cliente de MQTT V3 avanzada. Utilícelo para almacenar y reenviar mensajes desde otros clientes MQTT. Se conecta a IBM WebSphere MQ como un cliente MQTT, pero también puede conectarle otros clientes MQTT.

El daemon es un intermediario de publicación/suscripción. Los clientes MQTT V3 se conectan al él para publicar y suscribirse a temas, utilizando las series de tema para publicar y los filtros de temas para suscribirse. La serie de tema es jerárquica, con niveles de tema divididos por /. Los filtros de temas son series de tema que pueden incluir comodines + de un solo nivel y un comodín # multinivel como la última parte de la serie de tema.

Nota: Los comodines del daemon siguen las reglas más restrictivas de WebSphere Message Broker, v6. IBM WebSphere MQ es diferente. Da soporte a varios comodines multinivel; puede representar cualquier número de niveles de la jerarquía, en cualquier lugar de la serie de tema.

Varios clientes MQTT v3 se conectan al daemon utilizando un puerto de escucha. El puerto de escucha predeterminado puede modificarse. Puede definir varios puertos de escucha y asignar espacios de nombres distintos a los mismos, consulte “Puertos de escucha del daemon de WebSphere MQ Telemetry para dispositivos” en la página 151. El daemon, en sí mismo, es un cliente MQTT v3. Configure una conexión de puente de daemon para conectar el daemon al puerto de escucha de otro daemon,o a un servicio de WebSphere MQ Telemetry (MQXR).

Puede configurar varios puentes para el daemon de WebSphere MQ Telemetry para dispositivos. Utilice los puentes para interconectar una red de daemons que puedan intercambiar publicaciones.

Cada puente puede publicar y suscribirse a temas en su daemon local. También puede publicar y suscribirse a temas en otro daemon, en un intermediario de publicación/suscripción de WebSphere MQ o en cualquier otro intermediario MQTT v3 al que se haya conectado. Al utilizar un filtro de temas, puede seleccionar las publicaciones para propagar de un intermediario a otro. Puede propagar las publicaciones en cualquier dirección. Puede propagar las publicaciones del daemon local a cada uno de sus intermediarios remotos conectados, o de cualquiera de los intermediarios conectados al daemon local; consulte “Daemon de IBM WebSphere MQ Telemetry para puentes de dispositivos” en la página 143.

Daemon de IBM WebSphere MQ Telemetry para puentes de dispositivos

Un puente de IBM WebSphere MQ Telemetry para el puente de dispositivos conecta dos intermediarios de publicación/suscripción utilizando el protocolo MQTT v3. El puente propaga las publicaciones de un intermediario a otro, en cualquier dirección. En un extremo hay una conexión de puente del daemon de WebSphere MQ Telemetry para dispositivos, y en el otro puede haber un gestor de colas, u otro daemon. Hay un gestor de colas conectado a la conexión de puente mediante un canal de telemetría. Hay un daemon conectado a la conexión de puente mediante un escucha de daemon.

El daemon IBM WebSphere MQ Telemetry para dispositivos admite una o más conexiones simultáneas a otros intermediarios. Las conexiones procedentes del daemon se llaman puentes, y se definen mediante las entradas de conexión del archivo de configuración del daemon. Las conexiones a IBM WebSphere MQ se establecen mediante los canales de telemetría de IBM WebSphere MQ, tal como se muestra en la figura siguiente:

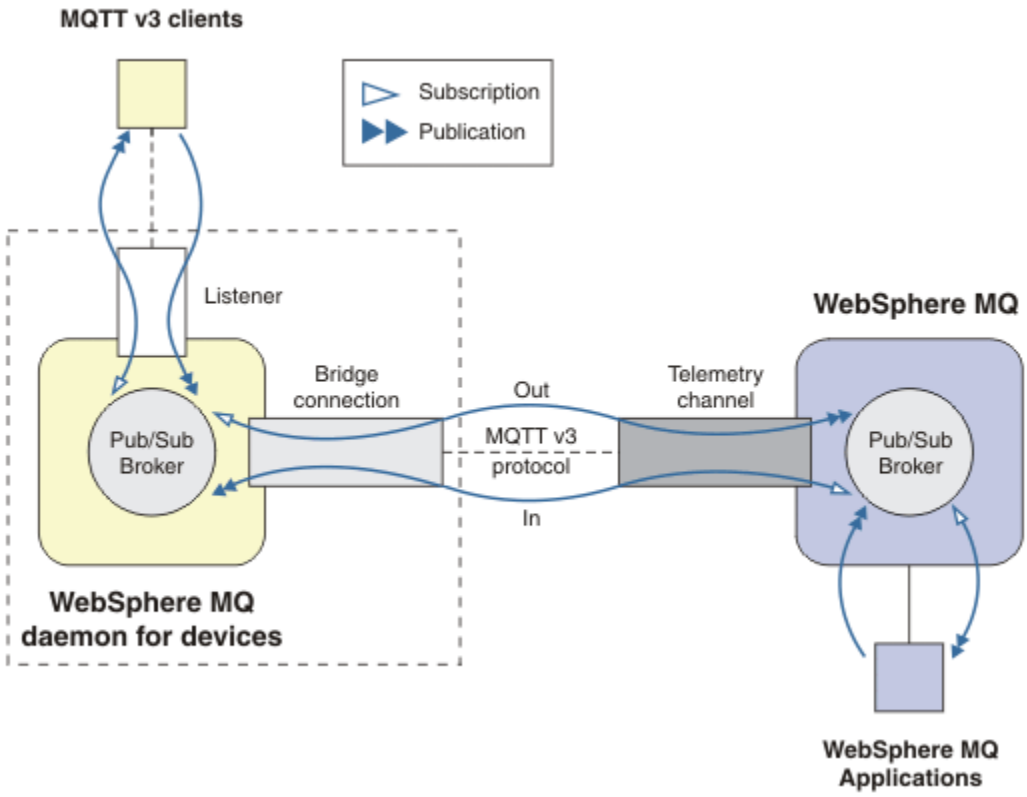


Figura 31. Conexión de IBM WebSphere MQ Telemetry daemon for devices a IBM WebSphere MQ

Un puente conecta el daemon a otro intermediario como un cliente MQTT v3. Los parámetros del puente duplican los atributos de un cliente MQTT v3.

Un puente es más que una conexión. Actúa como un agente de publicación y suscripción, situado entre dos intermediarios de publicación/suscripción. El intermediario local es el daemon IBM WebSphere MQ Telemetry para dispositivos, y el intermediario es cualquier intermediario de publicación/suscripción remoto que dé soporte al protocolo MQTT v3. Normalmente, el intermediario remoto es otro daemon, o IBM WebSphere MQ.

La función del puente es a propagar publicaciones entre ambos intermediarios. El puente es bidireccional. Propaga las publicaciones en cualquier dirección. En la Figura 31 en la página 144 se ilustra el modo en que el puente conecta el daemon de IBM WebSphere MQ Telemetry para dispositivos a IBM WebSphere MQ. En “Valores de tema de ejemplo para el puente” en la página 145 se utilizan ejemplos que ilustran cómo utilizar el parámetro topic para configurar el puente.

Las flechas In (entrada) y Out (salida) que aparecen en la Figura 31 en la página 144 indican la bidireccionalidad del puente. En un extremo de la flecha se crea una suscripción. Las publicaciones que coincidan con la suscripción se publican en el intermediario situado en el extremo opuesto de la flecha. La etiqueta de la flecha depende del flujo de las publicaciones. El flujo de las publicaciones entra (In) en el daemon, y sale (Out) del daemon. La importancia de las etiquetas es que se utilizan en la sintaxis del mandato. Recuerde que In y Out hacen referencia a dónde fluyen las publicaciones, y no a dónde se envía la suscripción.

Otros clientes, otras aplicaciones u otros intermediarios pueden estar conectados a IBM WebSphere MQ o al daemon de WebSphere MQ Telemetry para dispositivos. Publican y se suscriben a temas en el intermediario al que están conectados. Si el intermediario es IBM WebSphere MQ, los temas pueden estar en clúster o distribuidos, y no definidos de forma explícita en el gestor de colas local.

Usos de los puentes

Puede conectar los daemons entre sí, utilizando conexiones de puente y escuchas. Puede conectar los demonios y los gestores de colas entre sí, utilizando conexiones de puente y canales de telemetría. Cuando se conectan varios intermediarios entre sí, es posible que se creen bucles. Atención: las publicaciones podrían circular indefinidamente dentro de un bucle de intermediarios, sin detectarse.

Algunas de las razones para utilizar daemons conectados a IBM WebSphere MQ son las siguientes:

Reducir el número de conexiones de cliente MQTT a WebSphere MQ.
Utilizar una jerarquía de daemons, puede conectar muchos clientes a WebSphere MQ; más clientes que los que un solo gestor de colas puede conectar a la vez.

Almacenar y reenviar los mensajes entre clientes MQTT y WebSphere MQ.
Puede utilizar el almacenamiento y el reenvío para evitar tener que mantener conexiones continuas entre los clientes y el IBM WebSphere MQ, si los clientes no disponen de su propio almacenamiento. Puede utilizar varios tipos de conexiones entre el cliente MQTT y WebSphere MQ; consulte Conceptos y escenarios de telemetría para supervisión y control.

Filtrar las publicaciones intercambiadas entre los clientes MQTT y WebSphere MQ.
Normalmente, las publicaciones se dividen en los mensajes que se procesan localmente y los mensajes en los que intervienen otras aplicaciones. Las publicaciones locales pueden incluir flujos de control entre sensores y actuadores, y las publicaciones remotas incluyen peticiones de mandatos de lecturas, estado y de configuración.

Cambiar los espacio de temas de las publicaciones.
Evite que las series de tema procedentes de clientes conectados a puertos de escucha distintos colisionen entre sí. En el ejemplo, se utiliza el daemon para etiquetar las lecturas de medidor procedentes de edificios diferentes; consulte Separación de espacios de temas de grupos de clientes diferentes.

Valores de tema de ejemplo para el puente

Publicar todo en el intermediario remoto: utilizar los valores predeterminados

La dirección predeterminada se conoce como out, y el puente publica los temas en el intermediario remoto. El parámetro topic controla qué temas se propagan utilizando los filtros de tema.

El puente utiliza el parámetro topic en Figura 32 en la página 145 para suscribirse a todo lo publicado en el daemon local por los clientes MQTT, o por otros intermediarios. El puente publica los temas en el intermediario remoto que se ha conectado mediante el puente.

```
connection Daemon1
topic #
```

Figura 32. Publicar todo en el intermediario remoto

Publicar todo en el intermediario remoto: explicito

El valor de topic en el fragmento de código siguiente proporciona el mismo resultado que utilizar los valores predeterminados. La única diferencia es que el parámetro **direction** es explícito. Utilice dirección `out` para suscribirse al intermediario local, el daemon, y publicar en el intermediario remoto. Las publicaciones creadas en el daemon local al cual el puente se haya suscrito, se publican en el intermediario remoto.

```
connection Daemon1
topic # out
```

Figura 33. Publicar todo en el intermediario remoto: explicito

Publicar todo en el intermediario local

En lugar de utilizar la dirección `out`, puede establecer la dirección contraria, `in`. El fragmento de código siguiente configura el puente para suscribirse a todo lo publicado en el intermediario remoto conectado al mismo. El puente publica los temas en el intermediario local, el daemon.

```
connection Daemon1
topic # in
```

Figura 34. Publicar todo en el intermediario local

Publicar todo el tema export del intermediario local en el tema import del intermediario remoto

Puede utilizar dos parámetros topic adicionales, **`local_prefix`** y **`remote_prefix`**, para modificar el filtro de tema, `#` en los ejemplos anteriores. Un parámetro se utiliza para modificar el filtro utilizado en el tema de la suscripción y el otro parámetro se utiliza para modificar el tema en el que se publica la publicación. El efecto que se consigue es sustituir el principio de la serie de tema que se utiliza en un intermediario por otra serie de tema en el otro intermediario.

En función de la dirección del mandato topic, el significado de **`local_prefix`** y de **`remote_prefix`** se invierte. Si la dirección es `out`, el valor predeterminado, se utiliza **`local_prefix`** como parte de la suscripción del tema, y **`remote_prefix`** sustituye a la parte **`local_prefix`** de la serie de tema, en la publicación remota. Si la dirección es `in`, **`remote_prefix`** se convierte en parte de la suscripción remota, y **`local_prefix`** sustituye a la parte **`remote_prefix`** de la serie de tema.

La primera parte de una serie de tema se considera, a menudo, como la parte que define un espacio de temas. Utilice los parámetros adicionales para cambiar el espacio de temas en el que se publica un tema. Puede hacer esto para evitar que el tema se propague y colisione con otro tema en el intermediario de destino, o para eliminar una serie de tema de punto de montaje.

Por ejemplo, en el fragmento de código siguiente, todas las publicaciones a la serie de tema `export/#` en el daemon, se vuelven a publicar como `import/#` en el intermediario remoto.

```
topic # out export/ import/
```

Figura 35. Publicar todo el tema export del intermediario local en el tema import del intermediario remoto

Publicar en el tema import del intermediario local todo lo procedente del tema export del intermediario remoto

El fragmento de código siguiente muestra la configuración invertida; el puente se suscribe a todo lo publicado con la serie de tema `export/#` en el intermediario remoto y lo publica en `import/#` en el intermediario local.

```
connection Daemon1
topic # in import/ export/
```

Figura 36. Publicar en el tema import del intermediario local todo lo procedente del tema export del intermediario remoto

Publicar todo desde el punto de montaje 1884/ al intermediario remoto, con las series de tema originales

En el fragmento de código siguiente, el puente se suscribe a todo lo publicado por los clientes conectados al punto de montaje 1884/ en el daemon local. El puente publica en el intermediario remoto todo lo publicado en el punto de montaje. La serie de tema del punto de montaje 1884/ se

elimina de los temas publicados en el intermediario remoto. El *local_prefix* es el mismo que la serie del punto de montaje 1884/ y el *remote_prefix* es una serie en blanco.

```
listener 1884
mount_point 1884/
connection Daemon1
topic # out 1884/ ""
```

Figura 37. Publicar todo desde el punto de montaje 1884/ al intermediario remoto, con las series de tema originales.

Separación de espacios de temas de clientes diferentes conectados a daemons diferentes

Una aplicación está escrita para que los medidores de corriente eléctrica publiquen las lecturas de medidor de un edificio. Las lecturas se publican utilizando clientes MQTT a un daemon que se aloja en el mismo edificio. El tema seleccionado para que las publicaciones es `power`. La misma aplicación se despliega en varios edificios de un complejo. Para la supervisión y el almacenamiento de datos in situ, se agregan las lecturas de todos los edificios, utilizando las conexiones de puente. Las conexiones enlazan los daemons del edificio a WebSphere MQ, que se encuentra en una ubicación central.

Las aplicaciones cliente en cada edificio son idénticas, pero los datos deben diferenciarse, según el edificio al que pertenezcan. Cada lectura tiene un tema `power`, y debe tener como prefijo el número de edificio, para poder distinguirlo. El puente del primer edificio del complejo utiliza el prefijo `meters/building01/`; el prefijo del segundo edificio es `meters/building02/`. Las lecturas de los demás edificios siguen el mismo patrón. WebSphere MQ recibe las lecturas con temas como `meters/building01/power`.

El ejemplo es provocado; en la práctica, es probable que el espacio de temas en el que la aplicación publica se pueda configurar.

El archivo de configuración de cada daemon tiene una sentencia de tema que sigue el patrón del fragmento de código siguiente:

```
connection Daemon1
topic power out "" meters/building01/
```

Figura 38. Separar espacios de temas de clientes conectados a daemons diferentes

Especifique una serie vacía como un espacio reservado para el parámetro local_prefix no utilizado.

Separar espacios de temas de clientes conectados al mismo daemon.

Supongamos que se utiliza único daemon para conectar todos los medidores de corriente eléctrica. Suponiendo que la aplicación puede configurarse para conectarse a diferentes puertos, se pueden distinguir los edificios conectando los medidores de edificios diferentes a puertos de escucha diferentes, tal como se muestra en el fragmento de código siguiente. De nuevo, el ejemplo no es real; en él se ilustra cómo se pueden utilizar los puntos de montaje.

```
listener 1884
mount_point meters/building01/
listener 1885
mount_point meters/building02/
connection Daemon1
topic meters/+/power out
```

Figura 39. Separar espacios de temas de clientes conectados al mismo daemon.

Volver a correlacionar temas diferentes para las publicaciones que fluyen en ambas direcciones.

En la configuración del siguiente fragmento de código, el puente se suscribe al tema único b en el intermediario remoto y reenvía publicaciones sobre b al daemon local, cambiando el tema a a. El puente también se suscribe al tema único x en el intermediario local y reenvía publicaciones sobre x al intermediario remoto, cambiando el tema a y.

```
connection Daemon1
topic "" in a b
topic "" out x y
```

Figura 40. Volver a correlacionar temas diferentes para las publicaciones que fluyen en ambas direcciones.

Un punto importante de este ejemplo es que ambos intermediarios están suscritos a diferentes temas y en ambos se publican los diferentes temas. Los espacios de temas en ambos intermediarios están separados.

Volver a correlacionar los mismos temas para las publicaciones que fluyen en ambas direcciones (repetición en bucle).

A diferencia del ejemplo anterior, la configuración de Figura 41 en la página 148 generalmente da como resultado un bucle. En la sentencia topic `topic "" in a b`, el puente se suscribe a b de forma remota, y publica en a de forma local. En la otra sentencia topic, el puente se suscribe a a de forma local, y publica de forma remota en b. La misma configuración puede escribirse tal como se muestra en Figura 42 en la página 148.

El resultado general es que si un cliente publica en b de forma remota, la publicación se transfiere al daemon local como una publicación sobre el tema a. Sin embargo, al ser publicado por el puente en el daemon local en el tema a, la publicación coincide con la suscripción realizada por el puente en el tema local a. La suscripción es `topic "" out a b`. Como resultado, la publicación se transfiere de nuevo al intermediario remoto como una publicación sobre el tema b. Ahora el puente está suscrito al tema remoto by el ciclo se inicia de nuevo.

Algunos intermediarios implementan la detección de bucles para evitar que se generen bucles. Sin embargo, el mecanismo de detección de bucles debe funcionar cuando diferentes tipos de intermediarios están conectados entre sí mediante un puente. La detección de bucle no funciona si WebSphere MQ tiene un puente que lo conecta al daemon de WebSphere MQ Telemetry para dispositivos. Sí funciona cuando dos daemons IBM WebSphere MQ Telemetry para dispositivos están conectados entre sí mediante un puente. De forma predeterminada, la detección de bucle está activada; consulte try_private.

```
connection Daemon1
topic "" in a b
topic "" out a b
```

Figura 41. !Volver a correlacionar los mismos temas para las publicaciones que fluyen en ambas direcciones

```
connection Daemon1
topic "" both a b
```

Figura 42. !Vuelva a correlacionar los mismos temas para las publicaciones que fluyen en ambas direcciones, utilizando `both`.

La configuración que aparece en Figura 40 en la página 148 es la misma que la de Figura 41 en la página 148.

Disponibilidad de las conexiones de puente del IBM WebSphere MQ Telemetry daemon for devices

Configure varias direcciones de conexión de puente del IBM WebSphere MQ Telemetry daemon for devices para poder conectarse al primer intermediario remoto que haya disponible. Si el intermediario es un gestor de colas de varias instancias, proporcione ambas direcciones TCP/IP. Configure una conexión primaria para conectar, o volver a conectar, al servidor primario, cuando éste esté disponible.

El parámetro del puente de la conexión, addresses, es una lista de direcciones de socket de TCP/IP. El puente intentará conectarse a cada dirección por turnos, hasta que se realice una conexión satisfactoria. Los parámetro de conexión round_robin y start_type controlan cómo se utilizan las direcciones una vez que ya se ha efectuado una conexión satisfactoria.

Si start_type es auto, manual o active, y la conexión falla, el puente intenta conectarse de nuevo. Utiliza cada dirección por turnos, con un retardo de 20 segundos entre cada intento de conexión. Si `start_type` es once, y la conexión falla, el puente no intenta conectarse de nuevo automáticamente.

Si round_robin es true, la conexión de puente intenta iniciarse en la primera dirección de la lista, y lo intenta con cada dirección de la lista, una por una. Cuando se agota la lista, vuelve a empezar en la primera dirección. Si sólo hay una dirección en la lista, lo vuelve a intentar cada 20 segundos.

Si round_robin es false, se otorga preferencia a la primera dirección de la lista, que se conoce como el servidor primario. Si el primer intento de conexión al servidor primario falla, el puente sigue intentando conectarse al servidor primario en segundo plano. Al mismo tiempo, el puente intenta conectarse utilizando el resto de direcciones de la lista. Cuando los intentos de conexión al servidor primario que se efectúan en segundo plano resulten satisfactorios, el puente se desconectará de la conexión actual, y conmutará a la conexión del servidor primario.

Si una conexión se ha desconectado de forma voluntaria, por ejemplo mediante la emisión de un mandato **`connection_stop`**, en ese caso, si se reinicia la conexión, ésta intenta utilizar la misma dirección otra vez. Si la conexión se ha desconectado debido a una anomalía de conexión o se debe a que el intermediario remoto ha finalizado la conexión, el puente espera durante 20 segundos. A continuación, trata de conectarse a la siguiente dirección de la lista, o a la misma dirección, si la lista sólo contiene una dirección.

Conexión a un gestor de colas de varias instancias

En una configuración de gestor de colas de varias instancias, el gestor de colas se ejecuta en dos servidores distintos con direcciones IP distintas. Normalmente, los canales de telemetría se configuran sin una dirección IP específica. Se configuran sólo con un número de puerto. Cuando se inicia el canal de telemetría, éste selecciona, de forma predeterminada, la primera dirección de red disponible que haya en el servidor local.

Configure el parámetro addresses de la conexión de puente con las dos direcciones IP que utilice el gestor de colas. Establezca round_robin en true.

Si la instancia de gestor de colas activo falla, el gestor de colas conmuta a la instancia que está en espera. El daemon detecta que la conexión con la instancia activa ha finalizado, e intenta conectarse de nuevo a la instancia que está en espera. Utiliza la otra dirección IP de la lista de direcciones configuradas para la conexión de puente.

El gestor de colas al cual se conecta el puente, sigue siendo el mismo gestor de colas. El gestor de colas recupera su propio estado. Si cleansession se ha establecido en false, la conexión de puente se restaura al mismo estado que tenía antes de producirse la migración tras error. La conexión se reanuda después de un retardo. Los mensajes con "al menos una vez" o "como máximo una vez" de calidad de servicio no se pierden y las suscripciones siguen funcionando.

El tiempo de reconexión depende del número de canales y de clientes que se reinicien al iniciarse la instancia que está en espera, y de cuántos mensajes había en curso. La conexión de puente puede tratar de volver a conectarse a ambas direcciones IP varias veces, antes de que se restablezca la conexión.

No configure un canal de telemetría de gestor de colas de varias instancias con una dirección IP específica. La dirección IP sólo es válida en un servidor.

Si utiliza una solución de alta disponibilidad alternativa, que gestione la dirección IP, esta opción puede ser correcta para configurar un canal de telemetría con una dirección IP específica.

cleansession

Una conexión de puente es una sesión de cliente MQTT v3. Puede controlar si una conexión se inicia una nueva sesión, o si restaura una sesión existente. Si restaura una sesión existente, la conexión de puente conserva las suscripciones y las publicaciones retenidas de la sesión anterior.

No establezca cleansession en false si figuran varias direcciones IP en addresses, y éstas se conectan a canales de telemetría alojados en gestores de colas diferentes, o a daemons diferentes. El estado de la sesión no se transfiere entre los gestores de colas ni los daemons. Tratar de reiniciar una sesión existente en un gestor de colas o daemon diferente da como resultado que se inicie una nueva sesión. Los mensajes dudosos se pierden, y es posible que las suscripciones no se comporten como cabría esperar.

notifications

Una aplicación puede hacer un seguimiento de si se está ejecutando la conexión de puente, utilizando las notificaciones. Una notificación es una publicación que tiene el valor `1`, conectada, o `0`, desconectada. Se publica en *SerieTema* que se define mediante el parámetro `notification_topic`. El valor predeterminado de *topicString* es $SYS/`broker`/`connection`/ *clientIdentifier*/`state`. El valor predeterminado de *topicString* contiene el prefijo $SYS. Puede suscribirse a los temas que empiecen por $SYS definiendo un filtro de tema que empiece por $SYS. El filtro de tema *#*, suscribirse a todo, no efectúa la suscripción a los temas que empiecen por $SYS en el daemon. Piense en $SYS como si definiera un espacio de temas del sistema especial distinto del espacio de temas de la aplicación.

Las notificaciones permite que el IBM WebSphere MQ Telemetry daemon for devices notifique a los clientes MQTT cuando un puente está conectado o desconectado.

keepalive_interval

El parámetro de conexión de puente keepalive_interval establece el intervalo que el puente utiliza para enviar un ping TCP/IP al servidor remoto. El intervalo predeterminado es de 60 segundos. El envío de pings impide que el servidor remoto, o un cortafuegos, cierre la sesión TCP/IP, si se detecta un período de inactividad en la conexión.

clientId

Una conexión de puente es una sesión de cliente MQTT v3, y tiene un valor `clientIdentifier` que se ha establecido mediante el parámetro de conexión de puente clientid. Si tiene la intención de que las reconexiones reanuden una sesión anterior estableciendo el parámetro cleansession en false, el valor de `clientIdentifier` que se utilice en cada sesión deberá ser el mismo. El valor predeterminado de clientid es *nombre_host.NombreConexión*, que continúa siendo el mismo.

Instalación, verificación, configuración y control del daemon de WebSphere MQ Telemetry daemon para dispositivos

La instalación, la verificación, la configuración y el control se basan en archivos.

Instale el daemon copiando el kit de desarrollo de software en el dispositivo donde va a ejecutar el daemon .

Como ejemplo, ejecute el programa de utilidad cliente MQTT y conéctese al daemon de WebSphere MQ Telemetry para dispositivos como intermediario de publicación/suscripción; consulte Publicar un mensaje en un cliente MQTT v3 específico .

Configure el daemon mediante la creación de un archivo de configuración; consulte Archivo de configuración del daemon de WebSphere MQ Telemetry para dispositivos.

Controle un daemon que esté en ejecución mediante la creación de mandatos en el archivo `amqtdd.upd`. Cada 5 segundos, el daemon lee el archivo, ejecuta los mandatos, y suprime el archivo; consulte Archivo de mandatos del daemon de WebSphere MQ Telemetry para dispositivos.

Puertos de escucha del daemon de WebSphere MQ Telemetry para dispositivos

Conecte los clientes MQTT V3 al daemon de WebSphere MQ Telemetry para dispositivos utilizando los puertos de escucha. Puede calificar un puerto de escucha con un punto de montaje y un número máximo de conexiones.

Un puerto de escucha debe corresponder al número de puerto especificado en el método `connect(serverURI)` del cliente MQTT, de un cliente que se conecte a este puerto. Toma, como valor predeterminado 1883, tanto en el cliente como en el daemon.

Puede cambiar el puerto predeterminado para el daemon estableciendo la definición global `port` del archivo de configuración del daemon. Puede establecer puertos específicos añadiendo una definición `listener` al archivo de configuración del daemon.

Para cada puerto de escucha, que no sea el puerto predeterminado, puede especificar un punto de montaje para aislar los clientes. Los clientes conectados a un puerto que tengan un punto de montaje están aislados de otros clientes; consulte “Puntos de montaje del daemon de WebSphere MQ Telemetry para dispositivos” en la página 151.

Puede limitar el número de clientes que pueden conectarse a cualquier puerto. Establezca la definición global `max_connections` para limitar las conexiones al puerto predeterminado, o calificar cada puerto de escucha con `max_connections`.

Ejemplo

Un ejemplo de un archivo de configuración que cambia el puerto predeterminado 1883 por 1880, y limita las conexiones al puerto 1880 a 10000. Las conexiones al puerto 1884 están limitadas a 1000 conexiones. Los clientes conectados al puerto 1884 están aislados de los clientes conectados a otros puertos.

```
port 1880
max_connections 10000
listener 1884
mount_point 1884/
max_connections 1000
```

Puntos de montaje del daemon de WebSphere MQ Telemetry para dispositivos

Puede asociar un punto de montaje con un puerto de escucha que los clientes MQTT utilizan para conectarse a un daemon de WebSphere MQ Telemetry para dispositivos. Un punto de montaje aísla las publicaciones y las suscripciones que los clientes MQTT intercambian utilizando un puerto de escucha de los clientes MQTT que estén conectados a un puerto de escucha distinto.

Los clientes conectados a un puerto de escucha que tengan un punto de montaje nunca puede intercambiar temas directamente con los clientes que estén conectados a cualquier otro puerto de escucha. Los clientes que estén conectados a un puerto de escucha que no tengan un punto de montaje pueden publicar o suscribirse a temas de cualquier cliente. Los clientes no son conscientes de si están conectados a través de un punto de montaje o no; no tiene ninguna diferencia respecto a las series de tema que crean los clientes.

Un punto de montaje es una serie de texto que se prefija en la serie de tema de las publicaciones y las suscripciones. Actúa como prefijo de todas las series de tema que han creado los clientes conectados al puerto de escucha que tengan un punto de montaje. La serie de texto se elimina de todas las series de tema enviadas a los clientes que estén conectados al puerto de escucha.

Si un puerto de escucha no tiene ningún punto de montaje, no se modifican las series de tema de las publicaciones y suscripciones creadas y recibidas por los clientes conectados al puerto.

Cree series de punto de montaje con un carácter / final. De este modo, el punto de montaje es el tema padre del árbol de tema del punto de montaje.

Ejemplo

Un archivo de configuración contiene los puertos de escucha siguientes:

```
listener 1883
mount_point 1883/
listener 1884 127.0.0.1
mount_point 1884/
listener 1885
```

Un cliente, conectado al puerto 1883, crea una suscripción a `MyTopic`. El daemon registra la suscripción como `1883/MyTopic`. Otro cliente conectado al puerto 1883 publica un mensaje sobre el tema, `MyTopic`. El daemon cambia la serie de tema a `1883/MyTopic` y busca suscripciones coincidentes. El suscriptor en el puerto 1883 recibe la publicación con la serie de tema original `MyTopic`. El daemon ha eliminado el prefijo del punto de montaje de la serie de tema.

Otro cliente, conectado al puerto 1884, también publica sobre el tema `MyTopic`. Esta vez el daemon registra el tema como `1884/MyTopic`. El suscriptor en el puerto 1883 no recibe la publicación, debido a que el punto de montaje diferente da como resultado una suscripción que tiene una serie de tema diferente.

Un cliente, conectado al puerto 1885, publica sobre el tema, `1883/MyTopic`. El daemon no cambia la serie de tema. El suscriptor en el puerto 1883 recibe la publicación en `MyTopic`.

Calidad de servicio del daemon de WebSphere MQ Telemetry para dispositivos, suscripciones duraderas y publicaciones retenidas

Los valores de la calidad del servicio se aplican sólo a un daemon que esté en ejecución. Si un daemon se detiene, ya sea de una forma controlada, o debido a una anomalía, se pierde el estado de los mensajes en curso. No se puede garantizar la entrega de un mensaje al menos una vez, o como máximo una vez, si se detiene el daemon. El daemon de WebSphere MQ Telemetry para dispositivos da soporte a la persistencia limitada. Establezca el parámetro de configuración **`retained_persistence`** para guardar las publicaciones retenidas y las suscripciones cuando se concluye el daemon.

A diferencia de WebSphere MQ, el daemon de WebSphere MQ Telemetry para dispositivos no registra por diario los datos persistentes. El estado de la sesión, el estado de los mensajes y las publicaciones retenidas no se guardan de forma transaccional. De forma predeterminada, el daemon descarta todos los datos cuando se detiene. Puede establecer una opción para aplicar un punto de comprobación a las suscripciones y a las publicaciones retenidas periódicamente. El estado de los mensajes se pierde siempre cuando se detiene el daemon. Se pierden todas las publicaciones que no se hayan retenido.

Establezca la opción de configuración del daemon `Retained_persistence` en `true`, para guardar periódicamente en un archivo las publicaciones retenidas. Cuando se reinicia el daemon, se vuelve a crear una instancia de las publicaciones retenidas que se guardaron automáticamente la última vez. De forma predeterminada, cuando el daemon se reinicia no se vuelven a crear las instancias de los mensajes retenidos que hayan creado los clientes.

Establezca la opción de configuración del daemon `Retained_persistence` en `true`, para guardar periódicamente en un archivo las subscripciones creadas en una sesión persistente. Si se establece `Retained_persistence` en `true`, las suscripciones que los clientes creen en una sesión en la que se haya establecido `CleanSession` en `false`, se restaurará una "sesión persistente". El daemon restaura las suscripciones cuando se reinicia, y se empiezan a recibir las publicaciones. El cliente recibe las publicaciones cuando se reinicia y se ha establecido `CleanSession` en `false`. De forma predeterminada, el estado de la sesión del cliente no se guarda cuando un daemon se detiene y, por tanto, no se restauran las suscripciones, aunque el cliente establezca `CleanSession` en `false`.

`Retained_persistence` es un mecanismo de guardado automático. Es posible que no guarde las publicaciones retenidas o las suscripciones más recientes. Puede cambiar la frecuencia con que se guardan las publicaciones retenidas y las suscripciones. Establezca el intervalo entre operaciones de guardado, o el número de cambios entre operaciones de guardado, utilizando las opciones de configuración `autosave_on_changes` y `autosave_interval`.

Configuración de ejemplo para establecer la persistencia

```
# Sample configuration
# Daemon listens on port 1882 with persistence in /tmp
# Autosave every minute
port 1882
persistence_location /tmp/
retained_persistence true
autosave_on_changes false
autosave_interval 60
```

Seguridad del daemon de WebSphere MQ Telemetry para dispositivos

El daemon de WebSphere MQ Telemetry para dispositivos puede autenticar clientes que se conecten al mismo, utilizar credenciales para conectarse a otros intermediarios, y controlar los accesos a los temas. La seguridad del daemon es limitada por el hecho de que se crea utilizando el cliente C de WebSphere MQ Telemetry, que no proporciona soporte SSL. Por consiguiente, puede que las conexiones a y desde el daemon no estén cifradas, y no se puedan autenticar utilizando los certificados.

De forma predeterminada, no hay activado ningún tipo de seguridad.

Autenticación de clientes

Los clientes MQTT pueden establecer un nombre de usuario y una contraseña utilizando los métodos `MqttConnectOptions.setUserName` y `MqttConnectOptions.setPassword`.

Autentique a un cliente que se conecta al daemon comprobando el nombre de usuario y la contraseña que un cliente proporciona respecto a las entradas del archivo de contraseñas. Para habilitar la autenticación, cree un archivo de contraseñas y establezca el parámetro `password_file` en el archivo de configuración del daemon; consulte password_file.

Establezca el parámetro `allow_anonymous` en el archivo de configuración del daemon para permitir a los clientes que se conectan sin nombre de usuario ni contraseña se conecten a un daemon que esté comprobando la autenticación; consulte allow_anonymous. Si un cliente proporciona un nombre de usuario o una contraseña, se comprueban siempre respecto al archivo de contraseñas, si se ha establecido el parámetro `password_file`.

Establezca el parámetro `clientid_prefixes` en el archivo de configuración del daemon para limitar las conexiones a clientes específicos. Los clientes deben tener `clientIdentifiers` que empiecen con uno de los prefijos indicados n el parámetro `clientid_prefixes`; consulte clientid_prefixes.

Seguridad de conexión de puente

Cada daemon de WebSphere MQ Telemetry para la conexión de puentes de dispositivos es un cliente MQTT V3. Puede establecer el nombre de usuario y la contraseña para cada conexión de puente como un parámetro de conexión de puente en el archivo de configuración del daemon; consulte username y password. Un puente puede, por tanto, autenticarse a sí mismo ante un intermediario.

Control de accesos de temas

Si los clientes se autentican, el daemon también puede proporcionar el control de acceso a temas para cada usuario. El daemon otorga el control de accesos en función de si existe coincidencia con el tema en el que un cliente publica o al que se suscribe con una serie de tema en el archivo de control de acceso; consulte acl_file.

La lista de control de accesos tiene dos partes. La primera parte controla los accesos de todos los clientes, incluidos los clientes anónimos. La segunda parte tiene una sección para cualquier usuario en el archivo de contraseñas. Lista el control de accesos específico para cada usuario.

Ejemplo

En el siguiente ejemplo se muestran los parámetros de seguridad.

```
acl_file c:\WMQTDaemon\config\acl.txt
password_file c:\WMQTDaemon\config\passwords.txt
allow_anonymous true
connection Daemon1
username daemon1
password deamonpassword
```

Figura 43. Archivo de configuración del daemon

```
Fred:Fredpassword
Barney:Barneypassword
```

Figura 44. Archivo de contraseñas, passwords.txt

```
topic home/public/#
topic read meters/#
user Fred
topic write meters/fred
topic home/fred/#
user Barney
topic write meters/barney
topic home/barney/#
```

Figura 45. Archivo de control de accesos, acl.txt

Administración de multidifusión

Utilice esta información para obtener información sobre las tareas de administración de WebSphere MQ Multicast como, por ejemplo, reducir el tamaño de mensajes de multidifusión y habilitar la conversión de datos.

Iniciación a la multidifusión

Utilice esta información para iniciarse en los temas y objetos de información de comunicación de WebSphere MQ.

Acerca de esta tarea

La mensajería de WebSphere MQ Multicast utiliza la red para entregar mensajes mediante la correlación de temas con direcciones de grupo. Las tareas siguientes son una forma rápida de probar si la dirección IP y puerto necesarios están configurados correctamente para la mensajería de multidifusión.

Creación de un objeto COMMINFO para multidifusión

El objeto de información de comunicación (COMMINFO) contiene los atributos asociados con la transmisión de multidifusión. Para obtener más información sobre los parámetros de objeto COMMINFO, consulte DEFINE COMMINFO.

Utilice el siguiente ejemplo de línea de mandatos para definir un objeto COMMINFO para multidifusión:

```
DEFINE COMMINFO(MC1) GRPADDR(group address) PORT(port number)
```

donde *MC1* es el nombre del objeto COMMINFO, *dirección de grupo* es la dirección IP o el nombre DNS de multidifusión de grupo y *número de puerto*es el puerto en el que se va a transmitir (el valor predeterminado es 1414).

Se crea un nuevo objeto COMMINFO denominado *MC1*; este nombre es el que debe especificar al definir un objeto TOPIC en el ejemplo siguiente.

Creación de un objeto TOPIC para multidifusión

Un tema es el asunto de la información que se publica en un mensaje de publicación/suscripción y un tema se define creando un objeto TOPIC. Los objetos TOPIC tienen dos parámetros que definen si pueden utilizarse con multidifusión o no. Estos parámetros son: **COMMINFO** y **MCAST**.

- **COMMINFO** Este parámetro especifica el nombre del objeto de información de comunicación de multidifusión. Para obtener más información sobre los parámetros de objeto COMMINFO, consulte DEFINE COMMINFO.
- **MCAST** Este parámetro especifica si la multidifusión está permitida en esta posición del árbol de temas.

Utilice el siguiente ejemplo de línea de mandatos para definir un objeto TOPIC para multidifusión:

```
DEFINE TOPIC(ALLSPORTS) TOPICSTR('Sports') COMMINFO(MC1) MCAST(ENABLED)
```

Se crea un nuevo objeto TOPIC denominado *ALLSPORTS*. Tiene una serie de tema *Sports*, su objeto de información de comunicación relacionado se denomina *MC1*(que es el nombre que ha especificado al definir un objeto COMMINFO en el ejemplo anterior), y la multidifusión está habilitada.

Prueba de la publicación/suscripción de multidifusión

Después de que se hayan creado los objetos TOPIC y COMMINFO, se pueden probar utilizando el ejemplo `amqspubc` y el ejemplo `amqssubc`. Para obtener más información sobre estos ejemplos, consulte Los programas de ejemplo de publicación/suscripción.

1. Abra dos ventanas de línea de mandatos; la primera línea de mandatos es para el ejemplo de publicación `amqspubc` y la segunda línea de mandatos es para el ejemplo de suscripción `amqssubc`.
2. Entre el siguiente mandato en la línea de mandatos 1:

   ```
   amqspubc Sports QM1
   ```

   donde *Sports* es la serie de tema del objeto TOPIC definido en un ejemplo anterior, y *QM1* es el nombre del gestor de colas.
3. Entre el siguiente mandato en la línea de mandatos 2:

   ```
   amqssubc Sports QM1
   ```

   donde *Sports* y *QM1* son los mismos que se utilizaron en el paso “2” en la página 155.
4. Especifique `Hello world` en la línea de mandatos 1. Si el puerto y la dirección IP especificados en el objeto COMMINFO están configurados correctamente; el ejemplo `amqssubc` , que está a la escucha en el puerto de las publicaciones de la dirección especificada, genera `Hello world` en la línea de mandatos 2.

Topología de temas de IBM WebSphere MQ Multicast

Utilice este ejemplo para entender la topología de temas de IBM WebSphere MQ Multicast.

El soporte de IBM WebSphere MQ Multicast requiere que cada subárbol tenga su propio grupo y corriente de datos de multidifusión dentro de la jerarquía total.

El esquema de direccionamiento IP *classful network* ha designado espacio de direcciones para la dirección de multidifusión. El rango de multidifusión completo de direcciones IP es de `224.0.0.0` a `239.255.255.255`, pero algunas de estas direcciones están reservadas. Para obtener una lista de direcciones reservadas, póngase en contacto con el administrador del sistema o consulte IPv4 Multicast Address Space Registry para obtener más información. Se recomienda utilizar la dirección de multidifusión con ámbito local en el rango de `239.0.0.0` a `239.255.255.255`.

En el diagrama siguiente, hay dos posibles corrientes de datos de multidifusión:

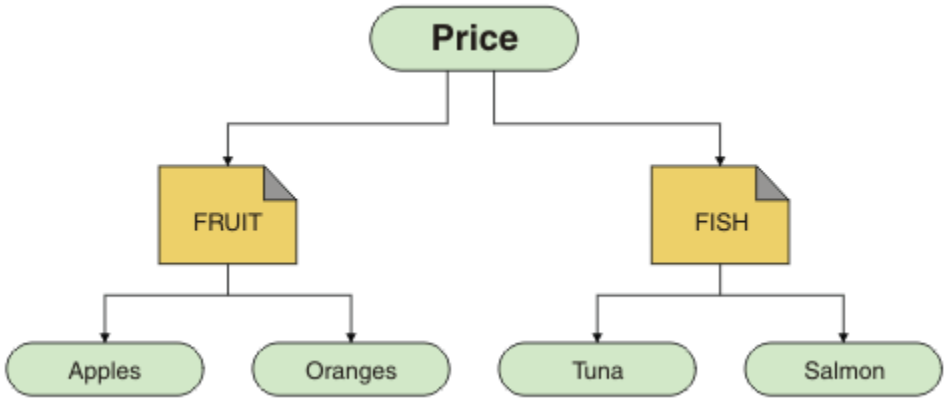
```
DEF COMMINFO(MC1) GRPADDR(239.XXX.XXX.XXX
)
```

```
DEF COMMINFO(MC2) GRPADDR(239.YYY.YYY.YYY)
```

donde *239.XXX.XXX.XXX* y *239.YYY.YYY.YYY* son direcciones de multidifusión válidas.

Estas definiciones de temas se utilizan para crear un árbol de temas tal como se muestra en el diagrama siguiente:

```
DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)
DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



Cada objeto de información de comunicación de multidifusión (COMMINFO) representa una corriente de datos diferente debido a que las direcciones de grupo son diferentes. En este ejemplo, el tema FRUIT está definido para utilizar el objeto COMMINFO MC1, el tema FISH está definido para utilizar el objeto COMMINFO MC2y el nodo Price no tiene definiciones de multidifusión.

WebSphere MQ Multicast tiene un límite de 255 caracteres para las series de tema. Esta limitación significa que hay que tener cuidado con los nombres de nodos y los nodos hoja dentro del árbol; si los nombres de nodos y los nodos hoja son demasiado largos, la serie de tema puede superar los 255 caracteres y devolver el código de razón de 2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR. Es recomendable que las series de tema sean lo más cortas posible porque unas series de tema más largas pueden tener un efecto perjudicial en el rendimiento.

Control del tamaño de mensajes de multidifusión

Utilice esta información para obtener información sobre el formato de mensajes de WebSphere MQ y reduzca el tamaño de los mensajes de WebSphere MQ.

Los mensajes de WebSphere MQ tienen varios atributos asociados con ellos que están contenidos en el descriptor de mensaje. Para los mensajes pequeños, estos atributos pueden representar la mayor parte del tráfico de datos y pueden tener un importante efecto perjudicial sobre la velocidad de transmisión. WebSphere MQ Multicast permite al usuario configurar cuáles de estos atributos, si existen, se transmiten junto con el mensaje.

La presencia de atributos de mensaje, distintos de la serie de tema, depende de si el objeto COMMINFO indica que deben enviarse o no. Si un atributo no se transmite, la aplicación receptora aplica un valor predeterminado. Los valores MQMD predeterminados no son necesariamente los mismos que el valor MQMD_DEFAULT y se describen en Tabla 9 en la página 157.

El objeto COMMINFO contiene el atributo MCPROP, que controla cuántos campos MQMD y propiedades de usuario fluyen con el mensaje. Al establecer el valor de este atributo en un nivel apropiado, puede controlar el tamaño de los mensajes de WebSphere MQ Multicast:

MCPROP
Las propiedades multidifusión controla cuántas de las propiedades MQMD y de las propiedades de usuario fluyen con el mensaje.

ALL
Todas las propiedades de usuario y todos los campos de MQMD se transmiten.

REPLY
Sólo se transmiten las propiedades de usuario y los campos MQMD que están relacionados con la respuesta a los mensajes. Estas propiedades son:

- MsgType
- MessageId
- CorrelId
- ReplyToQ
- ReplyToQmgr

USER
Sólo se transmiten las propiedades de usuario

NINGUNO
No se transmiten las propiedades de usuario ni los campos MQMD

COMPAT
Este valor hace que la transmisión del mensaje se realice en una modalidad compatible para RMM, que permite determinada interoperación con las aplicaciones XMS actuales y las aplicaciones WebSphere Message Broker RMM.

Atributos de los mensajes de multidifusión

Los atributos de mensaje puede proceder de varios lugares, como MQMD, los campos de MQRFH2 y propiedades de mensaje.

La tabla siguiente muestra lo que sucede cuando los mensajes se envían sujetos al valor de MCPROP y el valor predeterminado utilizado cuando no se envía un atributo.

Tabla 9. Atributos de mensajería y cómo se relacionan con la multidifusión

Atributo	Acción cuando se utiliza multidifusión	Valor predeterminado si no se transmite
TopicString	Siempre incluido	No aplicable
MQMQ StrucId	No transmitido	No aplicable
MQMD Version	No transmitido	No aplicable
Informe	Incluido si no es predeterminado	0
MsgType	Incluido si no es predeterminado	MQMT_DATAGRAM
Caducidad	Incluido si no es predeterminado	0
Comentarios	Incluido si no es predeterminado	0
Codificación	Incluido si no es predeterminado	MQENC_NORMAL(equiv)
CodedCharSetId	Incluido si no es predeterminado	1208
Formato	Incluido si no es predeterminado	MQRFH2
Priority	Incluido si no es predeterminado	4
Persistence	Incluido si no es predeterminado	MQPER_NOT_PERSISTENT
MsgId	Incluido si no es predeterminado	Null
CorrelId	Incluido si no es predeterminado	Null
BackoutCount	Incluido si no es predeterminado	0

Tabla 9. Atributos de mensajería y cómo se relacionan con la multidifusión (continuación)		
Atributo	**Acción cuando se utiliza multidifusión**	**Valor predeterminado si no se transmite**
ReplyToQ	Incluido si no es predeterminado	Espacio en blanco
GestorColasRespuesta	Incluido si no es predeterminado	Espacio en blanco
UserIdentifier	Incluido si no es predeterminado	Espacio en blanco
AccountingToken	Incluido si no es predeterminado	Null
PutAppIType	Incluido si no es predeterminado	MQAT_JAVA
PutAppIName	Incluido si no es predeterminado	Espacio en blanco
PutDate	Incluido si no es predeterminado	Espacio en blanco
PutTime	Incluido si no es predeterminado	Espacio en blanco
ApplOriginData	Incluido si no es predeterminado	Espacio en blanco
GroupID	Excluidos	No aplicable
MsgSeqNumber	Excluidos	No aplicable
Desplazamiento	Excluidos	No aplicable
MsgFlags	Excluidos	No aplicable
OriginalLength	Excluidos	No aplicable
UserProperties	Incluidos	No aplicable

Referencia relacionada
ALTER COMMINFO
DEFINE COMMINFO

Habilitación de la conversión de datos para la mensajería de Multicast

Utilice esta información para entender cómo funciona la conversión de datos para la mensajería de WebSphere MQ Multicast.

WebSphere MQ Multicast es un protocolo compartido sin conexión y, por tanto, no es posible que cada cliente realice solicitudes específicas para la conversión de datos. Todos los clientes suscritos a la misma secuencia de multidifusión reciben los mismos datos binarios; por lo tanto, si es necesaria la conversión de datos de WebSphere MQ, esta se realiza localmente en cada cliente.

En una instalación de plataforma mixta, es posible que la mayoría de los clientes requieran los datos en un formato que no sea el formato nativo de la aplicación transmisora. En esta situación, los valores de **CCSID** y **ENCODING** del objeto COMMINFO de multidifusión se pueden utilizar para definir la codificación de la transmisión de mensajes para conseguir eficiencia.

WebSphere MQ Multicast da soporte a la conversión de datos de la carga útil de mensaje para los siguientes formatos incorporados:

- MQADMIN
- MQEVENT
- MQPCF
- MQRFH
- MQRFH2
- MQSTR

Además de estos formatos, también puede definir sus propios formatos y utilizar una salida de conversión de datos de MQDXP: Parámetro de salida de conversión de datos.

Si desea información sobre la programación de conversiones de datos, consulte Conversión de datos en MQI para la mensajería de multidifusión.

Para obtener más información sobre la conversión de datos, consulte Conversión de datos.

Si desea más información sobre las salidas de la conversión de datos y `ClientExitPath`, consulte Stanza ClientExitPath del archivo de configuración de cliente.

Supervisión de aplicaciones de multidifusión

Utilice esta información para aprender a administrar y supervisar WebSphere MQ Multicast.

El estado de los publicadores y suscriptores actuales para el tráfico de multidifusión (por ejemplo, el número de mensajes enviados y recibidos, o el número de mensajes perdidos) se transmite periódicamente al servidor desde el cliente. Cuando se recibe el estado, el atributo COMMEV del objeto COMMINFO especifica si el gestor de colas transmite o no un mensaje de suceso en SYSTEM.ADMIN.PUBSUB.EVENT. El mensaje de suceso contiene la información de estado recibida. Esta información es una inestimable ayuda de diagnóstico para buscar el origen de un problema.

Utilice el mandato MQSC **DISPLAY CONN** para visualizar información de conexión sobre las aplicaciones conectadas al gestor de colas. Para obtener más información sobre el mandato **DISPLAY CONN**, consulte DISPLAY CONN.

Utilice el mandato MQSC **DISPLAY TPSTATUS** para visualizar el estado de los publicadores y suscriptores. Para obtener más información sobre el mandato **DISPLAY TPSTATUS** , consulte DISPLAY TPSTATUS.

COMMEV y el indicador de la fiabilidad de mensajes de multidifusión

El *indicador de fiabilidad*, utilizado conjuntamente con el atributo **COMMEV** del objeto COMMINFO, es un elemento clave de la supervisión de los publicadores y suscriptores de WebSphere MQ Multicast. El indicador de fiabilidad (el campo **MSGREL** que se devuelve en los mandatos de estado Publish o Subscribe) es un indicador de WebSphere MQ que ilustra el porcentaje de transmisiones que no tienen errores. A veces es preciso retransmitir mensajes debido a un error de transmisión, lo que se refleja en el valor de **MSGREL**. Las causas potenciales de errores de transmisión incluyen los suscriptores lentos, redes ocupadas e interrupciones de la red. **COMMEV** controla si se generan mensajes de suceso para manejadores de multidifusión que se crean mediante el objeto COMMINFO y se establece en uno de tres valores posibles:

DISABLED
Los mensajes de suceso no se graban.

ENABLED
Los mensajes de suceso se graban siempre, con una frecuencia definida en el parámetro COMMINFO **MONINT**.

EXCEPTION
Se graban mensajes de sucesos si la fiabilidad del mensaje está por debajo del umbral de fiabilidad. Un nivel de fiabilidad de mensaje de 90% o menos indica que puede existir un problema con la configuración de red, o que una o varias de las aplicaciones de publicación/suscripción se ejecuta demasiado lentamente:

- El valor **MSGREL(100,100)** indica que no ha habido problemas en el período de tiempo de corto plazo o largo plazo.
- El valor **MSGREL(80,60)** indica que el 20% de los mensajes tienen problemas actualmente, pero también una mejora del valor a largo plazo de 60.

Los clientes pueden continuar transmitiendo y recibiendo tráfico de multidifusión incluso cuando se interrumpe la conexión de difusión única para el gestor de colas, por lo tanto, los datos pueden estar desfasados.

Fiabilidad de los mensajes de multidifusión

Utilice esta información para aprender a establecer la suscripción y el historial de mensajes de WebSphere MQ Multicast.

Un elemento clave para superar una anomalía de transmisión con la multidifusión es el almacenamiento intermedio de datos transmitidos de WebSphere MQ (un historial de mensajes para mantenerse en el extremo transmisor del enlace). Este proceso significa que ningún almacenamiento intermedio de mensajes es necesario en el proceso de la aplicación transmisora debido a que WebSphere MQ proporciona la fiabilidad. El tamaño de este historial se configura mediante el objeto de información de comunicación (COMMINFO), tal como se describe en la siguiente información. Un almacenamiento intermedio de transmisión mayor significa que hay más historial de transmisión que retransmitir si es necesario, pero debido a la naturaleza de la multidifusión, no se puede dar soporte a una garantía del 100% de entrega.

El historial de mensajes de WebSphere MQ Multicast se controla en el objeto de información de comunicación (COMMINFO) mediante el atributo **MSGHIST**:

MSGHIST
Este valor es la cantidad de historial de mensajes en kilobytes que mantiene el sistema para manejar las retransmisiones en el caso de acuses de recibo negativos (NACK).

El valor `0` ofrece el nivel mínimo de fiabilidad. El valor predeterminado es `100` KB.

El nuevo historial de suscripciones de WebSphere MQ Multicast se controla en el objeto de información de comunicación (COMMINFO) mediante el atributo **NSUBHIST**:

NSUBHIST
El historial de nuevas suscripciones controla si un suscriptor que se suscribe a una corriente de datos de publicación recibe todos los datos disponibles en este momento, o bien recibe únicamente publicaciones desde el momento de la suscripción.

NINGUNO
El valor `NONE`hace que el transmisor transmita sólo las publicaciones realizadas desde el momento de la suscripción. `NONE` es el valor predeterminado.

ALL
Un valor `ALL` hace que el transmisor retransmita todo el historial del tema que sea conocido. En algunas circunstancias, esta situación puede proporcionar un comportamiento similar a las publicaciones retenidas.

Nota: La utilización del valor `ALL` puede tener un efecto perjudicial en el rendimiento si hay un historial de tema extenso, ya que se retransmite todo el historial del tema.

Referencia relacionada
DEFINE COMMINFO
ALTER COMMINFO

Tareas avanzadas de multidifusión

Utilice esta información para obtener información sobre tareas avanzadas de administración de WebSphere MQ Multicast como, por ejemplo, configurar archivos `.ini` y la interoperatividad con WebSphere MQ LLM.

Para obtener consideraciones sobre la seguridad en una instalación de multidifusión, consulte Seguridad de multidifusión .

Puente entre dominios de publicación/suscripción de multidifusión y no de multidifusión

Use esta información para entender lo que sucede cuando un publicador no de multidifusión publica en un tema habilitado para WebSphere MQ Multicast.

Si un publicador no de multidifusión publica en un tema definido como habilitado para **MCAST** y para **BRIDGE**, el gestor de colas transmite el mensaje a través de multidifusión directamente a los suscriptores que puedan estar escuchando. Un publicador de multidifusión no puede publicar en los temas que no están habilitados para la multidifusión.

Los temas existentes pueden estar habilitados para multidifusión si se establecen los parámetros **MCAST**y **COMMINFO** de un objeto de tema. Consulte Conceptos de multidifusión iniciales para obtener más información sobre estos parámetros.

El atributo **BRIDGE** del objeto COMMINFO controla las publicaciones de publicaciones que no utilizan multidifusión. Si **BRIDGE** se establece en ENABLED y el parámetro **MCAST** del tema también se establece en ENABLED, las publicaciones de las aplicaciones que no están utilizando multidifusión se usan como puente con las aplicaciones que sí. Para obtener más información sobre el parámetro **BRIDGE**, consulte DEFINE COMMINFO.

Configuración de los archivos .ini para Multicast

Utilice esta información para entender los campos de WebSphere MQ Multicast en los archivos `.ini`.

La configuración adicional de WebSphere MQ Multicast puede realizarse en un archivo `ini`. El archivo `ini` específico que debe utilizar depende del tipo de aplicaciones:

- Cliente: configure el archivo *MQ_DATA_PATH*`/mqclient.ini`.
- Gestor de colas: configure el archivo *MQ_DATA_PATH*`/qmgrs/`*QMNAME*`/qm.ini`.

donde *MQ_DATA_PATH* es la ubicación del directorio de datos de WebSphere MQ (`/var/mqm/mqclient.ini`) y *QMNAME* es el nombre del gestor de colas al que se aplica el archivo `.ini`.

El archivo `.ini` contiene campos utilizados para ajustar el comportamiento de WebSphere MQ Multicast:

```
Multicast:
   Protocol              = IP | UDP
   IPVersion             = IPV4 | IPV6 | ANY | BOTH
   LimitTransRate        = DISABLED | STATIC | DYNAMIC
   TransRateLimit        = 100000
   SocketTTL             = 1
   Batch                 = NO
   Loop                  = 1
   Interface             = <IPaddress>
   FeedbackMode          = ACK | NACK | WAIT1
   HeartbeatTimeout      = 20000
   HeartbeatInterval     = 2000
```

Protocolo

UDP
En esta modalidad, los paquetes se envían utilizando el protocolo UDP. Los elementos de la red no pueden proporcionar asistencia en la distribución de multidifusión como lo hacen en la modalidad IP. El formato de paquete sigue siendo compatible con PGM. Este es el valor predeterminado.

IP
En esta modalidad, el transmisor envía paquetes IP sin formato. Los elementos de red con soporte PGM ayudan en la distribución de paquetes de multidifusión fiables. Esta modalidad es totalmente compatible con el estándar PGM.

IPVersion

IPV4
Comunicarse utilizando sólo el protocolo IPv4. Este es el valor predeterminado.

IPV6
Comunicarse utilizando sólo el protocolo IPv6.

CUALQUIERA
Comunicarse utilizando IPv4, IPv6, o ambos, dependiendo de qué protocolo está disponible.

BOTH
Admite la comunicación utilizando tanto IPv4 como IPv6.

LimitTransRate

DISABLED
No hay ningún control de velocidad de transmisión. Este es el valor predeterminado.

STATIC
Implementa el control de velocidad de transmisión estático. El transmisor no podría transmitir a una velocidad que supere la especificada por el parámetro TransRateLimit.

DYNAMIC
El transmisor adapta su velocidad de transmisión de acuerdo con los comentarios que obtiene de los receptores. En este caso, el límite de la velocidad de transmisión no puede ser mayor que el valor especificado por el parámetro TransRateLimit. El transmisor intenta alcanzar una velocidad de transmisión óptima.

TransRateLimit
Límite de la velocidad de transmisión en Kbps.

SocketTTL
El valor de SocketTTL determina si el tráfico de multidifusión puede pasar a través de un direccionador, o el número de direccionadores por los que puede pasar a través.

Lote
Controla si los mensajes se envían por lotes o inmediatamente; hay 2 valores posibles:

- *NO* Los mensajes no se envían por lotes, se envían inmediatamente.
- *YES* Los mensajes se envían por lotes.

Loop
Establezca el valor en 1 para habilitar el bucle de multidifusión. El bucle de multidifusión define si los datos enviados recorren un bucle de retorno al host o no.

Interfaz
Dirección IP de la interfaz en la que fluye el tráfico de multidifusión. Para obtener más información y solucionar problemas, consulte: Prueba de aplicaciones de multidifusión en una red de no multidifusión y Establecimiento de la red adecuada para el tráfico de multidifusión

FeedbackMode

NACK
Comentarios por acuses de recibo negativos. Este es el valor predeterminado.

ACK
Comentarios por acuses de recibo positivos.

WAIT1
Comentarios por acuses de recibo positivos cuando el transmisor sólo espera 1 ACK de cualquiera de los receptores.

HeartbeatTimeout
Tiempo de espera de pulsaciones en milisegundos. Un valor de 0 indica que los sucesos de tiempo de espera de pulsaciones no están planteadas por el receptor o los receptores del tema. El valor predeterminado es 20000.

HeartbeatInterval
Intervalo de pulsaciones en milisegundos. Un valor de 0 indica que no se envían pulsaciones. El intervalo de pulsaciones debe ser considerablemente menor que el valor **HeartbeatTimeout** para evitar sucesos de tiempo de espera de pulsaciones falsos. El valor predeterminado es 2000.

Interoperatividad de multidifusión con WebSphere MQ Low Latency Messaging

Utilice esta información para comprender la interoperatividad entre WebSphere MQ Multicast y WebSphere MQ Low Latency Messaging (LLM).

La transferencia de carga útil básica es posible para una aplicación que utilice LLM, con otra aplicación que utilice la multidifusión para intercambiar mensajes en ambas direcciones. Aunque la multidifusión

utiliza la tecnología LLM, el producto LLM en sí no está incorporado. Por lo tanto, es posible instalar LLM y WebSphere MQ Multicast y operar y dar servicio a los dos productos por separado.

Las aplicaciones LLM que se comunican con la multidifusión quizá tengan que enviar y recibir propiedades de mensaje. Las propiedades de mensaje de WebSphere MQ y los campos de MQMD se transmiten como propiedades de mensaje de LLM con códigos de propiedad de mensaje LLM específicos tal como se muestra en la tabla siguiente:

Tabla 10. Propiedades de mensajes de WebSphere MQ para las correlaciones de propiedades de WebSphere MQ LLM

Propiedad de WebSphere MQ	Tipo de propiedad de WebSphere MQ LLM	Clase de propiedad de LLM	Código de propiedad de LLM
MQMD.Report	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1001
MQMD.MsgType	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1002
MQMD.Expiry	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1003
MQMD.Feedback	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1004
MQMD.Encoding	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1005
MQMD.CodedCharSetId	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1006
MQMD.Format	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1007
MQMD.Priority	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1008
MQMD.Persistence	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1009
MQMD.MsgId	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_ByteArray	-1010
MQMD.BackoutCount	RMM_MSG_PROP_INT32	LLM_PROP_KIND_Int32	-1012
MQMD.ReplyToQ	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1013
MQMD.ReplyToQMger	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1014
MQMD.PutDate	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1020
MQMD.PutTime	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1021
MQMD.ApplOriginData	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1022
MQPubOptions	RMM_MSG_PROP_INT32	LLM_PROP_KIND_int32	-1053

Para obtener más información sobre LLM, consulte la documentación del producto de LLM: WebSphere MQ Low Latency Messaging.

Administración de HP Integrity NonStop Server

Utilice esta información para obtener información sobre las tareas de administración para el cliente de IBM WebSphere MQ para HP Integrity NonStop Server.

Tiene a su disposición dos tareas de administración:

1. El inicio manual de TMF/Gateway desde Pathway.
2. La detención de TMF/Gateway desde Pathway.

Inicial manual de TMF/Gateway desde Pathway

Puede permitir a Pathway iniciar automáticamente TMF/Gateway en la primera solicitud de inscripción, o puede iniciar manualmente TMF/Gateway desde Pathway.

Procedimiento

Para iniciar manualmente TMF/Gateway desde Pathway, entre el siguiente mandato PATHCOM:

```
START SERVER <server_class_name>
```

Si una aplicación cliente realiza una solicitud de inscripción antes de que TMF/Gateway complete la recuperación de transacciones pendientes, la solicitud se retiene durante 1 segundo como máximo. Si la recuperación no se completa en ese tiempo, la inscripción se rechaza. A continuación, el cliente recibe un error MQRC_UOW_ENLISTMENT_ERROR del uso de una MQI transaccional.

Detención de TMF/Gateway desde Pathway

Este tarea describe cómo detener TMF/Gateway desde Pathway y cómo reciniciarlo después de detenerlo.

Procedimiento

1. Para evitar que se realicen nuevas solicitudes de inscripción a TMF/Gateway, entre el mandato siguiente:

   ```
   FREEZE SERVER <server_class_name>
   ```

2. Para desencadenar TMF/Gateway para que complete las operaciones en curso y para que termine, entre el mandato siguiente:

   ```
   STOP SERVER <server_class_name>
   ```

3. Para permitir a TMF/Gateway reiniciarse automáticamente después de la primera inscripción o manualmente, siguiendo los pasos 1 y 2, entre el mandato siguiente:

   ```
   THAW SERVER <server_class_name>
   ```

   Se impide a las aplicaciones que realicen nuevas solicitudes de inscripción y no es posible emitir el mandato **START** hasta que emita el mandato **THAW**.

Avisos

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 disponibles actualmente en su zona. Las referencias a programas, productos o servicios de IBM no pretenden indicar ni implicar que sólo puedan utilizarse los 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 la operación de cualquier producto, programa o servicio ajeno a 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 EE.UU.

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:

Licencia de propiedad intelectual Legal y Ley de propiedad intelectual IBM Japan, Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 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, COMERCIABILIDAD 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 efectuar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento y sin previo aviso.

Cualquier referencia en esta información a sitios web que no son de IBM se realiza por razones prácticas y de ninguna manera sirve como un respaldo de dichos sitios web. Los materiales de dichos sitios web no forman parte de este producto de IBM y la utilización de los mismos será por 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 respeto 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, Departamento 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, algunas mediciones pueden haberse estimado por extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a su entorno específico.

La información relacionada con productos que no sean de IBM se ha obtenido de los distribuidores de dichos productos, de sus anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha comprobado estos productos y no puede confirmar la precisión de su rendimiento, compatibilidad ni contemplar ninguna otra reclamación relacionada con los productos que no son de IBM. Las preguntas relativas a las posibilidades de los productos que no son de IBM deberán dirigirse a los proveedores de dichos productos.

Todas las declaraciones referentes a acciones e intenciones futuras de IBM pueden cambiar o ser retiradas sin aviso previo y solamente representan 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 COPYRIGHT:

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 de interfaz de programación, si se proporciona, está pensada para ayudarle a crear software de aplicación para su uso con este programa.

Este manual contiene información sobre las interfaces de programación previstas que permiten al cliente escribir programas para obtener los servicios de IBM WebSphere 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 porque está sujeta a cambios.

Marcas registradas

IBM, el logotipo de IBM , ibm.com, son marcas registradas de IBM Corporation, registradas en muchas jurisdicciones de todo el mundo. Hay disponible una lista actual de marcas registradas de IBM en la web en "Copyright and trademark information"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 EE.UU. y/o en 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 Eclipse Project (http://www.eclipse.org/).

Java y todas las marcas registradas y logotipos son marcas registradas de Oracle o sus afiliados.

Número Pieza:

(1P) P/N: