7.5

Administrando o IBM WebSphere MQ

Nota

Antes de usar estas informações e o produto que elas suportam, leia as informações em “Avisos” na página 165.

Esta edição se aplica à versão 7 liberação 5 do IBM® WebSphere MQ e a todas as liberações e modificações subsequentes até que seja indicado de outra forma em novas edições.

Ao enviar informações para a IBM, você concede à IBM um direito não exclusivo de usar ou distribuir as informações da maneira que julgar apropriada, sem incorrer em qualquer obrigação para com você

Índice

Administrando IBM WebSphere MQ

Administrar os gerenciadores de filas e os recursos associados inclui as tarefas que você frequentemente executa para ativar e gerenciar esses recursos. Escolha o método que prefere para administrar os gerenciadores de filas e recursos associados.

É possível administrar os objetos IBM WebSphere MQ localmente ou remotamente, consulte “Administração remota e local” na página 7.

Existem inúmeros métodos diferentes que é possível usar para criar e administrar os seus gerenciadores de filas e seus recursos relacionados em IBM WebSphere MQ. Esses métodos incluem as interfaces da linha de comandos, uma interface gráfica com o usuário e uma API de administração. Consulte as seções e os links neste tópico para obter informações adicionais sobre cada uma dessas interfaces.

Existem diferentes conjuntos de comandos que podem ser usados para administrar o IBM WebSphere MQ dependendo da sua plataforma:

- “IBM WebSphere MQ comandos de controle” na página 5
- “IBM WebSphere MQ Comandos do Script (MQSC)” na página 5
- “Formatos de Comandos Programáveis (PCFs)” na página 6

Também há outras seguintes opções para criar e gerenciar os objetos do IBM WebSphere MQ:

- “O IBM WebSphere MQ Explorer” na página 6
- “O Aplicativo de Configuração Padrão do Windows” na página 7
- “O Microsoft Cluster Service (MSCS)” na página 7

É possível automatizar algumas tarefas de administração e monitoramento para gerenciadores de filas locais e remotas usando comandos PCF. Estes comandos também podem ser simplificados por meio do uso do IBM WebSphere MQ Administration Interface (MQAI) em algumas plataformas. Para obter informações adicionais sobre a automatização das tarefas de administração, consulte “Automatizando Tarefas de Administração” na página 8.

IBM WebSphere MQ comandos de controle

Os comandos de controle permitem executar as tarefas administrativas nos próprios gerenciadores de fila.

IBM WebSphere MQ para Windows, UNIX and Linux® sistemas fornecem os *comandos de controle* que você emite na linha de comando do sistema

Os comandos de controle são descritos em Criando e gerenciando gerenciadores de filas. Para obter a referência de comando para os comandos de controle, consulte IBM WebSphere MQ Controle de comandos.

IBM WebSphere MQ Comandos do Script (MQSC)

Use os comandos MQSC para gerenciar os objetos do gerenciador de filas, incluindo o próprio gerenciador de filas, filas, definições de processo, listas, canais, canais de conexão do cliente, listeners, serviços e objetos de informações de autenticação.

Você emite comandos do MQSC para um gerenciador de filas usando o comando `runmqsc`. Isso pode ser feito interativamente, emitindo os comandos a partir do teclado ou é possível redirecionar o dispositivo de entrada padrão (stdin) para executar uma sequência de comandos de um arquivo de texto ASCII. Em ambos os casos, o formato dos comandos é o mesmo.

É possível executar o comando `runmqsc` em três modos, dependendo do conjunto de sinalizadores no comando:

- *Modo de verificação*, em que os comandos MQSC são verificados em um gerenciador de filas locais, mas não são executados
- *Modo direto*, em que os comandos MQSC são executados em um gerenciador de filas locais
- *Modo indireto*, em que os comandos MQSC são executados em um gerenciador de filas remoto

Os atributos de objeto especificados nos comandos MQSC são mostrados nesta seção em maiúsculas (por exemplo, RQMNAME), embora não façam distinção entre maiúsculas e minúsculas. Os nome de atributo do comando MQSC são limitados a oito caracteres.

Os comandos MQSC estão disponíveis em todas as plataformas . Os comandos MQSC são resumidos em Comparando Conjuntos de Comandos.

No Windows, UNIX ou Linux, é possível usar o MQSC como um único comando emitido na linha de comando do sistema. Para emitir diversos comandos ou mais complicados, o MQSC pode ser construído em um arquivo que você executa a partir da linha de comando do sistema Windows, UNIX ou Linux. O MQSC pode ser enviado a um gerenciador de filas remotas. Para obter detalhes integrais, consulte Referência de MQSC.

O “Comandos do Script (MQSC)” na página 75 contém uma descrição de cada comando MQSC e sua sintaxe.

Consulte “Executando tarefas de administração locais usando comandos MQSC” na página 74 para obter informações adicionais sobre o uso de comandos MQSC na administração local.

Formatos de Comandos Programáveis (PCFs)

Os Formatos de Comando Programável (PCFs) definem as mensagens de resposta e comando que podem ser trocadas entre um programa e qualquer gerenciador de filas (que suporta PCFs) em uma rede. É possível usar os comandos PCF em um programa de aplicativo de gerenciamento de sistemas para administração de objetos IBM WebSphere MQ: objetos de informações de autenticação, canais, listeners de canais, listas de nomes, definições de processo, gerenciadores de fila, filas, serviços e classes de armazenamento. O aplicativo pode operar a partir de um único ponto na rede para comunicar informações de resposta e comando com qualquer gerenciador de filas, local ou remoto, usando o gerenciador de filas locais.

Para obter informações adicionais sobre PCFs, consulte “Introdução aos Formatos de Comando Programável” na página 9.

Para definição de PCFs e estruturas para comandos e respostas, consulte Referência de Formatos de Comando Programável.

O IBM WebSphere MQ Explorer

Usando o IBM WebSphere MQ Explorer, é possível executar as seguintes ações:

- Definir e controlar vários recursos incluindo gerenciadores de filas, filas, definições de processo, listas de nomes, canais, canais de conexão do cliente, listeners, serviços e clusters.
- Iniciar ou parar um gerenciador de filas locais e seus processos associados.
- Visualizar os gerenciadores de filas e seus objetos associados em sua estação de trabalho ou de outras estações de trabalho.
- Verificar o status de gerenciadores de filas, clusters e canais.
- Verifique se os aplicativos, usuários ou canais possuem uma fila específica aberta, a partir do status de fila.

Nos sistemas Windows e Linux , é possível iniciar IBM WebSphere MQ Explorer usando o menu do sistema, o arquivo executável `MQExplorer` ou o comando **`strmqcfg`** .

No Linux, para iniciar o IBM WebSphere MQ Explorer com êxito, deve-se conseguir gravar um arquivo para o seu diretório inicial e o diretório inicial deve existir.

É possível usar o IBM WebSphere MQ Explorer para administrar gerenciadores de filas remotas em outras plataformas, incluindo z/OS, para obter detalhes e para fazer download do SupportPac MS0T, consulte https://www.ibm.com/support/docview.wss?uid=swg24021041.

Consulte a “Administração usando o IBM WebSphere MQ Explorer” na página 57 para obter mais informações.

O Aplicativo de Configuração Padrão do Windows

É possível usar o programa de Configuração Padrão do Windows para criar um *starter* (ou conjunto padrão) de objetos IBM WebSphere MQ .. Um resumo dos objetos padrão criados é listado na Tabela 1: Objetos criados pelo aplicativo de configuração padrão do Windows.

O Microsoft Cluster Service (MSCS)

O Microsoft Cluster Service (MSCS) permite conectar servidores em um *cluster*, fornecendo maior disponibilidade de dados e aplicativos e facilitando o gerenciamento do sistema. O MSCS pode automaticamente detectar e recuperar-se das falhas no servidor ou aplicativo.

É importante não confundir os clusters na detecção MSCS com clusters IBM WebSphere MQ. A distinção é:

Clusters IBM WebSphere MQ
São grupos de dois ou mais gerenciadores de filas em um ou mais computadores, fornecendo interconexão automática e permitindo que filas sejam compartilhadas entre eles para balanceamento de carga e redundância.

Clusters do MSCS
Grupos de computadores conectados e configurados de maneira que, se um falhar, o MSCS executa um *failover*, transferindo os dados de estado dos aplicativos do computador com falha para outro computador no cluster e reiniciando sua operação nele.

Suportar o Microsoft Cluster Service (MSCS) fornece informações detalhadas sobre como configurar seu sistema IBM WebSphere MQ para Windows para usar o MSCS.

Conceitos relacionados
Visão geral técnica do WebSphere MQ
“Administrando objetos locais do IBM WebSphere MQ” na página 70
Esta seção informa como administrar objetos locais do IBM WebSphere MQ para suportar programas de aplicativos que utilizam uma MQI (Message Queue Interface). Nesse contexto, a administração local significa criar, exibir, mudar, copiar e excluir os objetos do IBM WebSphere MQ.

“Administrando objetos remotos do IBM WebSphere MQ” na página 107

Considerações quando o contato estiver perdido com o gerenciador de recursos XA
Tarefas relacionadas
Planejamento
Configurar
Referências relacionadas
Cenários de Suporte Transacional

Administração remota e local

É possível administrar objetos do WebSphere MQ localmente ou remotamente.

Local de administração significa executar tarefas de administração em quaisquer gerenciadores de filas que você definiu no seu sistema local. É possível acessar outros sistemas, por exemplo, através do programa de emulação de terminal TCP/IP **telnet** e realizar a administração lá. No WebSphere MQ, é possível considerar isso como uma administração local porque nenhum canal está envolvido, ou seja, a comunicação é gerenciada pelo sistema operacional

O WebSphere MQ suporta a administração de um único ponto de contato por meio do que é conhecido como *administração remota*. Isso permite emitir comandos de seu sistema local que são processados em outro sistema e se aplica também ao WebSphere MQ Explorer. Por exemplo, é possível emitir um comando remoto para mudar uma definição de fila em um gerenciador de filas remotas. Você não precisa efetuar logon nesse sistema, embora você precise ter os canais adequados definidos. O gerenciador de filas e o servidor de comandos no sistema de destino devem estar em execução.

Alguns comandos não podem ser emitidos desta forma, em especial, criar ou iniciar gerenciadores de filas e o comando iniciar servidores. Para executar esse tipo de tarefa, deve-se efetuar logon no sistema remoto e emitir os comandos a partir de lá ou criar um processo que possa emitir os comandos para você. Essa restrição também se aplica ao WebSphere MQ Explorer.

"Administrando objetos remotos do IBM WebSphere MQ" na página 107 descreve o assunto de administração remota em maiores detalhes.

Como usar comandos de controle do IBM WebSphere MQ

Esta seção descreve como usar os comandos de controle do IBM WebSphere MQ.

Se desejar emitir comandos de controle, seu ID do usuário deverá ser um membro do grupo mqm. Para obter mais informações sobre isso, consulte Autoridade para administrar WebSphere MQ em UNIX, Linux e sistemas Windows . Além disso, observe as seguintes informações específicas do ambiente:

WebSphere MQ para Windows
Todos os comandos de controle podem ser emitidos a partir de uma linha de comandos. Os nomes de comandos e seus sinalizadores não fazem distinção entre maiúsculas e minúsculas: é possível digitá-los em maiúsculas, minúsculas ou em uma combinação de maiúsculas e minúsculas. No entanto, os argumentos para comandos de controle (como nomes de filas) fazem distinção entre maiúsculas e minúsculas.

Nas descrições de sintaxe, o hífen (-) é utilizado como um indicador de sinalizador. É possível utilizar a barra (/) em vez do hífen.

WebSphere MQ para sistemas UNIX and Linux
Todos os comandos de controle do WebSphere MQ podem ser emitidos de um shell. Todos os comandos fazem distinção entre maiúsculas e minúsculas.

Um subconjunto dos comandos de controle pode ser emitido usando o IBM WebSphere MQ Explorer.

Para obter mais informações, consulte Os comandos de controle do WebSphere MQ

Automatizando Tarefas de Administração

É possível concluir que será benéfico para sua instalação automatizar algumas tarefas de administração e monitoramento. É possível automatizar tarefas de administração para gerenciadores de filas locais e remotas usando comandos programmable command format (PCF). Esta seção assume que você tem experiência na administração de objetos do WebSphere MQ

comandos PCF

Os comandos de formato de comando programável (PCF) do WebSphere MQ podem ser usados para programar tarefas de administração em um programa de administração. Desta maneira, a partir de um programa é possível manipular os objetos do gerenciador de filas (filas, definições do processo, listas, canais, canais de conexão do cliente, listeners, serviços e objetos das informações de autenticação) e até manipular os próprios gerenciadores de fila.

Os comandos PCF abrangem a mesma faixa de funções fornecidas pelos comandos MQSC. É possível gravar um programa para emitir comandos PCF para qualquer gerenciador de filas na rede a partir de um único nó. Desta maneira, é possível centralizar e automatizar as tarefas de administração.

Cada comando PCF é uma estrutura de dados integrada na parte de dados do aplicativo de uma mensagem do WebSphere MQ . Cada comando é enviado ao gerenciador de filas de destino usando a função MQI MQPUT da mesma maneira que qualquer outra mensagem. Desde que o servidor de

comandos esteja em execução no gerenciador de filas que recebe a mensagem, o servidor de comandos interpreta como uma mensagem de comando e executa o comando. Para obter as respostas, o aplicativo emite uma chamada MQGET e os dados de resposta são retornados em outra estrutura de dados. O aplicativo pode então processar a resposta e agir de acordo.

Nota: Diferente de comandos MQSC, os comandos PCF e suas respostas não estão em um formato de texto que você possa ler.

Resumidamente, estas são algumas das coisas necessárias para criar uma mensagem de comando PCF:

Descritor de Mensagens
Esse é um descritor de mensagens padrão do WebSphere MQ no qual:

- Tipo de mensagem (*MsqType*) é MQMT_REQUEST.
- Formato da mensagem (*Format*) é MQFMT_ADMIN.

Dados do aplicativo
Contém a mensagem PCF que inclui o cabeçalho PCF, em que:

- O tipo de mensagem PCF (*Type*) especifica MQCFT_COMMAND.
- O identificador de comando especifica o comando, por exemplo, *Change Queue* (MQCMD_CHANGE_Q)

Para obter uma descrição completa das estruturas de dados PCF e como implementá-las, consulte "Introdução aos Formatos de Comando Programável" na página 9.

Atributos de Objeto PCF

Os atributos do objeto no PCF não são limitados a oito caracteres como se fossem para comandos MQSC. Eles são mostrados neste guia em itálico. Por exemplo, o PCF equivalente de RQMNAME for *RemoteQMgrName*.

PCFs de Escape

PCFs de escape são comandos PCF que contêm comandos MQSC no texto de mensagem. É possível usar PCFs para enviar comandos para um gerenciador de filas remotas. Para obter mais informações sobre PCFs de escape, consulte Escape.

Introdução aos Formatos de Comando Programável

Os Formatos de Comando Programável (PCFs) definem as mensagens de resposta e comando que podem ser trocadas entre um programa e qualquer gerenciador de filas (que suporta PCFs) em uma rede. Os PCFs simplificam a administração do gerenciador de filas e outra administração de rede. Eles podem ser usados para resolver o problema da administração complexa de redes distribuídas especialmente conforme as redes crescem de tamanho e complexidade.

Os Formatos de Comando Programável descritos na documentação desse produto são suportados por:

- 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

Problemas que os comandos PCF resolvem

A administração de redes distribuídas pode ser complexa. Os problemas de administração continuam crescendo conforme as redes aumentam de tamanho e complexidade.

Exemplos de administração específica para o sistema de mensagens e enfileiramento incluem:

- Gerenciamento de recurso.

  Por exemplo, a criação e exclusão da fila.
- Monitoramento de desempenho.

  Por exemplo, profundidade da fila máxima ou taxa de mensagens.
- Controle.

  Por exemplo, ajustando parâmetros de fila como a profundidade máxima da fila, comprimento máximo de mensagem e ativação e desativação de filas.
- Roteamento de mensagem.

  Definição de rotas alternativas por meio de uma rede.

WebSphere MQ Os comandos PCF podem ser usados para simplificar a administração do gerenciador de fila e outra administração de rede. Os comandos PCF permitem usar um único aplicativo para executar a administração de rede de um único gerenciador de filas na rede.

O que são PCFs?

Os PCFs definem as mensagens de resposta e comando que podem ser trocadas entre um programa e qualquer gerenciador de filas (que suporta PCFs) em uma rede. É possível usar comandos PCF em um programa de aplicativo de gerenciamento de sistemas para administração de objetos do WebSphere MQ : objetos de informações sobre autenticação, canais, listeners de canais, listas de nomes, definições de processos, gerenciadores de fila, filas, serviços e classes de armazenamento.. O aplicativo pode operar a partir de um único ponto na rede para comunicar informações de resposta e comando com qualquer gerenciador de filas, local ou remoto, usando o gerenciador de filas locais.

Cada gerenciador de filas tem uma fila de administração com um nome de fila padrão e seu aplicativo pode enviar mensagens de comando PCF para essa fila. Cada gerenciador de filas também tem um servidor de comandos para atender as mensagens de comando da fila de administração. Portanto, as mensagens de comando PCF podem ser processadas por qualquer gerenciador de filas na rede e os dados de resposta podem ser retornados ao seu aplicativo, usando a sua fila de resposta especificada. As mensagens de resposta e comandos PCF são enviados e recebidos usando a Message Queue Interface (MQI) normal.

Para obter uma lista dos comandos PCF disponíveis, incluindo seus parâmetros, consulte Definições dos formatos de comando programáveis.

Usando formatos de comando programáveis

É possível usar PCFs em um programa de gerenciamento de sistemas para a administração remota do WebSphere MQ .

Esta seção inclui:

- “Mensagens de comando PCF” na página 10
- “Respostas” na página 13
- Regras para nomear objetos IBM WebSphere MQ
- “Verificação de autoridade para comandos PCF” na página 15

Mensagens de comando PCF

Mensagens de comando PCF consistem em um cabeçalho PCF, parâmetros identificados nesse cabeçalho e também em dados da mensagem definidos pelo usuário. As mensagens são emitidas utilizando chamadas de interface de Fila de Mensagens.

Cada comando e seus parâmetros são enviados como uma mensagem de comando separada contendo um cabeçalho PCF seguido por um número de estruturas de parâmetros; para obter detalhes sobre o cabeçalho PCF, consulte MQCFH – cabeçalho PCF e para um exemplo de uma estrutura do parâmetro, consulte MQCFST – sequência do parâmetro PCF. O cabeçalho PCF identifica o comando e o número de

estruturas de parâmetro que seguem na mesma mensagem. Cada estrutura do parâmetro fornece um parâmetro para o comando.

As respostas para os comandos geradas pelo servidor de comandos têm uma estrutura semelhante. Há um cabeçalho PCF, seguido por um número de estruturas de parâmetros. Respostas podem consistir em mais de uma mensagem, mas os comandos sempre consistem em uma única mensagem.

Em plataformas diferentes de z/OS, a fila para a a qual os comandos PCF são enviados é sempre chamada de SYSTEM.ADMIN.COMMAND.QUEUE.

Como emitir mensagens de comando PCF

Utilize as chamadas normais do Message Queue Interface (MQI), MQPUT, MQGET e assim por diante, para colocar e recuperar o comando PCF e as mensagens de resposta para e a partir de suas filas.

Nota:

Certifique-se de que o servidor de comandos esteja em execução no gerenciador de filas de destino para que o comando PCF processe nesse gerenciador de filas.

Para obter uma lista de arquivos de cabeçalho fornecidos, consulte WebSphere MQ COPY, header, include e module files.

Descritor de mensagens para um comando PCF

O descritor de mensagens WebSphere MQ está totalmente documentado em MQMD-Message descriptor.

Uma mensagem de comando PCF contém os seguintes campos no descritor de mensagens:

Report
Qualquer valor válido, conforme necessário.

MsgType
Este campo deve ser MQMT_REQUEST para indicar uma mensagem que requer uma resposta.

Expiry
Qualquer valor válido, conforme necessário.

Feedback
Configure para MQFB_NONE

Encoding
Se você estiver enviando para sistemas Windows, UNIX ou Linux, configure este campo para a codificação utilizada para os dados da mensagem; a conversão é executada, se necessário.

CodedCharSetId
Se você estiver enviando paraPara sistemas Windows, UNIX ou Linux, configure este campo para o identificador de conjunto de caracteres codificado utilizado para os dados da mensagem; a conversão será executada se necessário.

Format
Configure para MQFMT_ADMIN.

Priority
Qualquer valor válido, conforme necessário.

Persistence
Qualquer valor válido, conforme necessário.

MsgId
O aplicativo de envio pode especificar qualquer valor ou MQMI_NONE pode ser especificado para solicitar ao gerenciador de filas que gere um identificador de mensagem exclusivo.

CorrelId
O aplicativo de envio pode especificar qualquer valor ou MQCI_NONE pode ser especificado para indicar nenhum identificador de correlação.

ReplyToQ
O nome da fila para receber a resposta.

ReplyToQMgr
O nome do gerenciador de filas para a resposta (ou em branco).

Campos de contexto da mensagem
Esses campos podem ser configurados para quaisquer valores válidos, conforme necessário. Normalmente a opção MQPMO_DEFAULT_CONTEXT de inserção de mensagem é utilizada para definir os campos de contexto da mensagem para os valores padrão.

Se você estiver utilizando uma estrutura MQMD da versão 2, deve-se configurar os campos adicionais a seguir:

GroupId
Configure para MQGI_NONE

MsgSeqNumber
Configurado para 1

Offset
Configurado para 0

MsgFlags
Configure para MQMF_NONE

OriginalLength
Configure como MQOL_UNDEFINED

Enviando dados do usuário

As estruturas PCF também podem ser utilizadas para enviar dados da mensagem definidos pelo usuário. Neste caso, o campo descritor de mensagens *Format* deve ser configurado como MQFMT_PCF.

Enviando e recebendo mensagens PCF em uma fila especificada

Enviando mensagens PCF para uma fila especificada

Para enviar uma mensagem para uma fila especificada, a chamada mqPutBag converte o conteúdo do pacote especificado em uma mensagem PCF e envia a mensagem para a fila especificada. O conteúdo do pacote é deixado inalterado após a chamada.

Como entrada para essa chamada, deve-se fornecer:

- Um MQI identificador de conexão.
- Um identificador de objeto para a fila na qual a mensagem deve ser colocada.
- Um descritor de mensagens. Para obter mais informações sobre o descritor de mensagens, consulte MQMD – Descritor de Mensagens.
- Opções de Colocação de Mensagens utilizando a estrutura MQPMO. Para obter mais informações sobre a estrutura MQPMO, consulte – MQPMO Put-opções de mensagens.
- A alça do pacote a ser convertidos em uma mensagem.

  Nota: Se o pacote contém uma mensagem de administração e a chamada mqAddInquiry foi utilizada para inserir valores no pacote, o valor do item de dados MQIASY_COMMAND deve ser um comando INQUIRE reconhecido pelo MQAI.

Para obter uma descrição completa da chamada mqPutBag, consulte mqPutBag.

Recebendo mensagens PCF de uma fila especificada

Para receber uma mensagem de uma fila especificada, a chamada mqGetBag obtém uma mensagem PCF a partir de uma fila especificada e converte os dados da mensagem em um pacote de dados.

Como entrada para essa chamada, deve-se fornecer:

- Um MQI identificador de conexão.
- Uma manipulação de objetos da fila a partir da qual a mensagem deve ser lida.
- Um descritor de mensagens. Dentro da estrutura MQMD, o parâmetro `Format` deve ser MQFMT_ADMIN, MQFMT_EVENT ou MQFMT_PCF.

  Nota: Se a mensagem for recebida em uma unidade de trabalho (ou seja, com a opção MQGMO_SYNCPOINT) e a mensagem possuir um formato não suportado, a unidade de trabalho poderá ser desfeita. A mensagem é então restabelecida na fila e pode ser recuperada através da chamada MQGET em vez de a chamada mqGetBag. Para obter informações adicionais sobre o descritor de mensagens, consulte – MQGMO Get-opções de mensagem.
- Opções de obtenção de mensagens utilizando a estrutura MQGMO. Para obter mais informações sobre a estrutura MQGMO, consulte MQMD – Descritor de Mensagens.
- A alça do pacote para conter a mensagem convertida.

Para obter uma descrição completa da chamada mqGetBag, consulte mqGetBag.

Respostas

Em resposta a cada comando, o servidor de comandos gerará uma ou mais mensagens de resposta. Uma mensagem de resposta possui um formato semelhante a uma mensagem de comando.

O cabeçalho PCF possui o mesmo valor identificador de comando que o comando para o qual ele é uma resposta (consulte MQCFH - Cabeçalho PCF para obter detalhes). O identificador de mensagem e o identificador de correlação são configurados de acordo com as opções de relatório da solicitação.

Se o tipo de cabeçalho PCF da mensagem de comando for MQCFT_COMMAND, somente as respostas padrão serão geradas. Esses comandos são suportados em todas as plataformas, exceto z/OS Os aplicativos mais antigos não suportam PCF no z/OS; o WebSphere MQ Windows Explorer é um desses aplicativos (no entanto, a Versão 6.0 ou posterior IBM WebSphere MQ Explorer suporta PCF no z/OS).

Se o tipo de cabeçalho PCF da mensagem de comando é MQCFT_COMMAND_XR, respostas estendidas ou padrão são geradas. Esses comandos são suportados no z/OS e em algumas outras plataformas Os comandos emitidos no z/OS geram apenas respostas estendidas Em outras plataformas, os dois tipos de resposta podem ser gerados.

Se um comando único especifica um nome de objeto genérico, uma resposta separada é retornada em sua própria mensagem para cada objeto correspondente. Para geração de resposta, um comando único com um nome genérico é tratado como vários comandos individuais (exceto para o campo de controle MQCFC_LAST ou MQCFC_NOT_LAST). Caso contrário, uma mensagem de comando gera uma mensagem de resposta.

Algumas respostas PCF podem retornar uma estrutura mesmo quando não solicitada. Essa estrutura é mostrada na definição da resposta (Definições dos Formatos de Comando Programáveis) como *sempre retornado*. O motivo é que, para essas respostas, é necessário nomear os objetos na resposta para identificar a qual objeto os dados se aplicam.

Descritor de mensagens para uma resposta

Uma mensagem de resposta possui os seguintes campos no descritor de mensagens:

MsgType
Este campo é MQMT_REPLY.

MsgId
Este campo é gerado pelo gerenciador de filas.

CorrelId
Este campo é gerado de acordo com as opções de relatório da mensagem de comando.

Format
Este campo é MQFMT_ADMIN.

Encoding
Configure para MQENC_NATIVE.

CodedCharSetId
Configure para MQCCSI_Q_MGR.

Persistence
O mesmo que na mensagem de comando.

Priority
O mesmo que na mensagem de comando.

A resposta é gerada com MQPMO_PASS_IDENTITY_CONTEXT.

Respostas padrão

Mensagens de comando com um tipo de cabeçalho de MQCFT_COMMAND, as respostas padrão são geradas. Esses comandos são suportados em todas as plataformas, exceto z/OS

Há três tipos de resposta padrão:

- Resposta OK
- Resposta de Erro
- Resposta de dados

Resposta OK

Esta resposta consiste em uma mensagem que começa com um cabeçalho em formato de comando, com um campo *CompCode* de MQCC_OK ou MQCC_WARNING.

Para MQCC_OK, o *Reason* é MQRC_NONE.

Para MQCC_WARNING, o *Reason* identifica a natureza do aviso. Neste caso, o cabeçalho de formato de comando pode ser seguido por uma ou mais estruturas de parâmetro de aviso apropriadas para esse código de razão.

Em qualquer caso, para um comando de consulta, mais estruturas de parâmetro podem seguir, conforme descrito nas seções a seguir.

Resposta de Erro

Se o comando tiver um erro, uma ou mais mensagens de resposta de erro são enviadas (mais de uma pode ser enviada mesmo para um comando que normalmente teria somente uma única mensagem de resposta). Essas mensagens de resposta de erro têm MQCFC_LAST ou MQCFC_NOT_LAST configurado conforme apropriado.

Cada mensagem começa com um cabeçalho em formato de resposta, com um valor *CompCode* de MQCC_FAILED e um campo *Reason* que identifica o erro específico. Em geral, cada mensagem descreve um erro diferente. Além disso, cada mensagem tem zero ou uma (nunca mais de uma) estrutura de parâmetro de erro após o cabeçalho. Esta estrutura de parâmetro, se houver uma, é uma estrutura MQCFIN, com um campo *Parameter* que contém um dos seguintes:

- MQIACF_PARAMETER_ID

  O campo *Value* na estrutura é o identificador do parâmetro que estava em erro (por exemplo, MQCA_Q_NAME).
- MQIACF_ERROR_ID

  Esse valor é utilizado com um valor *Reason* (no cabeçalho em formato de comando) de MQRC_UNEXPECTED_ERROR. O campo *Value* na estrutura MQCFIN é o código de razão inesperado recebido pelo servidor de comandos.
- MQIACF_SELECTOR

Esse valor ocorre se uma estrutura de lista (MQCFIL) enviada com o comando contém um seletor duplicado ou um que não é válido. O campo *Reason* no cabeçalho em formato de comando identifica o erro e o campo *Value* na estrutura MQCFIN é o valor do parâmetro na estrutura MQCFIL do comando que estava em erro.

- MQIACF_ERROR_OFFSET

  Esse valor ocorre quando há um erro de comparação de dados no comando Ping Channel. O campo *Value* na estrutura é o deslocamento do erro de comparação Ping Channel.

- MQIA_CODED_CHAR_SET_ID

  Esse valor ocorre quando o identificador de conjunto de caracteres codificado no descritor de mensagens da mensagem de comando PCF recebida não corresponde àquela do gerenciador de filas de destino. O campo *Value* na estrutura é o identificador do conjunto de caracteres codificado do gerenciador de filas.

A última (ou única) mensagem de resposta de erro é uma resposta de resumo, com um campo *CompCode* de MQCC_FAILED e um campo *Reason* de MQRCCF_COMMAND_FAILED. Essa mensagem não possui estrutura de parâmetros após o cabeçalho.

Resposta de dados

Essa resposta consiste em uma resposta OK (conforme descrito anteriormente) para um comando de consulta. A resposta OK é seguida por estruturas adicionais contendo os dados solicitados conforme descrito em Definições dos formatos de comando programáveis.

Aplicativos não devem depender dessas estruturas de parâmetro adicionais que estão sendo retornadas em qualquer ordem particular.

Verificação de autoridade para comandos PCF

Quando um comando PCF é processado, o *UserIdentifier* do descritor de mensagens na mensagem de comandos é usado para as verificações de autoridade de objeto necessárias do WebSphere MQ .. A verificação de autoridade é implementada de forma diferente em cada plataforma, conforme descrito neste tópico.

As verificações são executadas no sistema no qual o comando está sendo processado; portanto, esse ID de usuário deve existir no sistema de destino e possuir as autoridades necessárias para processar o comando. Se a mensagem chegou de um sistema remoto, uma forma de conseguir o ID existente no sistema de destino deve ter um ID do usuário correspondente em ambos os sistemas local e remoto.

IBM WebSphere MQ para sistemas Windows, UNIX and Linux

Para processar qualquer comando do PCF, o ID do usuário deve ter autoridade *dsp* para o objeto de gerenciador de filas no sistema de destino. Além disso, as verificações de autoridade de objeto do WebSphere MQ são executadas para determinados comandos PCF, conforme mostrado em Tabela 1 na página 16

Para processar qualquer um dos seguintes comandos o ID do usuário deve pertencer ao grupo *mqm*.

Nota: Para Windows **apenas**, o ID do usuário pode pertencer ao grupo *Administradores* ou grupo *mqm*.

- Alterar Canal
- Copiar Canal
- Criar Canal
- Excluir Canal
- Executar ping no Canal
- Redefinir Canal
- Iniciar o Canal

- Parar Canal
- Iniciar Iniciador de Canal
- Iniciar Ouvinte de Canal
- Resolver Canal
- Reconfigurar Cluster
- Refresh Cluster
- Suspender Gerenciador de Filas
- Retomar Gerenciador de Filas

WebSphere MQ para HP Integrity NonStop Server

Para processar qualquer comando do PCF, o ID do usuário deve ter autoridade *dsp* para o objeto de gerenciador de filas no sistema de destino. Além disso, as verificações da autoridade de objeto do IBM WebSphere MQ são executadas para determinados comandos PCF, conforme mostrado em Tabela 1 na página 16.

Para processar qualquer um dos seguintes comandos o ID do usuário deve pertencer ao grupo *mqm*:

- Alterar Canal
- Copiar Canal
- Criar Canal
- Excluir Canal
- Executar ping no Canal
- Redefinir Canal
- Iniciar o Canal
- Parar Canal
- Iniciar Iniciador de Canal
- Iniciar Ouvinte de Canal
- Resolver Canal
- Reconfigurar Cluster
- Refresh Cluster
- Suspender Gerenciador de Filas
- Retomar Gerenciador de Filas

WebSphere MQ Autoridades de Objeto

Tabela 1. Windows, HP Integrity NonStop Server, UNIX and Linux sistemas-autoridades de objeto

Comando:	**Autoridade do objeto WebSphere MQ**	**Autoridade de classe (para tipo de objeto)**
Alterar Informações sobre Autenticação	dsp e chg	n/a
Alterar Canal	dsp e chg	n/a
Mudar ouvinte de canal	dsp e chg	n/a
Mudar canal de conexão do cliente	dsp e chg	n/a
Alterar Lista de Nomes	dsp e chg	n/a
Processo de Mudança	dsp e chg	n/a

Tabela 1. Windows, HP Integrity NonStop Server, UNIX and Linux sistemas-autoridades de objeto (continuação)		
Comando:	**Autoridade do objeto WebSphere MQ**	**Autoridade de classe (para tipo de objeto)**
Alterar a Fila	dsp e chg	n/a
Alterar Gerenciador de Filas	chg *consulte Nota 3 e Nota 5*	n/a
mudar Serviço	dsp e chg	n/a
Limpar Fila	clr	n/a
Copiar Informações sobre Autenticação	dsp	crt
Copiar Informações de Autenticação (Substituir) *consulte a Nota 1*	*de:* dsp *para:* chg	crt
Copiar Canal	dsp	crt
Copiar Canal (Substituir) *consulte a Nota 1*	*de:* dsp *para:* chg	crt
Copiar ouvinte de canal	dsp	crt
Copiar ouvinte de canal (Substituir) *consulte a Nota 1*	*de:* dsp *para:* chg	crt
Copiar canal de conexão do cliente	dsp	crt
Copiar canal de conexão do cliente (Substituir) *consulte a Nota 1*	*de:* dsp *para:* chg	crt
Copiar Lista de Nomes	dsp	crt
Copiar lista de nomes (Substituir) *consulte a Nota 1*	*a partir de:* dsp *para:* dsp e chg	crt
Copiar processo	dsp	crt
Copiar processo (substituir) *consulte a Nota 1*	*de:* dsp *para:* chg	crt
Copiar Fila	dsp	crt
Copiar fila (substituir) *consulte a Nota 1*	*a partir de:* dsp *para:* dsp e chg	crt
Criar Informações sobre Autenticação	*(informações de autenticação padrão do sistema)* dsp	crt
Criar informações de autenticação (substituir) *consulte a Nota 1*	*(informações de autenticação padrão do sistema)* dsp *para:* chg	crt
Criar Canal	*(canal padrão do sistema)* dsp	crt
Crie canal (substituir) *consulte a Nota 1*	*(canal padrão do sistema)* dsp *para:* chg	crt
Criar ouvinte de canal	*(listener padrão do sistema)* dsp	crt

Tabela 1. Windows, HP Integrity NonStop Server, UNIX and Linux sistemas-autoridades de objeto (continuação)		
Comando:	**Autoridade do objeto WebSphere MQ**	**Autoridade de classe (para tipo de objeto)**
Criar ouvinte de canal (substituir) *consulte a Nota 1*	*(listener padrão do sistema)* dsp *para:* chg	crt
Criar canal de conexão do cliente	*(canal padrão do sistema)* dsp	crt
Criar canal de conexão do cliente (substituir) *consulte a Nota 1*	*(canal padrão do sistema)* dsp *para:* chg	crt
Criar Lista de Nomes	*(lista de nomes padrão do sistema)* dsp	crt
Criar lista de nomes (substituir) *consulte a Nota 1*	*(lista de nomes padrão do sistema)* dsp *para:* dsp e chg	crt
Criar processo	*(processo padrão do sistema)* dsp	crt
Criar processo (substituir) *consulte a Nota 1*	*(processo padrão do sistema)* dsp *para:* chg	crt
Criar fila	*(fila padrão do sistema)* dsp	crt
Criar fila (substituir) *consulte a Nota 1*	*(fila padrão do sistema)* dsp *para:* dsp e chg	crt
Criar um ID do serviço	*(fila padrão do sistema)* dsp	crt
Criar serviço (substituir) *consulte a Nota 1*	*(fila padrão do sistema)* dsp *para:* chg	crt
Excluir Informações sobre Autenticação	dsp e dlt	n/a
Excluir Registro de Autoridade	*(objeto do gerenciador de filas)* chg *consulte a Nota 4*	*consulte a Nota 4*
Excluir Canal	dsp e dlt	n/a
Excluir ouvinte de canal	dsp e dlt	n/a
Excluir canal de conexão do cliente	dsp e dlt	n/a
Delete Namelist	dsp e dlt	n/a
Excluir Processo	dsp e dlt	n/a
Excluir fila	dsp e dlt	n/a
Excluir Serviço	dsp e dlt	n/a
Consultar Informações sobre Autenticação	dsp	n/a
Consultar Registros de Autoridade	*consulte a Nota 4*	*consulte a Nota 4*
Consultar Canal	dsp	n/a
Consultar ouvinte de canal	dsp	n/a
Consultar status do canal (para **ChannelType** MQCHT_CLSSDR)	inq	n/a

Tabela 1. Windows, HP Integrity NonStop Server, UNIX and Linux sistemas-autoridades de objeto (continuação)		
Comando:	**Autoridade do objeto WebSphere MQ**	**Autoridade de classe (para tipo de objeto)**
Consultar canal de conexão do cliente	dsp	n/a
Consultar Lista de Nomes	dsp	n/a
Consultar Processo	dsp	n/a
Consultar Fila	dsp	n/a
Consultar Gerenciador de Filas	*consulte a nota 3*	n/a
Consultar Status da Fila	dsp	n/a
Consultar Serviço	dsp	n/a
Executar ping no Canal	ctrl	n/a
Executar Ping do Gerenciador de Filas	*consulte a nota 3*	n/a
Atualizar Gerenciador de Filas	(objeto do gerenciador de filas) chg	n/a
Atualizar segurança (para **SecurityType** MQSECTYPE_SSL)	(objeto do gerenciador de filas) chg	n/a
Redefinir Canal	ctrlx	n/a
Reconfigurar Gerenciador de Filas	(objeto do gerenciador de filas) chg	n/a
Reconfigurar as Estatísticas de Fila	dsp e chg	n/a
Resolver Canal	ctrlx	n/a
Configurar Registro de Autoridade	*(objeto do gerenciador de filas)* chg *consulte a Nota 4*	*consulte a Nota 4*
Iniciar o Canal	ctrl	n/a
Parar Canal	ctrl	n/a
Para Conexão	(objeto do gerenciador de filas) chg	n/a
Iniciar Atendente	ctrl	n/a
Parar Atendente	ctrl	n/a
Iniciar Serviço	ctrl	n/a
Parar Serviços	ctrl	n/a
Escapar	*consulte a Nota 2*	*consulte a Nota 2*

Notes:

1. Este comando aplica-se se o objeto a ser substituído existir, caso contrário, a verificação de autoridade é como para Criar ou Copiar sem Substituir.
2. A autoridade necessária é determinada pelo comando MQSC definido pelo texto de escape e é equivalente a um dos comandos anteriores.

3. Para processar qualquer comando PCF, o ID do usuário deve ter autoridade de dsp para o objeto de gerenciador de filas no sistema de destino.
4. Este comando PCF for autorizado a menos que o servidor de comandos foi iniciado com o parâmetro -a. Por padrão, o servidor de comandos é iniciado quando o gerenciador de filas é iniciado e sem o parâmetro -a. Consulte o Guia de administração do sistema para obter informações adicionais.
5. Conceder a um ID do usuário a autoridade *chg* para um gerenciador de filas fornece a capacidade de configurar os registros de autoridade para todos os grupos e usuários. Não conceda essa autoridade para usuários ou aplicativos comuns.

WebSphere MQ também fornece alguns pontos de saída de segurança do canal para que você possa fornecer seus próprios programas de saída de usuário para verificação de segurança. Detalhes são fornecidos no manual Exibindo um canal.

Usando o MQAI para simplificar o uso de PCFs

O MQAI é uma interface de administração para WebSphere MQ disponível nas plataformas AIX, HP-UX, IBM i, Linux, Solaris e Windows .

O MQAI executa tarefas de administração em um gerenciador de filas por meio do uso de *pacotes de dados*. Pacotes de dados permitem manipular propriedades (ou parâmetros) de objetos de uma forma que é mais fácil do que usar PCFs.

Utilize o MQAI das seguintes maneiras:

Para simplificar a utilização das mensagens PCF
O MQAI é uma maneira fácil de administrar o WebSphere MQ; não é necessário gravar suas próprias mensagens PCF, evitando os problemas associados a estruturas de dados complexas

Para transmitir parâmetros em programas gravados utilizando chamadas MQI, a mensagem PCF deve conter o comando e os detalhes dos dados de sequência ou de número inteiro. Para fazer isso, você precisa várias instruções em seu programa para cada estrutura e espaço de memória deve ser alocado. Esta tarefa pode ser longa e trabalhosa.

Programas escritos utilizando os parâmetros de transmissão MQAI no pacote de dados apropriados e você precisa somente de uma instrução para cada estrutura. O uso de pacotes de dados MQAI remove a necessidade para você manipular as matrizes e alocar o armazenamento e fornece algum grau de isolamento dos detalhes do PCF.

Para manipular as condições de erro mais facilmente
É difícil obter códigos de retorno de volta a partir de comandos PCF, mas o MQAI torna mais fácil para o programa tratar condições de erro.

Depois de ter criado e preenchido o pacote de dados, é possível enviar uma mensagem de comando de administração para o servidor de comandos de um gerenciador de filas usando a chamada `mqExecute`, que aguarda todas as mensagens de resposta. A chamada `mqExecute` manipula a troca com o servidor de comandos e retorna respostas em um *pacote de respostas*.

Para obter mais informações sobre o MQAI, consulte “Introdução ao IBM WebSphere MQ Administration Interface (MQAI)” na página 20.

Introdução ao IBM WebSphere MQ Administration Interface (MQAI)

A IBM WebSphere MQ Administration Interface (MQAI) é uma interface de programação para o IBM WebSphere MQ. Ele desempenha tarefas de administração em um gerenciador de filas usando pacotes de dados do IBM WebSphere MQ manipular propriedades (ou parâmetros) de objetos de forma que é mais fácil do que usar Programmable Command Formats (PCFs).

Conceitos e Terminologia do MQAI

O MQAI é uma interface de programação para WebSphere MQ, usando a linguagem C e também Visual Basic para Windows. Ele está disponível em plataformas diferentes do z/OS..

Ele executa tarefas de administração em um Gerenciador de Filas do WebSphere MQ usando pacotes de dados Os pacotes de dados permitem manipular propriedades (ou parâmetros) de objetos de maneira mais fácil do que com o uso de outras interfaces de administração, como Programmable Command Formats (PCFs). O MQAI oferece manipulação de PCFs mais fácil do que as chamadas MQGET e MQPUT.

Para obter mais informações sobre pacotes de dados, consulte “Pacotes de dados” na página 47. Para obter mais informações sobre PCFs, consulte “Introdução aos Formatos de Comando Programável” na página 9

Uso do MQAI

É possível usar MQAI para:

- Simplificar o uso de mensagens do PCF. O MQAI é uma maneira fácil de administrar o WebSphere MQ; não é necessário gravar suas próprias mensagens PCF e, portanto, evitar os problemas associados a estruturas de dados complexas
- Manipular condições de erro com facilidade. É difícil obter códigos de retorno de volta dos comandos de script do WebSphere MQ (MQSC), mas o MQAI facilita para o programa manipular condições de erro.
- Trocar dados entre aplicativos. Os dados do aplicativo são enviados em formato PCF e compactados e descompactados pelo MQAI. Se seus dados da mensagem consistirem em números inteiros e sequências de caracteres, será possível usar o MQAI para aproveitar a conversão de dados integrados do WebSphere MQ para dados PCF. Isso evita a necessidade de gravar saídas de conversão de dados. Para obter mais informações sobre como usar MQAI para administrar o WebSphere MQ e trocar dados entre aplicativos, consulte “Usando o MQAI para simplificar o uso de PCFs” na página 20.

Exemplos do Uso do MQAI

A lista mostrada fornece alguns programas de exemplo que demonstram o uso do MQAI. As amostras executam as seguintes tarefas:

1. Crie uma fila local “Programa C de amostra para criar uma fila local (amqsaicq.c)” na página 22
2. Exibir eventos na tela usando um monitor de eventos simples.“Programa C de amostra para exibir eventos usando um monitor de eventos (amqsaiem.c)” na página 26
3. Imprimir uma lista de todas as filas locais e suas atuais profundidades.“Programa C de amostra para consultar filas e informações de impressão (amqsailq.c)” na página 38
4. Imprimir uma lista de todos os canais e seus tipos.“Programa C de amostra para consulta sobre objetos do canal (amqsaicl.c)” na página 33

Construindo seu Aplicativo do MQAI

Para construir seu aplicativo usando o MQAI, você vincula às mesmas bibliotecas que faz para o WebSphere MQ. Para obter informações sobre como construir seus aplicativos WebSphere MQ , consulte Construindo um aplicativo WebSphere MQ .

Sugestões e Dicas para Configurar o WebSphere MQ Usando MQAI

O MQAI usa mensagens do PCF para enviar comandos de administração para o servidor de comandos em vez de lidar diretamente com o servidor de comandos. Dicas para configurar o WebSphere MQ usando o MQAI podem ser localizadas em “Sugestões e dicas para configurar o IBM WebSphere MQ” na página 42

IBM WebSphere MQ Interface de Administração (MQAI)

IBM WebSphere MQ para Windows, AIX, Linux, HP-UXe Solaris suportam a IBM WebSphere MQ Administration Interface (MQAI). A MQAI é uma interface de programação para IBM WebSphere MQ que dá a você uma alternativa para o MQI, para enviar e receber PCFs.

O MQAI usa os *pacotes de dados* que permitem manipular as propriedades (ou parâmetros) de objetos mais facilmente do que usar PCFs diretamente por meio do MQAI.

O MQAI fornece acesso de programação mais fácil para mensagens PCF passando parâmetros no pacote de dados, de modo de que apenas uma instrução seja necessária para cada estrutura. Este acesso remove a necessidade para o programador manipular as matrizes e alocar o armazenamento e fornece algum isolamento dos detalhes de PCF.

O MQAI administra o WebSphere MQ enviando mensagens PCF para o servidor de comandos e aguardando uma resposta

O MQAI está descrito na segunda seção deste manual. Consulte a documentação Usando Java para uma descrição de uma interface do modelo de objeto componente para MQAI.

Programa C de amostra para criar uma fila local (amqsaicq.c)

O programa C de amostra `amqsaicq.c` cria uma fila local usando o 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 amostra para exibir eventos usando um monitor de eventos (amqsaiem.c)

O programa C de amostra `amqsaiem.c` demonstra um monitor de eventos básico usando o 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, &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 amostra para consulta sobre objetos do canal (amqsaicl.c)

O exemplo de programa C `amqsaicl.c` consulta objetos de canal usando o 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 amostra para consultar filas e informações de impressão (amqsailq.c)

O programa de amostra C `amqsailq.c` consulta a profundidade atual das filas locais usando o 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);
}
```

Sugestões e dicas para configurar o IBM WebSphere MQ

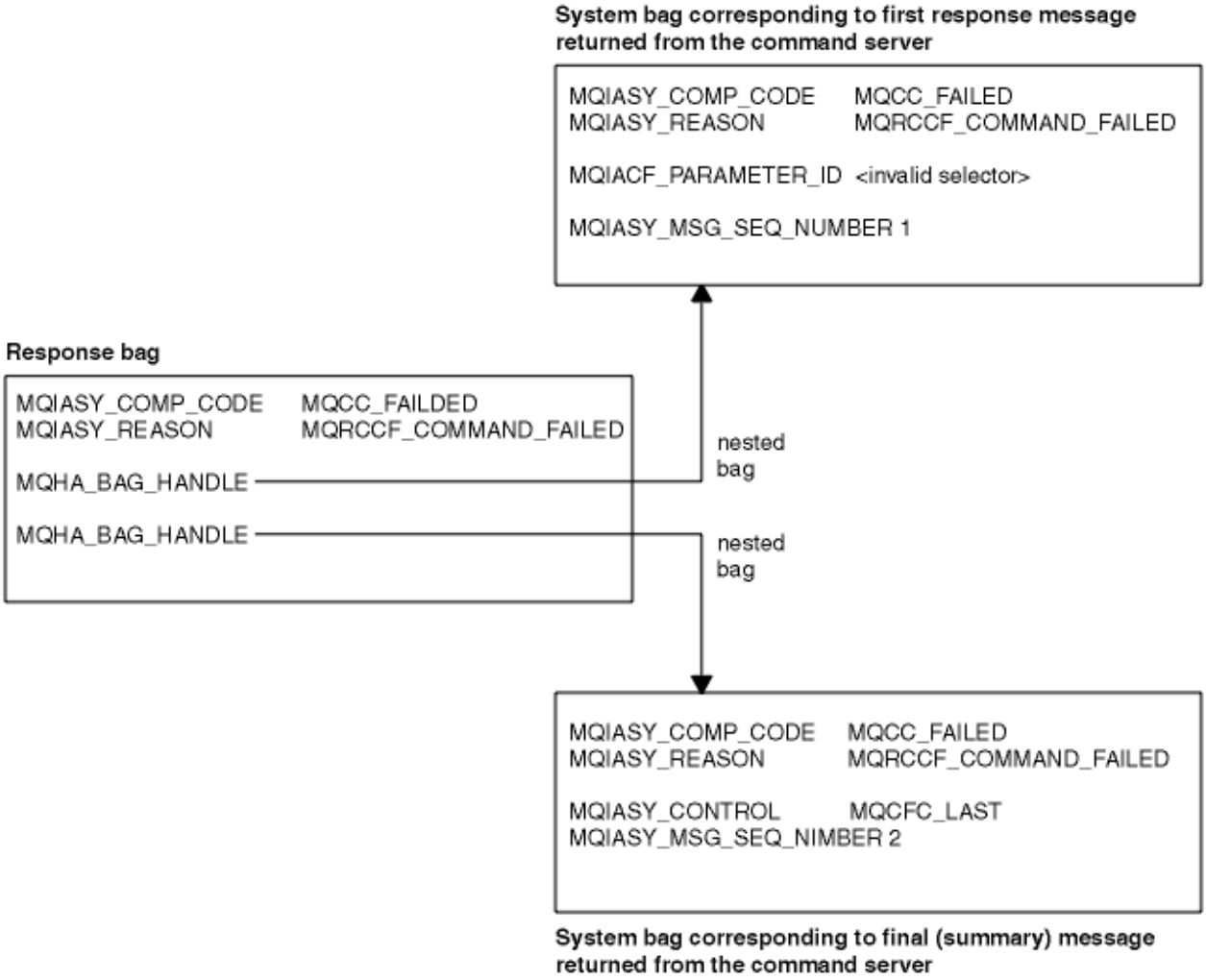
Dicas e sugestões de programação ao usar o MQAI

O MQAI usa mensagens do PCF para enviar comandos de administração para o servidor de comandos em vez de lidar diretamente com o servidor de comandos. Aqui estão algumas dicas para configurar o WebSphere MQ usando o MQAI:

- As sequências de caracteres no WebSphere MQ são preenchidas em branco para um comprimento fixo Usando C, sequências terminadas por nulo normalmente podem ser fornecidas como parâmetros de entrada para interfaces de programação do WebSphere MQ .
- Para limpar o valor de um atributo de sequência, configure-o como um único em branco em vez de uma sequência vazia.
- Considere previamente os atributos que você deseja mudar e consultar somente naqueles atributos.
- Certos atributos não podem ser mudados, por exemplo, um nome de fila ou um tipo de canal. Verifique se você tenta mudar somente aqueles atributos que possam ser modificados. Consulte a lista de parâmetros obrigatórios e opcionais para o objeto de mudança PCF específico. Consulte Definições dos formatos de comando programáveis.
- Se uma chamada MQAI falhar, alguns detalhes da falha são retornados ao pacote de respostas. Detalhes adicionais podem ser localizados em um pacote aninhado que pode ser acessado pelo seletor MQHA_BAG_HANDLE. Por exemplo, se uma chamada mqExecute falhar com um código de razão de MQRCCF_COMMAND_FAILED, essas informações serão retornadas no pacote de respostas. Uma razão possível para esse código de razão é que um seletor especificado não era válido para o tipo de mensagem de comando e esse detalhe de informações é encontrado em um pacote aninhado que pode ser acessado por um identificador de pacote.

  Para obter mais informações sobre MQExecute, consulte “Enviando comandos de administração para o servidor de comandos usando a chamada mqExecute” na página 56

  O diagrama a seguir mostra este cenário:



Tópicos avançados do MQAI

Informações sobre indexação, conversão de dados e a utilização de descritor de mensagens

- Indexando

  Índices são usados ao substituir ou remover itens de dados existentes de um pacote para preservar a ordem de inserção. É possível localizar detalhes completos sobre indexação em "Indexação no MQAI" na página 43.

- Conversão de Dados

  As sequências contidas em um pacote de dados MQAI pode estar em uma variedade de conjuntos de caracteres codificados e elas podem ser convertidas usando a chamada mqSetInteger. É possível localizar detalhes completos sobre conversão de dados em "Conversão de dados no MQAI" na página 44.

- Uso do descritor de mensagens

  MQAI gera um descritor de mensagens que é configurado para um valor inicial quando o pacote de dados é criado. É possível localizar detalhes completos sobre o uso do descritor de mensagens em "Uso do descritor de mensagens no MQAI" na página 46.

Indexação no MQAI

Os índices são usados ao substituir ou remover itens de dados existentes de um pacote. Há três tipos de indexação, permitindo que os itens de dados sejam facilmente recuperados.

Cada seletor e valor dentro de um item de dados em um pacote têm três números de índice associados:

- O índice relativo a outros itens que têm o mesmo seletor.
- O índice relativo à categoria de seletor (usuário ou sistema) ao qual o item pertence.
- O índice relativo a todos os itens de dados no pacote (usuário e sistema).

Isso permite a indexação por seletores de usuários, seletores de sistema ou ambos, conforme mostrado na Figura 1 na página 44.

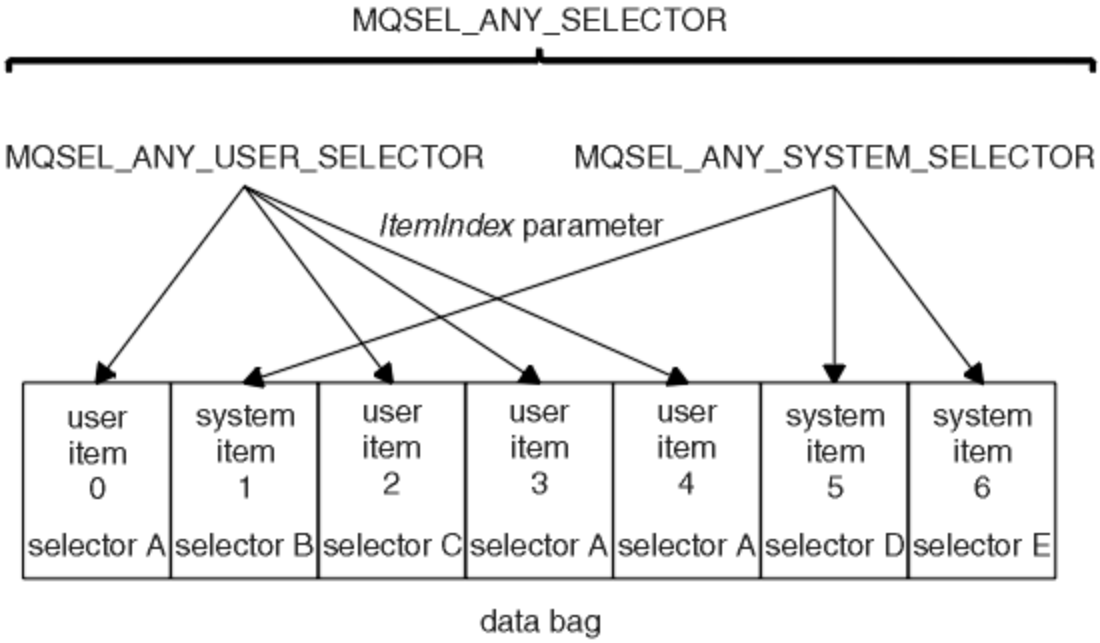


Figura 1. Indexando

Na Figura Figura 1 na página 44, o item 3 do usuário (seletor A) pode ser referido pelos pares de índice a seguir:

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

O índice baseia-se em zero como uma matriz em C; se houver 'n' ocorrências, o índice variará de zero a 'n-1', sem diferenças.

Os índices são usados ao substituir ou remover itens de dados existentes de um pacote. Quando usados dessa forma, a ordem de inserção é preservada, mas os índices de outros itens de dados podem ser afetados. Para obter exemplos disso, consulte Mudando as informações de um pacote e Excluindo itens de dados.

Os três tipos de indexação permitem fácil recuperação de itens de dados. Por exemplo, se houver três instâncias de um determinado seletor em um pacote, a chamada mqCountItems poderá contar o número de instâncias desse seletor e as chamadas mqInquire* poderão especificar tanto o seletor quanto o índice para consultar apenas esses valores. Isso é útil para atributos que podem ter uma lista de valores, como algumas das saídas em canais.

Conversão de dados no MQAI

As sequências contidas em um pacote de dados MQAI podem estar em uma variedade de conjuntos de caracteres codificados. Essas sequências podem ser convertidas usando a chamada mqSetInteger.

Como mensagens PCF, as sequências contidas em um pacote de dados MQAI podem estar em uma variedade de conjuntos de caracteres codificados. Geralmente, todas as sequências em uma mensagem

PCF estão no mesmo conjunto de caracteres codificados, ou seja, o mesmo conjunto que o gerenciador de filas.

Cada item de sequência em um pacote de dados contém dois valores; a própria sequência e o CCSID. A sequência incluída no pacote é obtida do parâmetro *Buffer* da chamada mqAddString ou mqSetString. O CCSID é obtido do item de sistema que contém um seletor de MQIASY_CODED_CHAR_SET_ID. Isso é conhecido como o *CCSID do pacote* e pode ser mudado usando a chamada mqSetInteger.

Quando você consulta o valor de uma sequência contida em um pacote de dados, o CCSID é um parâmetro de saída da chamada.

Tabela 2 na página 45 mostra as regras aplicadas ao converter pacotes de dados em mensagens e vice-versa:

Tabela 2. Processamento do CCSID

Chamada MQAI	CCSID	Entrada a ser chamada	Saída a ser chamada
mqBagToBuffer	CCSID do pacote (1)	Ignor.	Sem mudança
mqBagToBuffer	CCSIDs de sequência no pacote	Utilizada	Sem mudança
mqBagToBuffer	CCSIDs de sequência no buffer	Não-aplicável	Copiada de CCSIDs de sequência no pacote
mqBufferToBag	CCSID do pacote (1)	Ignor.	Sem mudança
mqBufferToBag	CCSIDs de sequência no buffer	Utilizada	Sem mudança
mqBufferToBag	CCSIDs de sequência no pacote	Não-aplicável	Copiada de CCSIDs de sequência no buffer
mqPutBag	CCSID do MQMD	Utilizada	Inalterado (2)
mqPutBag	CCSID do pacote (1)	Ignor.	Sem mudança
mqPutBag	CCSIDs de sequência no pacote	Utilizada	Sem mudança
mqPutBag	CCSIDs de sequência na mensagem enviada	Não-aplicável	Copiada de CCSIDs de sequência no pacote
mqGetBag	CCSID do MQMD	Usada para conversão de dados de mensagem	Configure como CCSID de dados retornados (3)
mqGetBag	CCSID do pacote (1)	Ignor.	Sem mudança
mqGetBag	CCSIDs de sequência na mensagem	Utilizada	Sem mudança
mqGetBag	CCSIDs de sequência no pacote	Não-aplicável	Copiada de CCSIDs de sequência na mensagem
mqExecute	CCSID do pacote de solicitação	Usado para MQMD da mensagem de solicitação (4)	Sem mudança
mqExecute	CCSID do pacote de resposta	Utilizado para conversão de dados da mensagem de resposta (4)	Configure como CCSID de dados retornados (3)
mqExecute	CCSIDs de sequência no pacote de solicitação	Usada para mensagem de solicitação	Sem mudança

Tabela 2. Processamento do CCSID (continuação)			
Chamada MQAI	**CCSID**	**Entrada a ser chamada**	**Saída a ser chamada**
mqExecute	CCSIDs de sequência no pacote de resposta	Não-aplicável	Copiada de CCSIDs de sequência na mensagem de resposta

Notes:

1. CCSID de pacote é o item de sistema com o seletor MQIASY_CODED_CHAR_SET_ID.
2. MQCCSI_Q_MGR mudou para o CCSID do gerenciador de filas real.
3. Se a conversão de dados for solicitada, o CCSID de dados retornados será o mesmo que o valor de saída. Se a conversão de dados não for solicitada, o CCSID de dados retornados será o mesmo que o valor da mensagem. Observe que nenhuma mensagem será retornada se a conversão de dados for solicitada, mas falhará.
4. Se o CCSID for MQCCSI_DEFAULT, o CCSID do gerenciador de filas será usado.

Uso do descritor de mensagens no MQAI

O descritor de mensagens gerado pelo MQAI é configurado como um valor inicial quando o pacote de dados é criado.

O tipo de comando PCF é obtido do item do sistema com o seletor MQIASY_TYPE. Quando você cria seu pacote de dados, o valor inicial desse item é configurado dependendo do tipo de pacote criado:

Tabela 3. Tipo de comando PCF	
Tipo de pacote	**Valor inicial do item MQIASY_TYPE**
MQCBO_ADMIN_BAG	MQCFT_COMMAND
MQCBO_COMMAND_BAG	MQCFT_COMMAND
MQCBO_*	MQCFT_USER

Quando o MQAI gera um descritor de mensagens, os valores usados nos parâmetros *Format* e *MsgType* dependem do valor do item de sistema com o seletor MQIASY_TYPE, conforme mostrado na Tabela 3 na página 46.

Tabela 4. Parâmetros Format e MsgType do MQMD		
Tipo de comando 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

A Tabela 4 na página 46 mostra que se você criar um pacote de administração ou um pacote de comandos, o *Format* do descritor de mensagens será MQFMT_ADMIN e o *MsgType* será MQMT_REQUEST. Isso é adequado para uma mensagem de solicitação do PCF enviada para o servidor de comandos quando se espera uma resposta de volta.

Outros parâmetros no descritor de mensagens usam os valores mostrados na Tabela 5 na página 47.

Tabela 5. Valores do descritor de mensagens	
Parâmetro	**Value**
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_1
Report	MQRO_NONE
MsgType	consulte Tabela 4 na página 46
Expiry	30 segundos (nota “1” na página 47)
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	depende do CCSID do pacote (nota “2” na página 47)
Format	consulte Tabela 4 na página 46
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NOT_PERSISTENT
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	consulte a nota “3” na página 47
ReplyToQMgr	Em branco

Notes:

1. Esse valor pode ser substituído na chamada **mqExecute** usando o parâmetro **OptionsBag** . Para obter informações sobre isso, consulte **mqExecute**..
2. Consulte o “Conversão de dados no MQAI” na página 44.
3. Nome da fila de resposta especificada pelo usuário ou da fila dinâmica temporária gerada pelo MQAI para mensagens do tipo **MQMT_REQUEST** Caso contrário, em branco.

Pacotes de dados

Um pacote de dados é um meio de manipulação de propriedades ou parâmetros de objetos utilizando o MQAI.

Pacotes de dados

- O pacote de dados contém zero ou mais *itens de dados*. Esses itens de dados são ordenados dentro do pacote conforme eles são colocados no pacote. Isto é chamado de *ordem de inserção*. Cada item de dados contém um *seletor* que identifica o item de dados e um *valor* desse item de dados que pode ser um número inteiro, um número inteiro de 64 bits, um filtro inteiro, uma sequência, um filtro de sequência, uma sequência de bytes, um filtro de sequência de bytes ou um identificador de outro pacote. Itens de dados são descritos em detalhes em “Item de dados” na página 50

  Há dois tipos de seletor: *seletores de usuário* e *seletores de sistema*. Esses são descritos em Seletores MQAI. Os seletores são geralmente exclusivos, mas é possível ter diversos valores para o mesmo seletor. Neste caso, um *índice* identifica a ocorrência específica de seletor que é necessário. Os índices são descritos em “Indexação no MQAI” na página 43.

Uma hierarquia desses conceitos é mostrada na Figura 1.

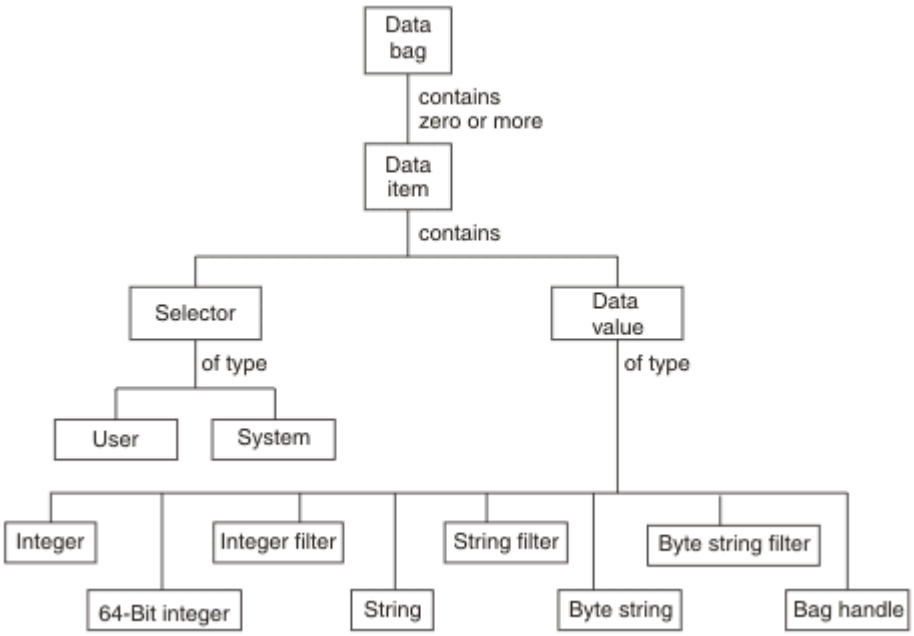


Figura 2. Hierarquia de conceitos MQAI

A hierarquia foi explicada em um parágrafo anterior.

Tipos de pacote de dados

É possível escolher o tipo de pacote de dados que você deseja criar dependendo da tarefa que você deseja desempenhar:

pacote do usuário???
Um pacote simples utilizado para dados do usuário.

pacote de administração
Um pacote criado para dados usados para administrar objetos do WebSphere MQ enviando mensagens de administração para um servidor de comandos O pacote de administração automaticamente implica determinadas opções conforme descrito em “Criação e exclusão de pacotes de dados” na página 49.

pacote de comandos???
Um pacote também criado para comandos para administrar objetos do WebSphere MQ . No entanto, ao contrário do pacote de administração, o pacote de comandos não implica automaticamente determinadas opções, embora estas opções estejam disponíveis. Para obter mais informações sobre as opções, consulte “Criação e exclusão de pacotes de dados” na página 49.

pacote de grupos???
Um pacote utilizado para conter um conjunto de itens de dados agrupados. Os pacotes de grupos não podem ser usados para administrar objetos do WebSphere MQ .

Além disso, o **pacote do sistema** é criado pelo MQAI quando uma mensagem de resposta é retornada do servidor de comandos e colocada no pacote de saída de um usuário. Um pacote do sistema não pode ser modificado pelo usuário.

Usando pacote de dados As diferentes maneiras de utilizar pacotes de dados são listados neste tópico:

Usando pacotes de dados

As diferentes maneiras de utilizar pacotes de dados são mostradas na lista a seguir:

- É possível criar e excluir dados de “Criação e exclusão de pacotes de dados” na página 49.
- É possível enviar dados entre aplicativos que utilizam dados de “Colocação e recebimento de pacotes de dados” na página 50.
- É possível incluir itens de dados para dados de “Incluindo pacotes para itens de dados” na página 51.
- É possível incluir um comando de consulta dentro de um pacote de dados do “Incluindo um comando de consulta a um pacote” na página 52.
- É possível consultar dentro de pacotes de dados do “Consultando dentro de pacotes de dados” na página 52.
- É possível contar itens de dados dentro de um pacote de dados “Contando itens de dados” na página 55.
- É possível mudar as informações dentro de um pacote de dados do “Mudando informações dentro de um pacote” na página 53.
- É possível limpar um pacote de dados do “Limpando um pacote através da chamada mqClearBag” na página 54.
- É possível truncar um pacote de dados do “Truncando um pacote usando a chamada mqTruncateBag” na página 54.
- É possível converter pacotes e buffers de “Convertendo pacotes e buffers” na página 54.

Criação e exclusão de pacotes de dados

Criando pacotes de dados

Para usar a MQAI, primeiro crie um pacote de dados usando a chamada mqCreateBag. Como entrada para essa chamada, forneça uma ou mais opções para controlar a criação do pacote.

O parâmetro *Options* da chamada MQCreateBag permite que você escolha se deseja criar um pacote do usuário, um pacote de comandos, um pacote de grupos ou um pacote de administração.

Para criar um pacote do usuário, um pacote de comandos ou um pacote de grupos, é possível escolher um ou mais opções adicionais para:

- Utilize o formulário da lista quando há duas ou mais ocorrências adjacentes da mesma seletor em um pacote.
- Reordene os itens de dados à medida que eles são incluídos em uma mensagem de PCF para assegurar que os parâmetros estão em sua ordem correta. Para obter informações adicionais sobre itens de dados, consulte “Item de dados” na página 50.
- Verifique os valores de seletores de usuário para os itens que você incluir no pacote.

Pacotes de administração automaticamente significam essas opções.

Um pacote de dados é identificado por seu identificador. O identificador de pacote é retornado a partir de mqCreateBag e deve ser fornecido em todas as outras chamadas que utilizam o pacote de dados.

Para uma descrição completa da chamada mqCreateBag, consulte mqCreateBag.

Exclusão de pacotes de dados

Qualquer pacote de dados que é criado pelo usuário também deve ser excluído usando a chamada mqDeleteBag. Por exemplo, se um pacote é criado no código do usuário, ele também deve ser excluído no código do usuário.

Pacotes do sistema são criados e excluídos automaticamente pela MQAI. Para obter informações adicionais sobre isto, consulte “Enviando comandos de administração para o servidor de comandos usando a chamada mqExecute” na página 56. O código do usuário não pode excluir um pacote do sistema.

Para obter uma descrição completa da chamada mqDeleteBag, consulte mqDeleteBag.

Colocação e recebimento de pacotes de dados

Os dados também podem ser enviados entre os aplicativos colocando e obtendo pacotes de dados utilizando as chamadas mqPutBag e mqGetBag. Isso permite que o MQAI a manipular o buffer em vez de o aplicativo. A chamada mqPutBag converte o conteúdo do pacote especificado em uma mensagem PCF e envia a mensagem para a fila especificada e a chamada mqGetBag remove a mensagem da fila especificada e converte-a de volta em um pacote de dados. Portanto, a chamada mqPutBag é o equivalente da chamada mqBagToBuffer seguido por MQPUT e o mqGetBag é o equivalente da chamada MQGET seguida por mqBufferToBag.

Para obter mais informações sobre o envio e recebimento de mensagens PCF em uma fila específica, consulte "Enviando e recebendo mensagens PCF em uma fila especificada" na página 12

Nota: Se você optar por utilizar a chamada mqGetBag, os detalhes de PCF dentro da mensagem devem ser corretos; se não forem, ocorrerá um erro apropriado e a mensagem PCF não será retornada.

Item de dados

Itens de dados são utilizados para preencher pacotes de dados quando eles são criados. Esses itens de dados podem ser itens de usuário ou de sistema.

Esses itens do usuário contêm dados do usuário como atributos de objetos que estão sendo administrados. Itens do sistema devem ser utilizados para maior controle sobre as mensagens geradas: por exemplo, a geração de cabeçalhos da mensagem. Para obter mais informações sobre os itens do sistema, consulte "Itens de sistema" na página 50.

Tipos de itens de dados

Quando você tiver criado um pacote de dados, é possível preenchê-lo com itens de sequência de caracteres ou de número inteiro. É possível consultar sobre todos os três tipos de item.

O item de dados pode ser um número inteiro ou itens de sequência. Aqui estão os tipos de item de dados disponível dentro do MQAI:

- Integer
- Número inteiro de 64 bits
- Filtro de número inteiro
- sequência de caracteres
- Filtragem de sequência
- Sequência de bytes
- Filtragem de sequência de bytes
- identificador de pacote

Utilizando itens de dados

Estas são as seguintes maneiras de utilizar itens de dados:

- "Contando itens de dados" na página 55.
- "Excluindo os itens de dados" na página 55.
- "Incluindo pacotes para itens de dados" na página 51.
- "Filtrando e consultando itens de dados" na página 52.

Itens de sistema

Itens de sistema podem ser usados para:

- A geração de cabeçalhos PCF. Itens de sistema podem controlar o identificador de comandos PCF, as opções de controle, o número de sequência da mensagem e o tipo de comando.

- Conversão de dados. Itens de sistema manipulam o identificador do conjunto de caracteres para os itens de sequência de caracteres no pacote.

Como todos os itens de dados, os itens do sistema consistem em um seletor e um valor. Para obter informações sobre esses seletores e o que eles são para, consulte MQAI Seletores.

Itens de sistema são exclusivos. Um ou mais itens do sistema pode ser identificado por um seletor de sistema. Existe somente uma ocorrência de cada seletor de sistema.

A maioria dos itens do sistema pode ser modificada (consulte “Mudando informações dentro de um pacote” na página 53), mas as opções de criação de pacote não podem ser mudadas pelo usuário. Não é possível excluir itens do sistema. (Consulte “Excluindo os itens de dados” na página 55.)

Incluindo pacotes para itens de dados

Quando um pacote de dados é criado, é possível preenchê-lo com itens de dados. Esses itens de dados podem ser itens de usuário ou de sistema. Para obter informações sobre itens de dados, consulte “Item de dados” na página 50.

A MQAI permite que você inclua itens de número inteiro, itens de número inteiro de 64 bits, itens de filtro de número inteiro, itens de sequência de caracteres, filtragem de sequência, itens da sequência de bytes e itens de filtro da sequência de byte para pacotes e isto é mostrado em Figura 3 na página 51. Os itens são identificados por um seletor. Geralmente um seletor identifica um único item, mas nem sempre é o caso. Se um item de dados com o seletor especificado já está presente no pacote, uma instância adicional desse seletor é incluída no final do pacote.

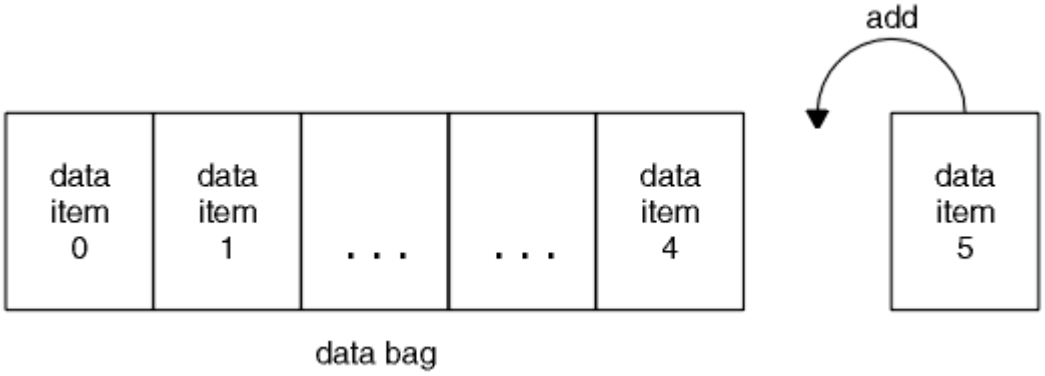


Figura 3. Incluindo itens de dados

Incluir itens de dados em um pacote usando as chamadas mqAdd*:

- Para incluir itens de número inteiro, use a chamada mqAddInteger conforme descrito em mqAddInteger
- Para incluir itens de número inteiro de 64 bits, use a chamada mqAddInteger64 conforme descrito em mqAddInteger64
- Para incluir itens de filtro de número inteiro, use a chamada mqAddIntegerFilter conforme descrito em mqAddIntegerFilter
- Para incluir itens de sequência de caracteres, use a chamada mqAddString conforme descrito em mqAddString
- Para incluir itens de filtragem de sequência, use a chamada mqAddStringFilter conforme descrito em mqAddStringFilter
- Para incluir itens de sequência de bytes, use a chamada mqAddByteString conforme descrito em mqAddByteString
- Para incluir itens de filtragem de sequência de bytes, use a chamada mqAddByteStringFilter conforme descrito em mqAddByteStringFilter

Para obter mais informações sobre a inclusão de itens de dados em um pacote, consulte “Itens de sistema” na página 50.

Incluindo um comando de consulta a um pacote

A chamada mqAddInquiry é utilizado para incluir um comando de consulta a um pacote. A chamada é especificamente para fins de administração, para que ele possa ser utilizado somente com pacotes de administração. Ele permite especificar os seletores de atributos nos quais você deseja consultar no WebSphere MQ.

Para obter uma descrição integral da chamada mqAddInquiry, consulte mqAddInquiry.

Filtrando e consultando itens de dados

Ao usar o MQAI para consultar sobre os atributos de objetos do WebSphere MQ , é possível controlar os dados que são retornados para seu programa de duas maneiras:

- É possível ***filtrar*** os dados que são retornados usando as chamadas mqAddInteger e mqAddString. Esta abordagem permite especificar um par de *Selector* e *ItemValue*, por exemplo:

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

  Este exemplo especifica que o tipo de fila (*Selector*) deve ser local (*ItemValue*) e esta especificação deve corresponder aos atributos do objeto (nesse caso, uma fila) sobre o qual você está consultando.

  Outros atributos que podem ser filtrados correspondem aos PCF Inquire* comandos que podem ser localizados em “Introdução aos Formatos de Comando Programável” na página 9. Por exemplo, para consultar sobre os atributos de um canal, consulte o comando Inquire Channel nesta documentação do produto. Os "parâmetros obrigatórios" e "parâmetros opcionais" do comando Inquire Channel identificam os seletores que podem ser utilizados para filtragem.
- É possível ***consultar*** atributos específicos de um objeto utilizando a chamada mqAddInquiry. Isso especifica o seletor no qual você está interessado. Se você não especificar o seletor, todos os atributos do objeto são retornados.

Aqui está um exemplo de filtragem e consulta os atributos de uma fila:

```
/* 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 obter mais exemplos de filtragem e consulta de itens de dados, consulte “Exemplos do Uso do MQAI” na página 21.

Consultando dentro de pacotes de dados

É possível consultar sobre:

- O valor de um item de número inteiro utilizando a chamada mqInquireInteger. Consulte mqInquireInteger.
- O valor de um item de inteiro 64-bit usando a chamada mqInquireInteger64. Consulte mqInquireInteger64.
- O valor de um item do filtro de número inteiro usando a chamada mqInquireIntegerFilter. Consulte mqInquireIntegerFilter.
- O valor de um item de sequência usando a chamada mqInquireString. Consulte mqInquireString.
- O valor de um item de filtragem de sequência usando a chamada mqInquireStringFilter. Consulte mqInquireStringFilter.

- O valor de um item de sequência de bytes utilizando a chamada mqInquireByteString. Consulte mqInquireByteString.
- O valor de um item de filtragem de sequência de byte usando a chamada mqInquireByteStringFilter. Consulte mqInquireByteStringFilter.
- O valor de um identificador de pacote através da chamada mqInquireBag. Consulte mqInquireBag.

Também é possível consultar sobre o tipo (número inteiro, inteiro de 64 bits, filtro de número inteiro, sequência de caracteres, filtragem de sequência, sequência de bytes, filtro de filtragem de sequência ou identificador de pacote) de um item específico usando a chamada mqInquireItemInfo. Consulte mqInquireItemInfo.

Mudando informações dentro de um pacote

O MQAI permite que você mude informações em um pacote usando as chamadas mqSet*. É possívelrá:

1. Modificar itens de dados dentro de um pacote. O índice permite que uma instância individual de um parâmetro seja substituída identificando a ocorrência do item a ser modificado (consulte Figura 4 na página 53).

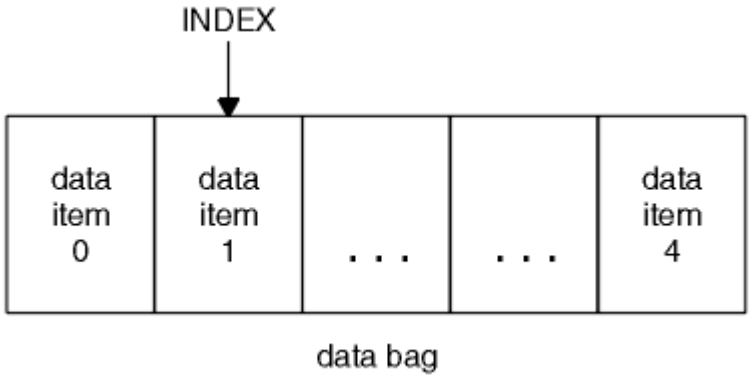


Figura 4. Modificando um item de dados único

2. Excluir todas as ocorrências existentes do seletor especificado e incluir uma nova ocorrência no final do pacote. (Consulte Figura 5 na página 53.) Um valor de índice especial permite que ***todas*** as instâncias de um parâmetro sejam substituídas.

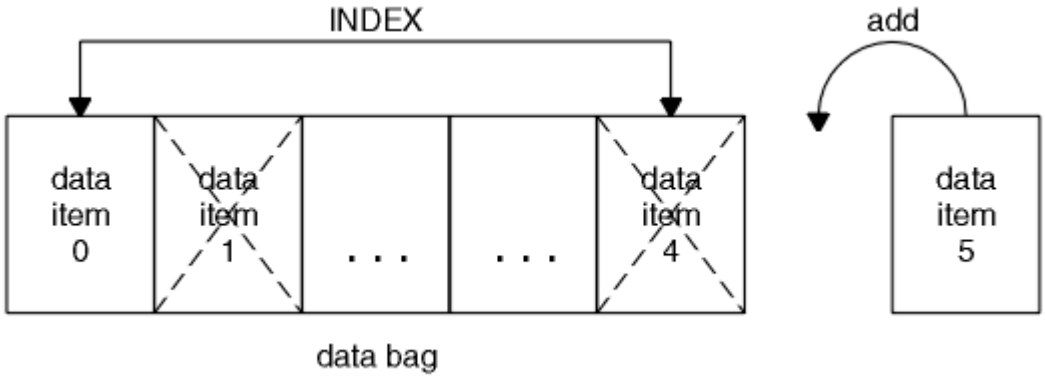


Figura 5. Modificando todos os itens de dados

Nota: O índice preserva a ordem de inserção dentro do pacote, mas pode afetar os índices de outros itens de dados.

A chamada mqSetInteger permite que você modifique itens de número inteiro dentro de um pacote. A chamada mqSetInteger64 permite que você modifique itens de número inteiro de 64 bits. A chamada mqSetIntegerFilter permite que você modifique itens de filtro de número inteiro. A chamada mqSetString permite que você modifique os itens de sequência de caracteres. A chamada mqSetStringFilter permite que você modifique os itens de filtragem de sequência. A chamada mqSetByteString permite que você modifique os itens de sequência de bytes. A chamada mqSetByteStringFilter permite que você modifique os itens de filtragem de sequência de bytes. Como alternativa, é possível usar essas chamadas para

excluir todas as ocorrências existentes do seletor especificado e incluir uma nova ocorrência no final do pacote. O item de dados pode ser um item do usuário ou um item de sistema.

Para ver uma descrição completa dessas chamadas, consulte:

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

Limpando um pacote através da chamada mqClearBag

A chamada mqClearBag removerá todos os itens do usuário a partir de um pacote do usuário e reconfigura os itens do sistema para seus valores iniciais. Pacotes do sistema contidos dentro do pacote também são excluídos.

Para obter uma descrição completa da chamada mqClearBag, consulte mqClearBag.

Truncando um pacote usando a chamada mqTruncateBag

A chamada mqTruncateBag reduz o número de itens do usuário em um pacote do usuário através da exclusão dos itens do final do pacote, começando com o item incluído mais recentemente. Por exemplo, ele pode ser utilizado quando utilizar as informações do cabeçalho mesmo para gerar mais de uma mensagem.

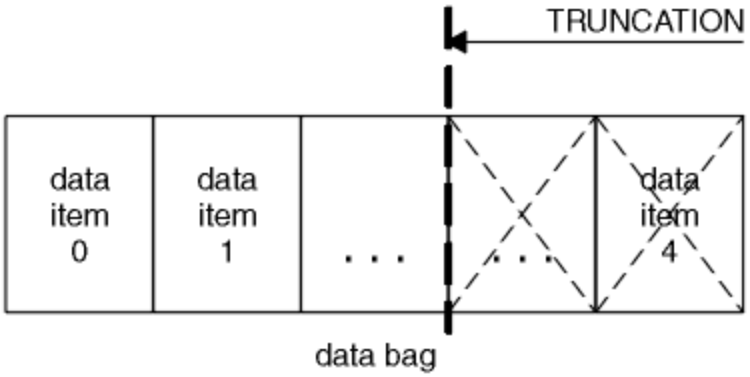


Figura 6. Truncando um pacote

Para obter uma descrição completa da chamada mqTruncateBag, consulte mqTruncateBag.

Convertendo pacotes e buffers

Para enviar dados entre aplicativos, primeiramente os dados da mensagem são colocados em um pacote. Em seguida, os dados no pacote são convertidos em uma mensagem PCF usando a chamada mqBagToBuffer. A mensagem PCF é enviada para a fila requerida usando a chamada MQPUT. Isso é mostrado na Figura do Figura 7 na página 54. Para obter uma descrição completa da chamada mqBagToBuffer, consulte mqBagToBuffer.

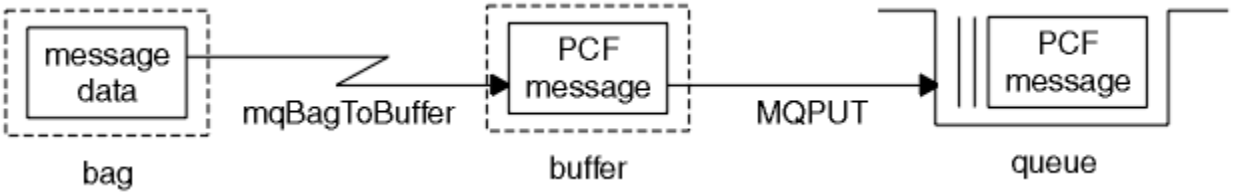


Figura 7. Convertendo pacotes em mensagens PCF

Para receber dados, a mensagem será recebida em um buffer usando a chamada MQGET. Os dados no buffer são, então, convertidos em um pacote usando a chamada mqBufferToBag, fornecendo o buffer que contém uma mensagem PCF válida. Isso é mostrado na Figura do Figura 8 na página 55. Para obter uma descrição completa da chamada mqBufferToBag, consulte mqBufferToBag.

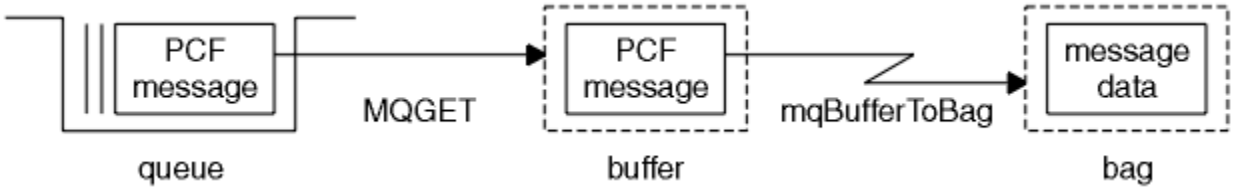


Figura 8. Convertendo para mensagens PCF do pacote

Contando itens de dados

A chamada mqCountItems conta o número de itens do usuário, itens do sistema ou ambos, que são armazenados em um pacote de dados e retorna esse número. Por exemplo, `mqCountItems(`*`Bag`*`, 7, ...)`, retorna o número de itens no pacote com um seletor de 7. Ele pode contar itens por seletor individual, por seletores de usuário, por seletores de sistema ou por todos os seletores.

Nota: Essa chamada conta o número de itens de dados, não o número de seletores exclusivos no pacote. Um seletor pode ocorrer várias vezes, portanto, pode haver seletores menos exclusivo no pacote de itens de dados.

Para obter uma descrição completa da chamada mqCountItems, consulte mqCountItems.

Excluindo os itens de dados

É possível excluir itens de pacotes de várias maneiras. É possívelrá:

- Remover um ou mais itens do usuário a partir de um pacote. Para obter informações detalhadas, consulte “Excluindo os itens de dados de um pacote usando a chamada mqDeleteItem” na página 55.
- Excluir ***todos*** os itens do usuário de um pacote, isto é, *limpar* um pacote. Para obter informações detalhadas, consulte “Limpando um pacote através da chamada mqClearBag” na página 54.
- Excluir itens do usuário a partir do final de um pacote, ou seja, *truncar* um pacote. Para obter informações detalhadas, consulte “Truncando um pacote usando a chamada mqTruncateBag” na página 54.

Excluindo os itens de dados de um pacote usando a chamada mqDeleteItem

A chamada mqDeleteItem remove um ou mais itens do usuário a partir de um pacote. O índice é utilizado para excluir um:

1. Uma única ocorrência do seletor especificado. (Consulte Figura 9 na página 55.)

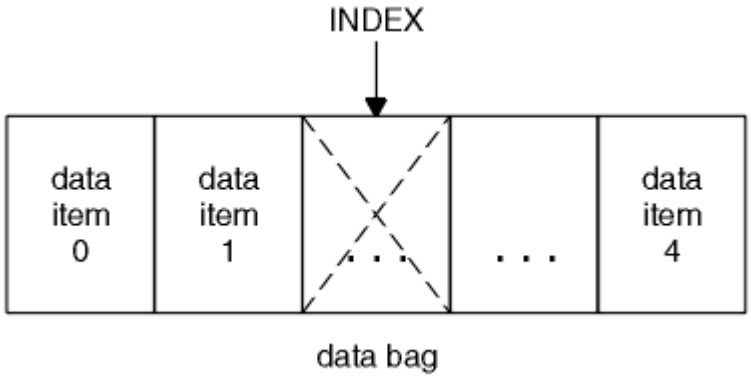


Figura 9. Excluindo um item de dados único

ou

2. Todas as ocorrências do seletor especificado. (Consulte Figura 10 na página 56.)

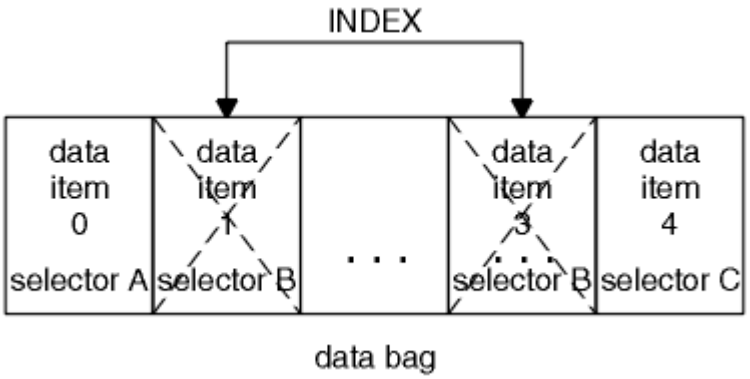


Figura 10. Excluindo todos os itens de dados

Nota: O índice preserva a ordem de inserção dentro do pacote, mas pode afetar os índices de outros itens de dados. Por exemplo, a chamada mqDeleteItem não preserva os valores de índice dos itens de dados que seguem o item excluído porque os índices estão reorganizados para preencher a lacuna que permanece do item excluído.

Para obter uma descrição integral da chamada mqDeleteItem, consulte mqDeleteItem.

Enviando comandos de administração para o servidor de comandos usando a chamada mqExecute

Quando um pacote de dados tiver sido criado e preenchido, uma mensagem de comando administrativo pode ser enviada para o servidor de comandos de um gerenciador de filas usando a chamada mqExecute. Isso controla a troca com o servidor de comandos e retorna respostas em um pacote.

Após ter criado e preenchido o pacote de dados, é possível enviar uma mensagem de comando de administração para o servidor de comandos de um gerenciador de filas. A maneira mais fácil de fazer isso é usando a chamada mqExecute. A chamada mqExecute envia uma mensagem de comando de administração como uma mensagem não persistente e aguarda as respostas. Respostas são retornadas em um pacote de respostas. Eles podem conter informações sobre atributos relacionados a vários objetos WebSphere MQ ou uma série de mensagens de resposta de erro PCF, por exemplo. Portanto, o pacote de respostas poderia conter um código de retorno somente ou poderia conter *pacotes aninhados*.

As mensagens de resposta são colocadas em pacotes do sistema que são criados pelo sistema. Por exemplo, para consultas sobre os nomes de objetos, um pacote do sistema é criado para conter esses nomes de objeto e o pacote é inserido no pacote do usuário. Identificadores para esses pacotes são inseridos no pacote de respostas e o pacote aninhado pode ser acessado pelo seletor MQHA_BAG_HANDLE. O pacote do sistema permanece no armazenamento, se ele não for excluído, até que o pacote de respostas seja excluído.

O conceito de *aninhando* será mostrado em Figura 11 na página 57.

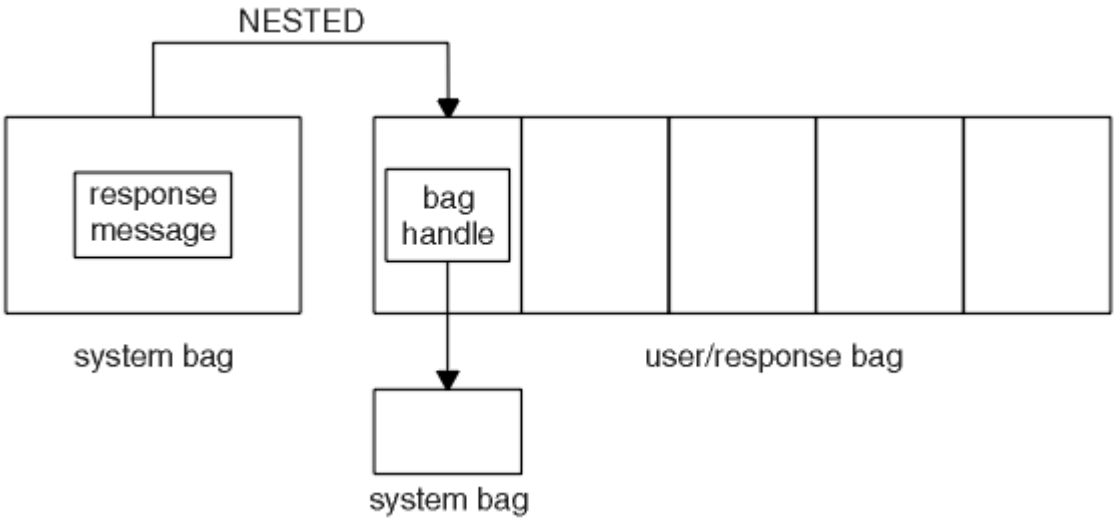


Figura 11. Agrupando

Como entrada para a chamada mqExecute, deve-se fornecer:

- Um MQI identificador de conexão.
- O comando a ser executado. Isso deve ser um dos valores MQCMD_*.

  Nota: Se este valor não for reconhecido pelo MQAI, o valor ainda será aceito. No entanto, se a chamada mqAddInquiry foi utilizada para inserir valores no pacote, esse parâmetro deve ser um comando INQUIRE reconhecido pelo MQAI. Ou seja, o parâmetro deve ter a forma MQCMD_INQUIRE_*.
- Opcionalmente, um identificador do pacote que contém opções que controlam o processamento da chamada. Este é também onde é possível especificar o tempo máximo em milissegundos que a MQAI deve aguardar para cada mensagem de resposta.
- Um identificador do pacote de administração que contém detalhes do comando de administração a ser emitido.
- Um identificador do pacote de respostas que recebe as mensagens de resposta.

Os seguintes são opcionais:

- Um identificador de objeto da fila para onde o comando de administração deve ser colocado.

  Se nenhum identificador de objeto for especificado, o comando de administração é colocada na SYSTEM.ADMIN.COMMAND.QUEUE que pertence ao gerenciador de filas atualmente conectados. Esse é o padrão.
- Um identificador de objeto da fila para onde as mensagens de resposta devem ser colocadas.

  É possível optar por colocar as mensagens de resposta em uma fila dinâmica que é criada automaticamente pelo MQAI. A fila criada existe para a duração da chamada somente e é excluída pelo MQAI na saída da chamada mqExecute.

Para obter exemplos de usos da chamada mqExecute, consulte Código de exemplo

Administração usando o IBM WebSphere MQ Explorer

O IBM WebSphere MQ Explorer permite executar administração local ou remota de sua rede a partir de um computador executando Windowsou Linux (plataformasx86 e x86-64) apenas.

IBM WebSphere MQ para Windowse IBM WebSphere MQ para Linux (plataformasx86 e x86-64) fornecem uma interface de administração chamada IBM WebSphere MQ Explorer para executar tarefas de administração como uma alternativa para usar comandos de controle ou MQSC. Comparando Conjuntos de Comandos mostra o que pode ser feito usando o IBM WebSphere MQ Explorer.

O IBM WebSphere MQ Explorer permite executar a administração local ou remota de sua rede a partir de um computador em execução Windowsou Linux (plataformasx86-64), apontando o IBM WebSphere MQ Explorer para os gerenciadores de filas e clusters nos quais você está interessado. As plataformas e os níveis de IBM WebSphere MQ que podem ser administrados usando o IBM WebSphere MQ Explorer são descritos em “Gerenciadores de filas remotas” na página 59.

Para configurar os gerenciadores de fila IBM WebSphere MQ remotos para que o IBM WebSphere MQ Explorer possa administrá-los, consulte “Software obrigatório e definições” na página 60.

Ele permite executar tarefas, geralmente associadas à configuração e ao ajuste fino do ambiente de trabalho para o IBM WebSphere MQ, localmente ou remotamente em um domínio do sistema Windows ou Linux (plataformasx86 e x86-64).

No Linux, o IBM WebSphere MQ Explorer pode falhar ao ser iniciado se você tiver mais de uma instalação Eclipse. Se isso acontecer, inicie o IBM WebSphere MQ Explorer usando um ID de usuário diferente daquele que você usa para a outra instalação Eclipse.

No Linux, para iniciar o IBM WebSphere MQ Explorer com êxito, deve-se conseguir gravar um arquivo para o seu diretório inicial e o diretório inicial deve existir.

O que você pode fazer com o IBM WebSphere MQ Explorer

Esta é uma lista das tarefas que é possível executar usando o IBM WebSphere MQ Explorer.

Com o IBM WebSphere MQ Explorer, é possível:

- Criar e excluir um gerenciador de filas (em sua máquina local somente).
- Iniciar e parar um gerenciador de filas (somente em sua máquina local).
- Definir, exibir e alterar as definições de objetos do WebSphere MQ como filas e canais.
- Procurar as mensagens em uma fila.
- Iniciar e parar um canal.
- Visualizar as informações de status sobre um canal, listener, fila ou objetos de serviço.
- Visualizar gerenciadores de filas em um cluster.
- Verificar para ver quais aplicativos, usuários ou canais possuem uma fila específica aberta.
- Criar um novo cluster de gerenciadores de filas usando o assistente *Criar Novo Cluster*.
- Incluir um gerenciador de filas em um cluster usando o assistente *Incluir Gerenciador de Filas para Cluster*.
- Gerenciar o objeto de informações sobre autenticação, utilizadas com SSL (Secure Sockets Layer) do canal de segurança.
- Criar e excluir iniciadores de canais, os monitores do acionador e listeners.
- Iniciar ou parar os servidores de comando, iniciadores de canais, monitores de acionadores e listeners.
- Configurar serviços específicos para iniciar automaticamente quando um gerenciador de filas é iniciado.
- Modificar as propriedades de gerenciadores de filas.
- Mudar o gerenciador de filas padrão local.
- Chamar a GUI do ikeyman para gerenciar certificados secure sockets layer (SSL), associar certificados com gerenciadores de filas e configurar e definir armazenamentos de certificados (em sua máquina local somente).
- Crie objetos JMS a partir de objetos WebSphere MQ e objetos WebSphere MQ a partir de objetos JMS.
- Crie um JMS Connection Factory para qualquer um dos tipos suportados atualmente..
- Modificar os parâmetros para qualquer serviço, como o número da porta TCP para um listener ou um nome de fila inicializador de canais.
- Iniciar ou parar o rastreio de serviço.

É possível desempenhar tarefas de administração utilizando uma série de *Conteúdo Visualizações* e *diálogos de propriedade*.

Visão de Conteúdo
Uma visualização Conteúdo é um painel que pode exibir o seguinte:

- Atributos e opções administrativas relacionados ao WebSphere MQ em si
- Atributos e opções administrativas relativas a um ou mais objetos relacionados.
- Atributos e opções administrativas para um cluster.

Diálogos de propriedades
Um diálogo de propriedade é um painel que exibe os atributos relacionados a um objeto em uma série de campos, alguns dos quais é possível editar.

Você navega pelo WebSphere MQ Explorer usando a *visualização doNavigator*. O Navegador permite que você selecione a Visualização Conteúdo que você precisa.

Gerenciadores de filas remotas

Há duas exceções para os gerenciadores de filas suportados que é possível se conectar.

Em um sistema Windows ou Linux (plataformasx86 e x86-64), o WebSphere MQ Explorer pode se conectar a todos os gerenciadores de filas suportados com as seguintes exceções:

- WebSphere MQ para z/OS gerenciadores de filas anteriores à Versão 6.0.
- Atualmente suportados MQSeries V2 gerenciadores de filas.

O IBM WebSphere MQ Explorer manipula as diferenças nas capacidades entre os diferentes níveis de comando e plataformas. No entanto, se ele encontra um atributo que ele não reconhece, o atributo não será visível.

Se você pretende administrar remotamente um gerenciador de filas V6.0 ou posterior no Windows usando o IBM WebSphere MQ Explorer em um computador WebSphere MQ V5.3 , deve-se instalar o Fix Pack 9 (CSD9) ou posterior em seu computador WebSphere MQ para Windows V5.3 .

Se você pretende administrar remotamente um gerenciador de filas V5.3 no iSeries usando o WebSphere MQ Explorer em um WebSphere MQ V6.0 ou computador posterior, você deve instalar o Fix Pack 11 (CSD11) ou posterior em seu computador WebSphere MQ para iSeries V5.3 . Este fix pack corrige problemas de conexões entre o WebSphere MQ Explorer e o gerenciador de filas do iSeries

Decidindo se deve usar o IBM WebSphere MQ Explorer

Ao decidir quando usar o IBM WebSphere MQ Explorer em sua instalação, considere as informações listadas neste tópico.

Você precisa estar ciente dos seguintes pontos:

Nomes de Objetos
Se você usar nomes minúsculos para gerenciadores de filas e outros objetos com o IBM WebSphere MQ Explorer, ao trabalhar com os objetos usando comandos MQSC, deverá colocar os nomes de objetos entre aspas simples ou o WebSphere MQ não os reconhecerá.

Gerenciadores de filas grandes
O IBM WebSphere MQ Explorer funciona melhor com os gerenciadores de filas pequenas. Se você tiver um grande número de objetos em um único gerenciador de filas, poderá ocorrer atrasos enquanto o WebSphere MQ Explorer extrai as informações necessárias a serem apresentadas em uma visualização

Grupos
WebSphere MQ clusters podem potencialmente conter centenas ou milhares de gerenciadores de filas. O WebSphere MQ Explorer apresenta os gerenciadores de filas em um cluster usando uma estrutura em árvore. O tamanho físico de um cluster não afeta a velocidade do IBM WebSphere MQ Explorer significativamente porque o IBM WebSphere MQ Explorer não se conecta aos gerenciadores de filas no cluster até que você os seleciona.

Configurando o IBM WebSphere MQ Explorer

Essa seção destaca as etapas necessárias para configurar o IBM WebSphere MQ Explorer.

Software obrigatório e definições

Certifique-se de atender os seguintes requisitos antes de tentar utilizar o IBM WebSphere MQ Explorer.

O IBM WebSphere MQ Explorer pode se conectar aos gerenciadores de filas remotas utilizando o protocolo de comunicação TCP/IP somente.

Verifique se:

1. Um servidor de comandos está em execução em cada gerenciador de filas administrado remotamente.
2. Um objeto de listener TCP/IP adequado deve estar em execução em cada gerenciador de filas remotas. Esse objeto pode ser o listener do IBM WebSphere MQ ou, em sistemas UNIX and Linux, o daemon inetd.
3. Um canal de conexão do servidor, por padrão denominado SYSTEM.ADMIN.SVRCONN, existe em todos os gerenciadores de filas remotas.

   É possível criar o canal usando o comando MQSC a seguir:

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

   Esse comando cria uma definição de canal básico. Se você deseja uma definição mais sofisticadas (para configurar a segurança, por exemplo), você precisa de parâmetros adicionais. Para obter mais informações, consulte DEFINE CHANNEL .
4. O sistema de filas, SYSTEM.MQEXPLORER.REPLY.MODEL, deve existir.

Segurança

Se você estiver usando o WebSphere MQ em um ambiente no qual é importante controlar o acesso de usuário a objetos específicos, poderá ser necessário considerar os aspectos de segurança do uso do IBM WebSphere MQ Explorer

Autorização para usar o IBM WebSphere MQ Explorer

Qualquer usuário pode usar o IBM WebSphere MQ Explorer, mas é necessário que determinadas autoridades se conectem, acessem e gerenciem gerenciadores de filas.

Para executar tarefas administrativas locais utilizando o WebSphere MQ Explorer, um usuário precisa ter a autoridade necessária para executar as tarefas administrativas. Se o usuário é um membro do grupo mqm, o usuário tem autoridade para executar todas as tarefas administrativas locais.

Para se conectar a um gerenciador de filas remotas e executar tarefas administrativas remotas usando o WebSphere MQ Explorer, o usuário que está executando o WebSphere MQ Explorer é necessário para ter as seguintes autoridades:

- Autoridade CONNECT no objeto do gerenciador de filas de destino
- Autoridade INQUIRE no objeto do gerenciador de filas de destino
- Autoridade DISPLAY no objeto do gerenciador de filas de destino
- Autoridade INQUIRE para a fila, SYSTEM.MQEXPLORER.REPLY.MODEL
- Autoridade DISPLAY para a fila, SYSTEM.MQEXPLORER.REPLY.MODEL

- Autoridade INPUT (get) para a fila, SYSTEM.MQEXPLORER.REPLY.MODEL
- Autoridade OUTPUT (put) para a fila, SYSTEM.ADMIN.COMMAND.QUEUE
- Autoridade INQUIRE na fila, SYSTEM.ADMIN.COMMAND.QUEUE
- Autoridade para executar a ação selecionada

Nota: Autoridade INPUT diz respeito à entrada para o usuário a partir de uma fila (uma operação get). Autoridade OUTPUT diz respeito à saída do usuário para uma fila (uma operação put).

Para se conectar a um gerenciador de filas remotas no WebSphere MQ para z/OS e executar tarefas administrativas remotas usando o IBM WebSphere MQ Explorer, o seguinte deve ser fornecido:

- Um perfil RACF para a fila do sistema SYSTEM.MQEXPLORER.REPLY.MODEL
- Um perfil RACF para as filas, AMQ.MQEXPLORER.*

Além disso, o usuário que está executando o WebSphere MQ Explorer precisa ter as seguintes autoridades:

- Autoridade UPDATE do RACF para a fila do sistema, SYSTEM.MQEXPLORER.REPLY.MODEL
- Autoridade UPDATE do RACF para as filas, AMQ.MQEXPLORER.*
- Autoridade CONNECT no objeto do gerenciador de filas de destino
- Autoridade para executar a ação selecionada
- Autoridade READ para todos os perfis hlq.DISPLAY.object na classe MQCMDS

Para obter informações sobre como conceder autoridade para objetos do WebSphere MQ , consulte concedendo acesso a um objeto do WebSphere MQ em sistemas UNIX ou Linux e Windows

Se um usuário tenta executar uma operação que ele não está autorizado a executar, o gerenciador de filas de destino chama procedimentos falha de autorização e a operação falha.

O filtro padrão no WebSphere MQ Explorer é exibir todos os objetos WebSphere MQ . Se houver quaisquer objetos do WebSphere MQ para os quais um usuário não possui autoridade DISPLAY, as falhas de autorização serão geradas Se eventos de autoridade estão sendo registrados, restringir o intervalo de objetos que são exibidos para aqueles objetos aos quais o usuário tem autoridade DISPLAY.

Segurança para conectar-se a gerenciadores de filas remotas

Deve-se proteger o canal entre o IBM WebSphere MQ Explorer e cada gerenciador de filas remotas.

O IBM WebSphere MQ Explorer se conecta a gerenciadores de filas remotas como um aplicativo cliente de MQI. Isso significa que cada gerenciador de filas remotas deve ter uma definição de um canal de conexão do servidor e um listener TCP/IP apropriado. Se você não proteger seu canal de conexão do servidor, é possível para um aplicativo malicioso se conectar ao canal de conexão do mesmo servidor e obter acesso a objetos do gerenciador de filas com autoridade ilimitada. A fim de proteger o canal de conexão do servidor ou especificar um valor não em branco para o atributo MCAUSER do canal, use registros de autenticação de canal ou use uma saída de segurança.

O valor padrão do atributo MCAUSER é o ID do usuário local. Se você especificar um nome de usuário que não esteja em branco como o atributo MCAUSER do canal de conexão do servidor, todos os programas que se conectam ao gerenciador de filas utilizando esse canal executam com a identidade do usuário nomeado e têm o mesmo nível de autoridade. Isso não ocorrerá se você utilizar registros de autenticação de canal.

Usando uma saída de segurança com o WebSphere MQ Explorer

É possível especificar uma saída de segurança padrão e saídas de segurança específicas do gerenciador de filas usando o WebSphere MQ Explorer

É possível definir uma saída de segurança padrão, que pode ser usada para todas as novas conexões do cliente do WebSphere MQ Explorer. Esta saída padrão pode ser substituída no momento em que uma conexão é feita. Também é possível definir uma saída de segurança para um gerenciador de filas único ou um conjunto de gerenciadores de filas, que entra em vigor quando uma conexão é feita. Você especifica

saídas usando o WebSphere MQ Explorer. Para obter mais informações, consulte a Central de Ajuda do WebSphere MQ

Usando o IBM WebSphere MQ Explorer para se conectar a um gerenciador de filas remotas que usa canais MQI ativados para SSL

O IBM WebSphere MQ Explorer se conecta a gerenciadores de filas remotas usando um canal MQI. Se você desejar proteger o canal MQI usando a segurança SSL, deve-se estabelecer o canal utilizando uma tabela de definição de canal do cliente.

Para obter informações sobre como estabelecer um canal MQI usando uma tabela de definição de canal do cliente, consulte Visão geral de IBM WebSphere MQ clientes MQI.

Quando você tiver estabelecido o canal utilizando uma tabela de definição de canal do cliente, é possível usar o IBM WebSphere MQ Explorer para se conectar a um gerenciador de filas remotas que usa canal MQI ativado para SSL, conforme descrito em "Tarefas no sistema que hospeda o gerenciador de filas remotas" na página 62 e "Tarefas no sistema que hospeda o IBM WebSphere MQ Explorer" na página 62.

Tarefas no sistema que hospeda o gerenciador de filas remotas

No sistema que hospeda o gerenciador de filas remotas, execute as seguintes tarefas:

1. Defina um servidor de conexão e par de canais de conexão do cliente e especifique o valor apropriado para a variável *SSLCIPH* na conexão do servidor em ambos os canais. Para obter mais informações sobre a variável *SSLCIPH*, consulte Protegendo canais com SSL
2. Envie a tabela de definição de canal `AMQCLCHL.TAB` , localizada no diretório `@ipcc` do gerenciador de filas, para o sistema que hospeda o IBM WebSphere MQ Explorer.
3. Inicie um listener TCP/IP em uma porta designada.
4. Coloque ambos os CA e certificados SSL pessoal no diretório SSL do gerenciador de filas:
   - `/var/mqm/qmgrs/+QMNAME+/SSL` para sistemas UNIX and Linux
   - `C:\Program Files\WebSphere MQ\qmgrs\+QMNAME+\SSL` para sistemas Windows

   Em que `+QMNAME+` é um token representando o nome do gerenciador de filas.
5. Crie um arquivo de banco de dados de chave do tipo CMS denominado `key.kdb` Armazene a senha em um arquivo, verificando a opção na GUI do iKeyman ou usando a opção `-stash` com os comandos **`runmqckm`** .
6. Inclua os certificados de CA para o banco de dados de chaves criado na etapa anterior.
7. Importe o certificado pessoal para o gerenciador de filas para o banco de dados de chave.

Para obter informações mais detalhadas sobre como trabalhar com o Secure Sockets Layer em sistemas Windows , consulte Trabalhando com SSL ou TLS em UNIX, Linux e sistemas Windows

Tarefas no sistema que hospeda o IBM WebSphere MQ Explorer

No sistema que hospeda o IBM WebSphere MQ Explorer, execute as seguintes tarefas:

1. Crie um arquivo de banco de dados de chave do tipo JKS denominado `key.jks`. Configure uma senha para este arquivo de banco de dados chave.

   O IBM WebSphere MQ Explorer usa Java arquivos keystore (JKS) para a segurança SSL e, portanto, o arquivo de armazenamento de chaves que está sendo criado para configurar o SSL para o IBM WebSphere MQ Explorer deve corresponder esse.
2. Inclua os certificados de CA para o banco de dados de chaves criado na etapa anterior.
3. Importe o certificado pessoal para o gerenciador de filas para o banco de dados de chave.
4. Nos sistemas Windows e Linux , inicie o MQ Explorer usando o menu do sistema, o arquivo executável `MQExplorer` ou o comando **`strmqcfg`** ..

5. Na barra de ferramentas do IBM WebSphere MQ Explorer , clique em **Janela-> Preferência**s, em seguida, expanda **WebSphere MQ Explorer** e clique em **Armazenamentos de Certificados de Cliente SSL** Digite o nome e a senha para o arquivo JKS criado na etapa 1 de "Tarefas no sistema que hospeda o IBM WebSphere MQ Explorer" na página 62, no Armazenamento de Certificados Confiáveis e Armazenamento de Certificados Pessoais e, em seguida, clique em **OK**.
6. Feche a janela **Preferências** e clique com o botão direito em **Gerenciadores de Filas**. Clique em **Mostrar/ocultar gerenciadores de filas** e, em seguida, clique em **Incluir** na tela **Mostrar/ocultar gerenciadores de filas**.
7. Digite o nome do gerenciador de filas e selecione o **Conectar Diretamente** a opção. Clique em Next.
8. Selecione **Utilizar tabela de definição de canal do cliente (CCDT)** e especifique o local do arquivo de tabela do canal que você transferiu do gerenciador de filas remotas na etapa 2 em "Tarefas no sistema que hospeda o gerenciador de filas remotas" na página 62 no sistema que hospeda o gerenciador de filas remotas.
9. Clique em **Concluir**. Agora é possível acessar o gerenciador de filas remotas no IBM WebSphere MQ Explorer.

Conectando através de outro gerenciador de filas

O IBM WebSphere MQ Explorer permite conectar a um gerenciador de filas por meio de um gerenciador de filas intermediário, para o qual o IBM WebSphere MQ Explorer já está conectado.

Neste caso, o IBM WebSphere MQ Explorer coloca as mensagens de comando PCF para o gerenciador de filas intermediário, especificando o seguinte:

- O parâmetro *ObjectQMgrName* no descritor de objetos (MQOD) como o nome do gerenciador de filas de destino. Para obter mais informações sobre a resolução do nome da fila, consulte Resolução do nome .
- O parâmetro *UserIdentifier* no descritor de mensagens (MQMD) como o ID do usuário local.

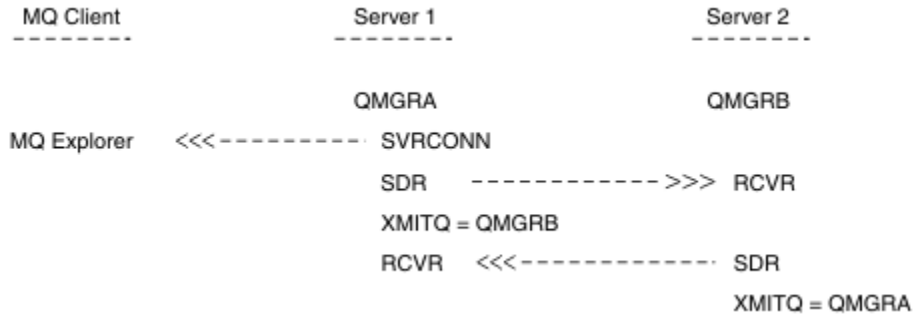
Se a conexão for, então, usada para conexão com o gerenciador de filas de destino por meio de um gerenciador de filas intermediário, o ID de usuário é transmitido no parâmetro *UserIdentifier* do descritor de mensagens (MQMD) novamente. Para que o listener MCA no gerenciador de filas de destino aceite essa mensagem, o atributo MCAUSER deve ser configurado ou o userId já deve existir com autoridade put.

O servidor de comandos no gerenciador de filas de destino coloca as mensagens para a fila de transmissão especificando o ID do usuário no parâmetro *UserIdentifier* no descritor de mensagens (MQMD). Para este put ser bem-sucedido, o ID do usuário já deve existir no gerenciador de filas de destino com autoridade put.

O exemplo a seguir mostra como conectar um gerenciador de filas, por um gerenciador de filas intermediário, ao WebSphere MQ Explorer.

Estabeleça uma conexão de administração remota para um gerenciador de filas. Verifique se o:

- O gerenciador de filas no servidor está ativo e tem um canal de conexão do servidor (SVRCONN) definido.
- O listener está ativo.
- O servidor de comandos está ativo.
- A fila SYSTEM.MQ EXPLORER.REPLY.MODEL foi criada e que você tenha autoridade suficiente.
- Os listeners do gerenciador de filas, servidores de comando e canais do emissor são iniciados.



Nesse exemplo:

- IBM WebSphere MQ Explorer está conectado ao gerenciador de filas QMGRA (em execução no Server1) utilizando uma conexão do cliente.
- O gerenciador de filas QMGRB no Server2 agora pode ser conectado ao IBM WebSphere MQ Explorer por meio de um gerenciador de filas intermediário (QMGRA)
- Ao conectar-se ao QMGRB com o WebSphere MQ Explorer, selecione QMGRA como o gerenciador de filas intermediário

Nesta situação, não há conexão direta para QMGRB do IBM WebSphere MQ Explorer; a conexão com o QMGRB é por meio de QMGRA.

O gerenciador de filas QMGRB em Server2 é conectado ao QMGRA em Server1 utilizando os canais do emissor do receptor. O canal entre o QMGRA e QMGRB devem ser configurados de tal modo que a administração remota é possível; consulte "Preparando canais e filas de transmissão para administração remota" na página 110.

Mostrando e ocultando gerenciadores de filas e clusters

O IBM WebSphere MQ Explorer pode exibir mais de um gerenciador de filas por vez. No painel Mostrar/Ocultar Gerenciador de Filas (selecionável a partir do menu para o nó da árvore Gerenciadores de Filas), é possível escolher se você exibirá informações sobre outra máquina (remota). Os gerenciadores de filas locais são detectados automaticamente.

Para mostrar um gerenciador de filas remotas:

1. Clique com o botão direito no nó da árvore `Queue Managers` e, em seguida, selecione *Mostrar / Ocultar Gerenciadores de Filas ...*.
2. Clique em **Incluir**. O painel Mostrar/Ocultar Gerenciadores de Filas é exibido.
3. Digite o nome do gerenciador de filas remotas e o nome do host ou endereço IP nos campos fornecidos.

   O nome do host ou o endereço IP é utilizado para estabelecer uma conexão do cliente com o gerenciador de filas remotas utilizando o seu canal de conexão do servidor padrão, SYSTEM.ADMIN.SVRCONN ou um servidor definido pelo usuário de conexão do canal.
4. Clique em **Concluir**.

O painel Mostrar/Ocultar Gerenciadores de Filas também exibe uma lista de todos os gerenciadores de filas visíveis. É possível utilizar esse painel para ocultar gerenciadores de filas a partir da visualização de navegação.

Se o IBM WebSphere MQ Explorer exibe um gerenciador de filas que é membro de um cluster, o cluster for detectado e exibido automaticamente.

Para exportar a lista de gerenciadores de filas remotas a partir deste painel:

1. Feche o painel Mostrar/Ocultar Gerenciadores de Filas.

2. Clique com o botão direito no nó da árvore do **IBM WebSphere MQ** superior na área de janela Navegação do WebSphere MQ Explorer, em seguida, selecione **Exportar MQ Configurações do Explorer**
3. Clique em **MQ Explorer> MQ Configurações do Explorer**
4. Selecione **Informações de conexão > Gerenciadores de filas remotas**.
5. Selecione um arquivo para armazenar as configurações exportadas.
6. Finalmente, clique em **Concluir** para exportar as informações de conexão do gerenciador de filas remotas para o arquivo especificado.

Para importar uma lista de gerenciadores de filas remotas:

1. Clique com o botão direito do mouse no nó da árvore do **IBM WebSphere MQ** superior na área de janela Navegação do WebSphere MQ Explorer e, em seguida, selecione **Importar MQ Configurações do Explorer**
2. Clique em **MQ Explorer> MQ Configurações do Explorer**
3. Clique em **Procurar** e navegue para o caminho do arquivo que contém as informações de conexão do gerenciador de filas remotas.
4. Click **Open**. Se o arquivo contiver uma lista de gerenciadores de filas remotas, a caixa **Informações de conexão > Gerenciadores de filas remotas** será selecionada.
5. Finalmente, clique em **Concluir** para importar as informações de conexão do gerenciador de filas remotas no WebSphere MQ Explorer.

Associação do cluster

IBM WebSphere MQ Explorer requer informações sobre gerenciadores de filas que são membros de um cluster.

Se um gerenciador de filas for um membro de um cluster, então, o nó da árvore de clusters será preenchido automaticamente.

Se os gerenciadores de filas se tornam membros de clusters enquanto o IBM WebSphere MQ Explorer está executando, então, deve-se manter o IBM WebSphere MQ Explorer com dados de administração atualizados sobre clusters para que ele possa se comunicar efetivamente com eles e exibir informações de cluster corretas quando solicitado. Para fazer isso, o WebSphere MQ Explorer precisa das seguintes informações:

- O nome de um gerenciador de filas de repositório
- O nome de conexão do gerenciador de filas do repositório, se ele estiver em um gerenciador de filas remotas

Com essas informações, o WebSphere MQ Explorer pode:

- Utilize o gerenciador de filas do repositório para obter uma lista de gerenciadores de filas no cluster.
- Administrar os gerenciadores de filas que são membros do cluster e estão em plataformas suportadas e níveis de comando.

Administração não é possível se:

- O repositório escolhido se torna indisponível. O WebSphere MQ Explorer não alterna automaticamente para um repositório alternativo.
- O repositório escolhido não pode ser contatado através de TCP/IP.
- O repositório escolhido está em execução em um gerenciador de filas que está em execução em uma plataforma e nível de comando não suportado pelo WebSphere MQ Explorer.

Os membros de cluster que podem ser administrados podem ser locais ou podem ser remotos se eles podem ser contactados utilizando TCP/IP. O IBM WebSphere MQ Explorer se conecta a gerenciadores de filas locais que são membros de um cluster diretamente, sem utilizar uma conexão do cliente.

Conversão de Dados

O IBM WebSphere MQ Explorer trabalha em CCSID 1208 (UTF-8). Isso permite que o IBM WebSphere MQ Explorer exiba os dados a partir de gerenciadores de filas corretamente. Indica se conectando a um gerenciador de filas diretamente ou utilizando um gerenciador de filas intermediário, o IBM WebSphere MQ Explorer requer que todas as mensagens recebidas sejam convertidos para CCSID 1208 (UTF-8).

Uma mensagem de erro será emitida se você tentar estabelecer uma conexão entre o IBM WebSphere MQ Explorer e um gerenciador de filas com um CCSID que o IBM WebSphere MQ Explorer não reconhece.

Conversões suportadas são descritas em Conversão da Página de Códigos.

Segurança em Windows

O assistente Preparar WebSphere MQ cria uma conta de usuário especial para que o serviço Windows possa ser compartilhado por processos que precisam usá-lo.

Um serviço é compartilhado entre os processos do cliente Windows para uma instalação do IBM WebSphere MQ. Um serviço é criado para cada instalação. Cada serviço é denominado `MQ_`*`InstallationName`*e possui um nome de exibição de `IBM WebSphere MQ(`*`InstallationName`*`)` Antes do Version 7.1, com apenas uma instalação em um servidor único, o serviço Windows foi nomeado `MQSeriesServices` com o nome de exibição `IBM MQSeries`

Como cada serviço deve ser compartilhado entre as sessões de logon não interativas e interativas, deve-se ativar cada uma delas sob uma conta de usuário especial. É possível usar uma conta do usuário especial para todos os serviços ou criar contas do usuário especiais diferentes. Cada conta de usuário especial deve ter o direito de usuário para "Efetuar logon como um serviço"; para obter mais informações, consulte “Direitos de usuário necessários para um serviço IBM WebSphere MQ Windows” na página 67. Se o ID do usuário não tiver a autoridade para executar o serviço, o serviço não será iniciado e retornará um erro no log de eventos do sistema Windows. Geralmente, você terá executado o assistente Preparar o IBM WebSphere MQ e configurado o ID de usuário corretamente. No entanto, se você configurou o ID do usuário manualmente, possivelmente ocorrerá um problema que precisará ser resolvido.

Ao instalar o IBM WebSphere MQ e executar o assistente Preparar IBM WebSphere MQ pela primeira vez, ele cria uma conta de usuário local para o serviço chamado MUSR_MQADMIN com as configurações e permissões requeridas, incluindo "Efetuar logon como um serviço".

Para instalações subsequentes, o assistente Preparar IBM WebSphere MQ cria uma conta de usuário denominada MUSR_MQADMINx, em que x é o número disponível próximo representa um ID de usuário que não existe. A senha para MUSR_MQADMINx é gerada aleatoriamente quando a conta é criada e utilizada para configurar o ambiente de logon para o serviço. A senha gerada não expira.

Esta conta do IBM WebSphere MQ não é afetada por quaisquer políticas de conta que são configuradas no sistema para exigir que as senhas das contas sejam mudadas após um determinado período.

A senha não é conhecida fora desse processamento único e é armazenada pelo sistema operacional Windows em uma parte segura do registro.

Usando o Active Directory (somenteWindows)

Em algumas configurações de rede, em que contas do usuário são definidas em controladores de domínio que estão usando Active Directory, a conta do usuário local sob a qual o IBM WebSphere MQ está executando pode não ter a autoridade requerida para consultar a associação ao grupo de outras contas de usuário do domínio. O assistente Preparar o IBM WebSphere MQ identifica se esse é o caso executando testes e fazendo perguntas para o usuário sobre a configuração de rede.

Se a conta do usuário local sob a qual o IBM WebSphere MQ está executando não tiver a autoridade necessária, o assistente de Preparar o IBM WebSphere MQ solicitará ao usuário detalhes de uma conta de usuário do domínio com direitos de usuário específico. Para obter os direitos do usuário que a conta do usuário do domínio requer, consulte “Direitos de usuário necessários para um serviço IBM WebSphere MQ Windows” na página 67. Depois que o usuário digitou detalhes da conta válida para a conta do usuário de domínio no Assistente de Preparação do IBM WebSphere MQ, ele configura um serviço do IBM

WebSphere MQ Windows para ser executado sob a nova conta. Os detalhes da conta são retidos na parte segura do Registro e não podem ser lidos pelos usuários.

Quando o serviço está em execução, um serviço do IBM WebSphere MQ Windows é iniciado e permanece em execução enquanto o serviço estiver em execução. Um administrador do IBM WebSphere MQ que efetua logon no servidor depois que o serviço do Windows é ativado pode usar o IBM WebSphere MQ Explorer para administrar gerenciadores de filas no servidor. Isso conecta o IBM WebSphere MQ Explorer para o processo de serviço existente do Windows . Essas duas ações precisam de diferentes níveis de permissão para que possam trabalhar:

- O processo de ativação requer uma permissão de ativação.
- O administrador do IBM WebSphere MQ requer permissão de acesso.

Direitos de usuário necessários para um serviço IBM WebSphere MQ Windows

A tabela neste tópico lista os direitos do usuário requeridos para a conta do usuário do domínio e local sob a qual o serviço do Windows para uma instalação do IBM WebSphere MQ é executada.

Efetuar logon como uma tarefa em lote	Permite que um serviço do IBM WebSphere MQ Windows seja executado sob essa conta do usuário
Efetue logon como serviço	Permite que os usuários configurem o serviço do IBM WebSphere MQ Windows para efetuar logon utilizando a conta configurada.
Desligar o sistema	Permite que o serviço IBM WebSphere MQ Windows reinicie o servidor, se estiver configurado para fazer isso quando a recuperação de um serviço falhar.
Aumentar cotas	Necessário para chamada de `CreateProcessAsUser` do sistema operacional.
Aja como parte do sistema operacional	Necessário para chamada de `LogonUser` do sistema operacional.
Verificação de passagem de desvio	Necessário para chamada de `LogonUser` do sistema operacional.
Substituir um símbolo em nível de processo	Necessário para chamada de `LogonUser` do sistema operacional.

Nota: Podem ser necessários direitos de programas de depuração em ambientes que executam os aplicativos ASP e IIS.

Sua conta do usuário do domínio deve ter esses direitos de usuário do Windows configurados como direitos de usuário efetivos, conforme listado no aplicativo Política de Segurança Local Se eles não forem, configure-os usando o aplicativo Política de Segurança Local localmente no servidor ou usando o domínio de Aplicativo de Segurança do Domínio amplo.

Mudando o nome do usuário associado ao Serviço do IBM WebSphere MQ

Pode ser necessário mudar o nome do usuário associado com o serviço do IBM WebSphere MQ MUSR_MQADMIN para algo diferente. (Por exemplo, pode ser necessário fazer isso se o seu gerenciador de filas estiver associado ao DB2, que não aceita nomes de usuário de mais de 8 caracteres.)

Procedimento

1. Crie uma nova conta do usuário (por exemplo **NEW_NAME**)
2. Use o Assistente para Preparar IBM WebSphere MQ para digitar os detalhes da nova conta de usuário.

Alterando a senha da conta do usuário do serviço IBM WebSphere MQ Windows

Sobre esta tarefa

Para mudar a senha do IBM WebSphere MQ Windows de serviço da conta de usuário local, execute as seguintes etapas:

Procedimento

1. Identifique o usuário do serviço está sendo executado.
2. Pare o serviço IBMIBM WebSphere MQ no painel Gerenciamento do Computador.
3. Mude a senha necessária da mesma maneira que você mudaria a senha de um indivíduo.
4. Acesse as propriedades para o serviço do IBM WebSphere MQ no painel Gerenciamento do Computador.
5. Selecione a página de **Efetuar logon**.
6. Confirme se o nome da conta especificado corresponde ao usuário para o qual a senha foi modificada.
7. Digite a senha no **Senha** e **Confirmar Senha** os campos e clique em **OK**.

Serviço IBM WebSphere MQ Windows para uma instalação em execução em uma conta do usuário do domínio

Sobre esta tarefa

Se o serviço do IBM WebSphere MQ Windows para uma instalação estiver executando sob uma conta de usuário de domínio, também é possível mudar a senha para a conta como se segue:

Procedimento

1. Mude a senha para a conta de domínio no controlador de domínio. Talvez seja necessário solicitar ao seu administrador de domínio para fazer isso para você.
2. Siga as etapas para modificar a página **Logon** para o serviço IBMIBM WebSphere MQ .

   A conta do usuário sob a qual executa o serviço IBM WebSphere MQ Windows executa quaisquer comandos MQSC que são emitidos por aplicativos de interface com o usuário ou executados automaticamente na inicialização, encerramento ou recuperação de serviço do sistema. Essa conta do usuário deve, portanto, ter direitos de administração do IBM WebSphere MQ. Por padrão, ele é incluído no grupo local **mqm** no servidor. Se essa associação for removida o serviço IBM WebSphere MQ Windows não funcionará. Para obter mais informações sobre os direitos de usuário, consulte “Direitos de usuário necessários para um serviço IBM WebSphere MQ Windows” na página 67

   Se um problema de segurança surgir com a conta do usuário sob a qual o serviço do IBM WebSphere MQ Windows executa, mensagens de erro e descrições aparecem no log de eventos do sistema.

Conceitos relacionados

“Usando o Active Directory (somenteWindows)” na página 66
Em algumas configurações de rede, em que contas do usuário são definidas em controladores de domínio que estão usando Active Directory, a conta do usuário local sob a qual o IBM WebSphere MQ está executando pode não ter a autoridade requerida para consultar a associação ao grupo de outras contas de usuário do domínio. O assistente Preparar o IBM WebSphere MQ identifica se esse é o caso executando testes e fazendo perguntas para o usuário sobre a configuração de rede.

IBM WebSphere MQ coordenando com Db2 como o gerenciador de recursos

Se você iniciar os seus gerenciadores de filas do IBM WebSphere MQ Explorer ou estiver usando o IBM WebSphere MQ V7 e estiver tendo problemas ao coordenar o Db2, verifique os logs de erros do gerenciador de filas.

Verifique os logs de erros do gerenciador de filas para um erro semelhante ao seguinte:

```
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.
```

Explicação: o ID do usuário (nome padrão é MUSR_MQADMIN) que executa o processo de Serviço do IBM WebSphere MQ `amqsvc.exe` ainda está em execução com um token de acesso que não contém informações de associação ao grupo para o grupo DB2USERS.

Resolver: após ter certeza de que o ID do usuário do Serviço IBM WebSphere MQ é um membro de DB2USERS, use a sequência de comandos a seguir:

- parar o serviço.
- parar quaisquer outros processos sendo executados sob o mesmo ID de usuário.
- reinicie estes processos.

Reinicializar a máquina asseguraria as etapas anteriores, mas não é necessário.

Ampliando o IBM WebSphere MQ Explorer

IBM WebSphere MQ para Windowse IBM WebSphere MQ para Linux (plataformasx86 e x86-64) fornecem uma interface de administração chamada IBM WebSphere MQ Explorer para executar tarefas de administração como uma alternativa para usar comandos de controle ou MQSC.

Estas informações se aplicam ao WebSphere MQ para Windows e ao WebSphere MQ para Linux (plataformasx86 e x86-64) apenas

O IBM WebSphere MQ Explorer apresenta informações em um estilo consistente com a estrutura Eclipse e os outros aplicativos de plug-in que o Eclipse suporta.

Por meio da extensão do IBM WebSphere MQ Explorer, os administradores do sistema têm a capacidade de customizar o WebSphere MQ Explorer para melhorar a maneira como administram o WebSphere MQ.

Para obter mais informações, consulte *Estendendo o IBM WebSphere MQ Explorer* na documentação do produto do IBM WebSphere MQ Explorer

Usando o aplicativo da barra de tarefas IBM WebSphere MQ (somente Windows)

O aplicativo barra de tarefas IBM WebSphere MQ exibe um ícone na bandeja do sistema Windows no servidor. O ícone fornece a você o status atual do IBM WebSphere MQ e um menu a partir do qual é possível executar algumas ações simples.

No Windows, o ícone WebSphere MQ está na bandeja do sistema no servidor e é sobreposto com um símbolo de status codificado por cores, que pode ter um dos seguintes significados:

Verde
Trabalhando corretamente; nenhum alerta no presente

Blue
Indeterminado; WebSphere MQ está inicializando ou encerrando

Amarela
Alerta; um ou mais serviços estão falhando ou já falharam

Para exibir o menu, clique com o botão direito no ícone WebSphere MQ . No menu, é possível executar as seguintes ações:

- Clique em **Abrir** para abrir o WebSphere MQ Alert Monitor
- Clique em **Sair** para sair do aplicativo da barra de tarefas WebSphere MQ
- Clique em **WebSphere MQ Explorer** para iniciar o IBM WebSphere MQ Explorer..

- Clique em **Parar WebSphere MQ** para parar WebSphere MQ
- Clique em **Sobre WebSphere MQ** para exibir informações sobre o Monitor de Alerta do WebSphere MQ

O aplicativo do monitor de alertas IBM WebSphere MQ (apenas Windows)

O monitor de alertas do IBM WebSphere MQ é uma ferramenta de detecção de erro que identifica e regista problemas com o IBM WebSphere MQ em uma máquina local.

O monitor de alertas exibe informações sobre o status atual da instalação local de um servidor WebSphere MQ . Ele também monitora o Windows Advanced Configuration and Power Interface (ACPI) e assegura que as configurações de ACPI sejam aplicadas.

No monitor de alertas do WebSphere MQ , é possível:

- Acesse o IBM WebSphere MQ Explorer diretamente
- Visualizar as informações relacionadas a todos os alertas pendentes
- Encerre o serviço WebSphere MQ na máquina local
- Rotear mensagens de alerta por meio da rede para uma conta do usuário configurável ou para uma estação de trabalho ou servidor Windows

Administrando objetos locais do IBM WebSphere MQ

Esta seção informa como administrar objetos locais do IBM WebSphere MQ para suportar programas de aplicativos que utilizam uma MQI (Message Queue Interface). Nesse contexto, a administração local significa criar, exibir, mudar, copiar e excluir os objetos do IBM WebSphere MQ.

Além das abordagens detalhadas nesta seção, é possível usar o IBM WebSphere MQ Explorer para administrar objetos WebSphere MQ locais; consulte “Administração usando o IBM WebSphere MQ Explorer” na página 57.

Esta seção inclui as informações a seguir:

Iniciando e Parando um Gerenciador de Filas

Use este tópico como uma introdução para parar e iniciar um gerenciador de filas.

Iniciando um Gerenciador de Filas

Para iniciar um gerenciador de filas, use o comando `strmqm` conforme a seguir:

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

No WebSphere MQ para Windows e WebSphere MQ para Linux (plataformasx86 e x86-64), é possível iniciar um gerenciador de filas da seguinte forma:

1. Abra o IBM WebSphere MQ Explorer.
2. Selecione o gerenciador de filas da Visualização do Navegador.
3. Clique em `Start`. O gerenciador de filas é iniciado.

Se a inicialização do gerenciador de filas levar mais de alguns segundos, o WebSphere MQ emitirá mensagens de informações intermitentemente detalhando o progresso da inicialização.

O comando `strmqm` não retorna o controle até que o gerenciador de filas tenha sido iniciado e esteja pronto para aceitar solicitações de conexão.

Iniciando um Gerenciador de Filas Automaticamente

No WebSphere MQ for Windows , é possível iniciar um gerenciador de filas automaticamente quando o sistema é iniciado usando o WebSphere MQ Explorer Para obter mais informações, consulte “Administração usando o IBM WebSphere MQ Explorer” na página 57.

Parando um Gerenciador de Filas

Use o comando **endmqm** para parar um gerenciador de filas.

Nota: Deve-se usar o comando **endmqm** a partir da instalação associada ao gerenciador de filas com o qual você está trabalhando. É possível descobrir com qual instalação um gerenciador de filas está associado usando o comando `dspmq -o installation`.

Por exemplo, para parar um gerenciador de filas chamado QMB, insira o seguinte comando:

```
endmqm QMB
```

Em sistemas WebSphere MQ para Windows e WebSphere MQ para Linux (plataformasx86 e x86-64), é possível parar um gerenciador de filas conforme a seguir:

1. Abra o IBM WebSphere MQ Explorer.
2. Selecione o gerenciador de filas da Visualização do Navegador.
3. Clique em `Stop...` O painel Terminar Gerenciador de Filas é exibido.
4. Selecione Controlado ou Imediato.
5. Clique em OK. O gerenciador de filas é parado.

Encerramento em Modo Quiesce

Por padrão, o comando **endmqm** executa um encerramento em modo quiesce do gerenciador de filas especificado. Isso pode levar algum tempo para ser concluído. Um encerramento em modo quiesce aguarda até que todos os aplicativos conectados sejam desconectados.

Use esse tipo de encerramento para notificar os aplicativos para parar. Se você emitir:

```
endmqm -c QMB
```

você não saberá quando todos os aplicativos forem interrompidos. (Um comando `endmqm -c QMB` é equivalente a um comando `endmqm QMB` .

Porém, se você emitir:

```
endmqm -w QMB
```

o comando aguardará até que todos os aplicativos tenham sido interrompidos e o gerenciador de filas tenha sido terminado.

Encerramento Imediato

Para um encerramento imediato, todas as chamadas MQI atuais têm permissão para conclusão, mas as chamadas novas falham. Esse tipo de encerramento não aguarda os aplicativos se desconectarem do gerenciador de filas.

Para um encerramento imediato, digite:

```
endmqm -i QMB
```

Encerramento Preemptivo

Nota: Não use esse método, a menos que todas as outras tentativas de parar o gerenciador de filas usando o comando **endmqm** tenham falhado. Esse método pode ter consequências imprevisíveis para aplicativos conectados.

Se um encerramento imediato não funcionar, deve-se recorrer a um encerramento *preemptivo* especificando o sinalizador `-p`. Por exemplo:

```
endmqm -p QMB
```

Isso para o gerenciador de filas imediatamente. Se esse método não funcionar, consulte "Parando um gerenciador de filas manualmente" na página 72 para conhecer uma solução alternativa.

Para obter uma descrição detalhada do comando **endmqm** e suas opções, consulte endmqm.

Em Caso de Problemas para Encerrar um Gerenciador de Filas

Muitas vezes os problemas para encerrar um gerenciador de filas são causados por aplicativos. Por exemplo, quando aplicativos:

- Não verificam códigos de retorno de MQI corretamente
- Não solicitam notificação de um quiesce
- São finalizados sem se desconectarem do gerenciador de filas (emitindo uma chamada `MQDISC`)

Se ocorrer um problema quando você parar o gerenciador de filas, é possível sair do comando **endmqm** usando Ctrl-C. É possível emitir outro comando **endmqm**, mas dessa vez com um sinalizador que especifique o tipo de encerramento que você requer.

Parando um gerenciador de filas manualmente

Se os métodos padrão para parar os gerenciadores de filas falhar, tente os métodos descritos aqui.

A maneira padrão de parar os gerenciadores de filas é utilizando o comando `endmqm`. Para parar um gerenciador de filas manualmente, utilize um dos procedimentos descritos nesta seção. Para obter detalhes sobre como executar operações em gerenciadores de fila usando comandos de controle, consulte Criando ou gerenciando gerenciadores de fila.

Parando gerenciadores de filas no Windows

Como terminar os processos e o serviço do IBM WebSphere MQ para parar os gerenciadores de fila no IBM WebSphere MQ para Windows

Para parar um gerenciador de filas em execução no WebSphere MQ para Windows:

1. Liste os nomes (IDs) dos processos que estão em execução, usando o Gerenciador de Tarefas do Windows ...
2. Termine os processos usando o Gerenciador de Tarefas do Windows ou o comando **taskkill** na ordem a seguir (se eles estiverem em execução):

AMQZMUC0	Gerenciador de processos crítico
AMQZXMA0	Controlador de execução
AMQZFUMA	Processo OAM
AMQZLAA0	Agentes LQM
AMQZLSA0	Agentes LQM
AMQZMUF0	Gerenciador de Utilitários

AMQZMGR0	Controlador de Processos
AMQZMUR0	Gerenciador de processos reinicializável
AMQFQPUB	Processo de publicação/assinatura
AMQFCXBA	Processo do trabalhador do broker
AMQRMPPA	Processo do conjunto de processos
AMQCRSTA	Processo de tarefa do respondente não encadeado
AMQCRS6B	Canal receptor da LU62 e conexão do cliente
AMQRRMFA	Processo do repositório (para clusters)
AMQZDMAA	Processador de mensagens adiado
AMQPCSEA	O servidor de comandos
RUNMQTRM	Chamar um monitor acionador para um servidor
RUNMQDLQ	Chamar manipulador da fila de devoluções
RUNMQCHI	Processo do inicializador de canais
RUNMQLSR	Processo do listener de canal
AMQXSSVN	Servidores de memória compartilhada
AMQZTRCN	Rastreio

3. Pare o serviço do WebSphere MQ a partir de **Ferramentas de Administração** > **Serviços** no Painel de Controle do Windows
4. Se você tiver tentado todos os métodos e o gerenciador de filas não foi interrompido, reinicialize o sistema.

O Windows Task Manager e o comando **tasklist** fornecem informações limitadas sobre tarefas. Para obter mais informações para ajudar a determinar quais processos estão relacionados a um gerenciador de filas específico, considere usar uma ferramenta como *Process Explorer* (procexp.exe), disponível para download no website da Microsoft em https://www.microsoft.com.

Parando gerenciadores de filas em sistemas UNIX and Linux

Como terminar os processos e o serviço IBM WebSphere MQ , para parar os gerenciadores de fila no IBM WebSphere MQ para UNIX and Linux. É possível tentar os métodos descritos aqui, se os métodos padrão para parar e remover gerenciadores de filas falharem.

Para parar um gerenciador de fila em execução em sistemas WebSphere MQ para UNIX and Linux :

1. Localize os IDs de processo do gerenciador de filas que ainda estão em execução programas usando o comando `ps`. Por exemplo, se o gerenciador de filas é chamado `QMNAME`, use o seguinte comando:

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

2. Termine quaisquer processos do gerenciador de filas que ainda estejam executando. Use o comando `kill`, especificando os IDs de processo descoberto utilizando o comando `ps`.

   Encerre os processos na seguinte ordem:

amqzmuc0	Gerenciador de processos crítico
amqzxma0	Controlador de execução
amqzfuma	Processo OAM
amqzlaa0	Agentes LQM
amqzlsa0	Agentes LQM

amqzmuf0	Gerenciador de Utilitários
amqzmur0	Gerenciador de processos reinicializável
amqzmgr0	Controlador de Processos
amqfqpub	Processo de publicação/assinatura
amqfcxba	Processo do trabalhador do broker
amqrmppa	Processo do conjunto de processos
amqcrsta	Processo de tarefa do respondente não encadeado
amqcrs6b	Canal receptor da LU62 e conexão do cliente
amqrrmfa	Processo do repositório (para clusters)
amqzdmaa	Processador de mensagens adiado
amqpcsea	O servidor de comandos
runmqtrm	Chamar um monitor acionador para um servidor
runmqdlq	Chamar manipulador da fila de devoluções
runmqchi	Processo do inicializador de canais
runmqlsr	Processo do listener de canal

Nota: É possível utilizar o comando **`kill -9`** para finalizar processos que falham ao parar.

Se você parar o gerenciador de filas manualmente, FFSTs poderão ser obtidos e arquivos FDC colocados em `/var/mqm/errors`. Não considere isso como um defeito no gerenciador de filas.

O gerenciador de filas será reiniciado normalmente, mesmo após você ter parado, utilizando este método.

Executando tarefas de administração locais usando comandos MQSC

Esta seção apresenta a você comandos MQSC e explica como utilizá-los para algumas tarefas comuns.

Se você usar IBM WebSphere MQ para Windows ou IBM WebSphere MQ para Linux (plataformasx86 e x86-64), também poderá executar as operações descritas nesta seção usando o IBM WebSphere MQ Explorer. Consulte a “Administração usando o IBM WebSphere MQ Explorer” na página 57 para obter mais informações.

É possível usar comandos MQSC para gerenciar objetos do gerenciador de filas, incluindo o próprio gerenciador de filas, filas, definições de processo, canais, canais de conexão do cliente, listeners, serviços, listas de nomes, clusters e objetos de informações de autenticação. Esta seção trata de gerenciadores de filas, filas e definições de processo; para obter informações sobre como administrar o canal, o canal de conexão do cliente e objetos do listener, consulte Objetos . Para obter informações sobre todos os comandos MQSC para gerenciar objetos do gerenciador de filas, consulte “Comandos do Script (MQSC)” na página 75.

Você emite comandos do MQSC para um gerenciador de filas usando o comando `runmqsc`. (Para obter detalhes deste comando, consulte runmqsc.) É possível fazer isso interativamente, emitindo comandos a partir de um teclado ou redirecionar o dispositivo de entrada padrão (`stdin`) para executar uma sequência de comandos a partir de um arquivo de texto ASCII. Em ambos os casos, o formato dos comandos é o mesmo. (Para obter informações sobre como executar os comandos a partir de um arquivo de texto, consulte “Executando comandos MQSC a partir de arquivos de texto” na página 78.)

É possível executar o comando `runmqsc` de três maneiras, dependendo do conjunto de sinalizadores no comando:

- Verifique um comando sem executá-lo, em que os comandos MQSC são verificados em um gerenciador de filas locais, mas não são executados.
- Execute um comando em um gerenciador de filas locais, em que os comandos MQSC são executados em um gerenciador de filas locais.

- Execute um comando em um gerenciador de filas remotas, em que os comandos MQSC são executados em um gerenciador de filas remoto.

Também é possível executar o comando seguido por um ponto de interrogação para exibir a sintaxe.

Os atributos de objeto especificados nos comandos MQSC são mostrados nesta seção em maiúsculas (por exemplo, RQMNAME), embora não façam distinção entre maiúsculas e minúsculas. Os nome de atributo do comando MQSC são limitados a oito caracteres. Comandos MQSC estão disponíveis em outras plataformas, incluindo IBM i e z/OS.

Os comandos MQSC são resumidos na coleta de tópicos na seção Referência de MQSC.

Comandos do Script (MQSC)

Os comandos MQSC fornecem um método uniforme de emitir comandos legíveis nas plataformas WebSphere MQ . Para obter informações sobre os comandos *Programmable Command Format* (PCF), consulte “Introdução aos Formatos de Comando Programável” na página 9.

O formato geral dos comandos é mostrado em Os Comandos MQSC.

Deve-se observar as seguintes regras ao usar os comandos MQSC:

- Cada comando inicia com um parâmetro primário (um verbo) e isso é seguido por um parâmetro secundário (um nome). Em seguida, isso é seguido pelo nome ou nome genérico do objeto (entre parênteses), se houver um, que existe na maioria dos comandos. Seguindo isso, os parâmetros geralmente podem ocorrer em qualquer ordem; se um parâmetro tiver um valor correspondente, o valor deve ocorrer diretamente após o parâmetro ao qual se relaciona.
- As palavras-chave, os parênteses e os valores podem ser separados por qualquer número de espaços em branco e vírgulas. Uma vírgula mostrada nos diagramas de sintaxe sempre pode ser substituída por um ou mais espaços em branco. Deve haver pelo menos um espaço em branco imediatamente antes de cada parâmetro (após o parâmetro primário)
- Pode ocorrer qualquer número de espaços em branco no início ou término do comando e entre parâmetros, pontuação e valores. Por exemplo, o seguinte comando é válido:

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

  Os espaços em branco em um par de marcas de aspas são significativos.
- As vírgulas adicionais podem aparecer em qualquer lugar em que os espaços em branco forem permitidos e tratados como se fossem espaços em branco (a menos que estejam, obviamente, dentro de sequências entre aspas).
- Não são permitidos parâmetros repetidos. Repetir um parâmetro com sua versão "NO", como em REPLACE NOREPLACE, também não é permitido.
- As sequências que contém espaços em branco, caracteres minúsculos ou caracteres especiais que não sejam:
  - Ponto (.)
  - Barra (/)
  - Sublinhado (_)
  - Sinal de percentual (%)

  devem estar entre aspas simples, a menos que sejam:
  - Os valores genéricos que terminam com um asterisco
  - Um único asterisco (por exemplo, TRACE(*))
  - Uma especificação de intervalo que contém um dois pontos (por exemplo, CLASS(01:03))

  Se a própria sequência contiver aspas simples, ela será representada por duas aspas simples.Os caracteres minúsculos não contidos nas aspas são convertidos em letras maiúsculas.

- Em plataformas diferentes de z/OS, uma sequência contendo nenhum caractere (ou seja, duas aspas simples sem espaço entre elas) é interpretada como um espaço em branco entre aspas simples, ou seja, interpretado da mesma maneira que (''). A exceção para isto é se o atributo que está sendo usado for um dos seguintes:
  - TOPICSTR
  - SUB
  - USERDATA
  - SELECTOR

  em seguida, duas aspas simples sem espaço serão interpretadas como uma sequência de comprimento zero.
- No v7.0, qualquer rastreio de espaço em branco nesses atributos de sequência que for baseado nos tipos MQCHARV, como SELECTOR, dados de sub-usuário, é tratado como significativo o que significa que 'abc ' não equivale a 'abc'.
- Um parêntese esquerdo seguido por um parêntese direito, sem informações significativas intermediárias, por exemplo

  ```
  NAME ( )
  ```

  não é válido, exceto onde observado especificamente.
- As palavras-chave não fazem distinção entre maiúsculas e minúsculas: AltERAR, alterar e ALTERAR são todas aceitáveis.Tudo o que não estiver contido nas aspas é convertido em letras maiúsculas.
- Os sinônimos são definidos para alguns parâmetros.Por exemplo, o DEF é sempre um sinônimo para DEFINE; portanto, DEF QLOCAL é válido. No entanto, os sinônimos não são apenas sequências mínimas; DEFI não é um sinônimo válido para DEFINE.

  Nota: Não há sinônimo para o parâmetro DELETE. Isso evita a exclusão acidental de objetos ao usar DEF, o sinônimo para DEFINE.

Para obter uma visão geral do uso de comandos MQSC para administrar IBM WebSphere MQ, consulte “Executando tarefas de administração locais usando comandos MQSC” na página 74.

Os comandos MQSC usam certos caracteres especiais para ter certos significados. Para obter informações adicionais sobre esses caracteres especiais e como usá-los, consulte Caracteres com Significado Especial.

Para descobrir como é possível construir scripts usando comandos MQSC, consulte Construindo Scripts de Comando.

Para obter a lista completa de comandos MQSC, consulte Comandos MQSC.

Tarefas relacionadas
Construindo Scripts de Comando

WebSphere MQ nomes de objetos

Como usar nomes de objetos em comandos MQSC.

Nos exemplos, utilizamos alguns nomes longos para objetos. Isso serve para ajudá-lo a identificar o tipo de objeto que você está lidando.

Ao emitir os comandos do MQSC, é necessário especificar somente o nome local da fila. Em nossos exemplos, utilizamos os nomes de filas, tais como:

```
ORANGE.LOCAL.QUEUE
```

A parte `LOCAL.QUEUE` do nome é para ilustrar que esta fila é uma fila local. ***Não*** é necessário para os nomes de filas locais em geral.

Também utilizamos o nome `saturn.queue.manager` como um nome do gerenciador de filas. A parte `queue.manager` do nome é para ilustrar que este objeto é um gerenciador de filas. *Não* é necessário para os nomes de gerenciadores de filas em geral.

Distinção entre maiúsculas e minúsculas nos comandos MQSC

Os comandos MQSC, incluindo seus atributos, podem ser gravados em letras maiúsculas ou minúsculas. Os nomes de objetos nos comandos MQSC são convertidos para maiúsculas (ou seja, QUEUE e filas não são diferenciados), a menos que os nomes sejam agrupados dentro de aspas simples. Se as aspas não forem utilizadas, o objeto é processado com um nome em letras maiúsculas. Consulte oReferência MQSC para obter mais informações

A chamada do comando `runmqsc` , em comum com todos os comandos de controle WebSphere MQ , faz distinção entre maiúsculas e minúsculas em alguns ambientes do WebSphere MQ . Consulte Usando comandos de controle para obter mais informações.

Entrada e saída padrão

O *padrão do dispositivo de entrada*, também referido como `stdin`, é o dispositivo a partir do qual a entrada para o sistema será executada. Geralmente este é o teclado, mas é possível especificar que a entrada virá de uma porta serial ou um arquivo de disco, por exemplo. O *dispositivo de saída padrão*, também referido como `stdout`, é o dispositivo para o qual a saída do sistema é enviada. Geralmente esse é um vídeo, mas é possível redirecionar a saída para uma porta serial ou um arquivo.

Em comandos do sistema operacional e comandos de controle do WebSphere MQ , o operador < redireciona a entrada.. Se este operador é seguido por um nome de arquivo, a entrada é retirada do arquivo. Da mesma forma, o operador > redireciona a saída; se esse operador for seguido por um nome de arquivo, a saída será direcionada para esse arquivo.

Usando comandos MQSC interativamente

É possível utilizar os comandos do MQSC de forma interativa, utilizando uma janela de comando ou shell.

Para utilizar comandos MQSC interativamente, abra uma janela de comandos ou shell e digite:

```
runmqsc
```

Nesse comando, um nome de gerenciador de filas não foi especificado, assim, os comandos MQSC são processados pelo gerenciador de filas padrão. Se desejar usar um gerenciador de filas diferente, especifique o nome do gerenciador de filas no comando **runmqsc**. Por exemplo, para executar comandos MQSC no gerenciador de filas `jupiter.queue.manager`, use o comando:

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

Depois disso, todos os comandos MQSC que você digitar serão processados por este gerenciador de filas, supondo que ele esteja no mesmo nó e já esteja em execução.

Agora é possível digitar qualquer comando MQSC, conforme necessário. Por exemplo, tente este:

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

Para comandos que possuem parâmetros em excesso para caber em uma única linha, utilize caracteres de continuação para indicar que um comando é continuado na linha a seguir:

- Um sinal de menos (-) indica que o comando deve ser continuado a partir do início da linha seguinte.
- Um sinal de mais (+) indica que o comando deve ser continuado do primeiro caractere não em branco na linha seguinte.

A entrada de comandos termina com o caractere final de uma linha não em branco que não é um caractere de continuação. É possível também finalizar de entrada de comando explicitamente digitando

um ponto e vírgula (;). (Isso é especialmente útil se você por engano digitar um caractere de continuação no término da linha final do comando de entrada.)

Feedback de comandos MQSC

Ao emitir os comandos do MQSC, o gerenciador de filas retorna mensagens do operador que confirmam suas ações ou informam sobre os erros que você cometeu. Por exemplo:

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

Esta mensagem confirma que uma fila foi criada.

```
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
```

Esta mensagem indica que você cometeu um erro de sintaxe.

Estas mensagens são enviadas para o dispositivo de saída padrão. Se você não tiver inserido o comando corretamente, consulte o Referência MQSC para a sintaxe correta.

Terminando a entrada interativa de comandos MQSC

Para parar o trabalho com os comandos do MQSC, digite o comando END.

Como alternativa, é possível utilizar o caractere de EOF para seu sistema operacional.

Executando comandos MQSC a partir de arquivos de texto

Executar comandos MQSC interativamente é adequado para testes rápidos, mas se você tiver comandos muito longos ou estiver utilizando uma determinada sequência de comandos repetidamente, considere redirecionar `stdin` a partir de um arquivo de texto.

"Entrada e saída padrão" na página 77 contém informações sobre `stdin` e `stdout`. Para redirecionar `stdin` a partir de um arquivo de texto, primeiro crie um arquivo de texto que contenha os comandos do MQSC usando seu editor de texto normal. Ao usar o comando `runmqsc`, utilize os operadores de redirecionamento. Por exemplo, o comando a seguir executa uma sequência de comandos contida no arquivo de texto `myprog.in`:

```
runmqsc < myprog.in
```

Da mesma forma, também é possível redirecionar a saída para um arquivo. Um arquivo que contém os comandos MQSC para a entrada é denominado um *arquivo de comandos MQSC*. O arquivo de saída contendo as respostas do gerenciador de filas é chamado de *arquivo de saída*.

Para redirecionar ambos `stdin` e `stdout` no comando `runmqsc`, utilize este formulário do comando:

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

Este comando chama os comandos MQSC contidos no arquivo de comando MQSC `myprog.in`. Como não especificamos um nome de gerenciador de fila, os comandos MQSC são executados no gerenciador de filas padrão. A saída é enviada para o arquivo de texto `myprog.out`.Figura 12 na página 79 Mostra uma extração do arquivo de comando MQSC `myprog.in` e Figura 13 na página 80 mostra a extração correspondente da saída em `myprog.out`

Para redirecionar `stdin` e `stdout` no comando `runmqsc` , para um gerenciador de filas (`saturn.queue.manager`) que não seja o padrão, use esta forma do comando:

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

Arquivos de comando MQSC

Os comandos MQSC são gravados em forma legível aos humanos, isto é, em texto ASCII.Figura 12 na página 79 é uma extração de um arquivo de comando MQSC que mostra um comando MQSC (DEFINE QLOCAL) com seus atributos. O Referência MQSC contém uma descrição de cada comando MQSC e sua sintaxe.

```
  .
  .
  .
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. Extraia a partir de um arquivo de comandos MQSC

Para portabilidade entre ambientes do WebSphere MQ , limite o comprimento da linha nos arquivos de comando MQSC para 72 caracteres.. O sinal de mais indica que o comando é continuado na próxima linha.

Relatórios de comando do MQSC

O comando **runmqsc** retorna um *report*, que é enviado para `stdout`. O relatório contém:

- Um cabeçalho identificar comandos MQSC como a origem do relatório:

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

  em que `jupiter.queue.manager` é o nome do gerenciador de filas.
- Uma listagem numerada opcional dos comandos MQSC emitidos. Por padrão, o texto da entrada é ecoado para a saída. Nessa saída, cada comando é prefixado por um número de sequência, conforme mostrado em Figura 13 na página 80. No entanto, é possível utilizar o sinalizador -e no comando `runmqsc` para suprimir a saída.

- Uma mensagem de erro de sintaxe para quaisquer comandos que estejam com erro.
- Uma *mensagem do operador*, indicando o resultado da execução de cada comando. Por exemplo, a mensagem do operador para a conclusão bem-sucedida de um comando DEFINE QLOCAL é:

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

- Outras mensagens resultantes de erros gerais ao executar o arquivo de script.
- Um breve resumo estatístico do relatório indicando o número de comandos de leitura, o número de comandos com erros de sintaxe e o número de comandos que não puderam ser processados.

  Nota: O gerenciador de filas tenta processar somente aqueles comandos que não possuem erros de sintaxe.

```
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. Extração de um arquivo de relatório de comandos MQSC

Executando os arquivos de comandos MQSC fornecidos

Os seguintes arquivos de comando MQSC são fornecidos com o WebSphere MQ:

amqscos0.tst
As definições de objetos usados por programas de amostra.

amqscic0.tst
Definições de filas para transações CICS .

No WebSphere MQ para Windows, estes arquivos estão localizados no diretório *MQ_INSTALLATION_PATH*\tools\mqsc\samples. *MQ_INSTALLATION_PATH* . Representa o diretório de alto nível no qual o WebSphere MQ está instalado

Nos sistemas UNIX and Linux , esses arquivos estão localizados no diretório *MQ_INSTALLATION_PATH*/samp. *MQ_INSTALLATION_PATH* Representa o diretório de alto nível no qual o WebSphere MQ está instalado

O comando que executa os é:

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

Utilizar runmqsc para verificar comandos

É possível utilizar o comando `runmqsc` para verificar os comandos MQSC em um gerenciador de filas locais sem realmente executá-los. Para fazer isso, configure o sinalizador -v no comando `runmqsc`, por exemplo:

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

Ao chamar `runmqsc` em um arquivo de comando do MQSC, o gerenciador de filas verifica cada comando e retorna um relatório sem realmente executar os comandos MQSC. Isso permite verificar a sintaxe dos comandos em seu arquivo de comando. Isto é particularmente importante se você estiver:

- Executando um grande número de comandos a partir de um arquivo de comando.
- Utilizando um arquivo de comandos MQSC várias vezes.

O relatório retornado é semelhante àquele mostrado em Figura 13 na página 80.

Não é possível usar este método para verificar se os comandos MQSC remotamente. Por exemplo, se você tentar este comando:

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

o sinalizador -w, que você utiliza para indicar que o gerenciador de filas for remoto, é ignorado e o comando é executado localmente no modo de verificação. 30 é o número de segundos que o WebSphere MQ aguarda respostas do gerenciador de fila remoto..

Executando comandos MQSC a partir de arquivos em lote

Se você tiver os comandos muito longos ou estiver utilizando uma determinada sequência de comandos repetidamente, considere redirecionar `stdin` a partir de um arquivo em lote.

Para redirecionar `stdin` a partir de um arquivo em lote, primeiro crie um arquivo em lote que contém os comandos do MQSC usando seu editor de texto comum. Ao usar o comando `runmqsc`, utilize os operadores de redirecionamento. O exemplo a seguir:

1. Cria um gerenciador de filas de teste, TESTQM
2. Cria um CLNTCONN correspondente e o listener configurado para utilizar a porta TCP/IP 1600
3. Cria uma fila de teste, TESTQ
4. Coloca uma mensagem na fila, utilizando o programa de amostra 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. Exemplo de script para executar os comandos do MQSC a partir de um arquivo em lote

Resolvendo problemas com os comandos do MQSC

Se você não puder obter os comandos MQSC para executar, utilize as informações neste tópico para ver se qualquer um desses problemas comuns se aplicam a você. Nem sempre é evidente que o problema é quando você lê o erro que um comando gera.

Se não for possível obter comandos MQSC para execução, use as informações a seguir para ver se algum desses problemas comuns se aplica a você. Nem sempre é óbvio qual é o problema quando você lê o erro gerado

Ao usar o comando `runmqsc`, lembre-se do seguinte:

- Use o operador **<** para redirecionar a entrada de um arquivo. Se você omitir este operador, o gerenciador de filas interpreta o nome do arquivo como um nome do gerenciador de filas e emite a seguinte mensagem de erro:

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

- Se você redirecionar a saída para um arquivo, use o operador de redirecionamento **>**. Por padrão, o arquivo é colocado no diretório de trabalho atual no momento `runmqsc` é chamado. Especifique um nome de arquivo completo para enviar a saída para um arquivo específico e diretório.
- Verifique se você tiver criado o gerenciador de filas que está indo para executar os comandos, usando o comando a seguir para exibir todos os gerenciadores de filas:

  ```
  dspmq
  ```

- O gerenciador de filas deve estar em execução. Se não estiver, inicie-o; (consulte Iniciando um gerenciador de filas). Você receberá uma mensagem de erro se tentar iniciar um gerenciador de filas que já esteja em execução.
- Especifique um nome do gerenciador de filas no comando `runmqsc` se você não tiver definido um gerenciador de filas padrão ou você receber este erro:

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

- Não é possível especificar um comando do MQSC como um parâmetro do comando `runmqsc`. Por exemplo, isso não é válido:

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

- Não é possível digitar os comandos do MQSC antes de emitir o comando `runmqsc`.
- Não é possível executar comandos de controle do `runmqsc`. Por exemplo, você não pode emitir o comando `strmqm` para iniciar um gerenciador de filas enquanto você estiver executando comandos MQSC interativamente. Se você fizer isto, receberá mensagens de erro semelhantes ao seguinte:

```
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
```

Trabalhando com Gerenciadores de Fila

Exemplos de comandos MQSC que é possível usar para exibir ou alterar os atributos do gerenciador de filas.

Exibindo Atributos do Gerenciador de Filas

Para exibir os atributos do gerenciador de filas especificado no comando **runmqsc**, use o seguinte comando MQSC:

```
DISPLAY QMGR
```

A saída típica desse comando é mostrada em Figura 15 na 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. Saída Típica de um Comando DISPLAY QMGR

O parâmetro ALL é o padrão no comando DISPLAY QMGR. Ele exibe todos os atributos do gerenciador de filas. Em particular, a saída informa o nome do gerenciador de filas padrão, o nome da fila de devoluções e o nome da fila de comandos.

É possível confirmar se essas filas existem, inserindo o comando:

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

Isso exibe uma lista de filas que correspondem à raiz SYSTEM.*.. Os parênteses são necessários.

Mudando os Atributos do Gerenciador de Filas

Para alterar os atributos do gerenciador de filas especificado no comando **runmqsc**, use o comando ALTER QMGR do MQSC, especificando os atributos e valores que você deseja alterar. Por exemplo, use os seguintes comandos para alterar os atributos de `jupiter.queue.manager`:

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

O comando ALTER QMGR altera a fila de devoluções usada e permite eventos de inibição.

Referências relacionadas

Atributos do gerenciador de filas

Trabalhando com Filas Locais

Esta seção contém exemplos de alguns comandos MQSC que podem ser usados para gerenciar filas locais, de modelo e de alias.

Consulte o Referência MQSC para obter informações detalhadas sobre esses comandos.

Definindo uma fila local

Para um aplicativo, o gerenciador de filas locais é o gerenciador de filas ao qual o aplicativo está conectado. As filas gerenciadas pelo gerenciador de filas locais são consideradas como locais para esse gerenciador de filas.

Use o comando do MQSC DEFINE QLOCAL para criar uma fila local. Também é possível utilizar o padrão definido na definição de fila local padrão ou é possível modificar as características da fila a partir daqueles da fila local padrão.

Nota: A fila local padrão é chamada SYSTEM.DEFAULT.LOCAL.QUEUE e foi criada na instalação do sistema.

Por exemplo, o comando DEFINE QLOCAL que segue define uma fila chamada ORANGE.LOCAL.QUEUE com estas características:

- Ele é ativado para gets, ativada para entradas e opera em uma base ordem de prioridade.
- É uma fila *normal*; ela não é uma fila de inicialização ou fila de transmissão e não gera mensagens do acionador.
- A profundidade da fila máxima é 5000 mensagens; o comprimento máximo da mensagem é 4194304 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. Com exceção do valor para a descrição, todos os valores de atributos mostrados são os valores padrão. Nós os mostramos aqui para fins de ilustração. É possível omiti-los se você estiver certo de que os padrões são o que você deseja ou não foram mudados. Consulte também “Exibindo atributos do objeto padrão” na página 86.
2. USAGE (NORMAL) indica que esta fila não é uma fila de transmissão.
3. Se você já tiver uma fila local no mesmo gerenciador de filas com o nome ORANGE.LOCAL.QUEUE, esse comando falhará. Utilize o atributo REPLACE se você deseja sobrescrever a definição existente de uma fila, mas consulte também “Mudando atributos de filas locais” na página 87.

Definindo uma fila de mensagens não entregues

Cada gerenciador de filas deve ter uma fila local para ser utilizada como uma fila de mensagens não entregues de forma que mensagens que não puderem ser entregues ao seu destino correto possam ser armazenadas para recuperação posterior. Deve-se informar o gerenciador de fila sobre a fila de mensagens não entregues

Para informar o gerenciador de filas sobre a fila de mensagens não entregues, especifique um nome da fila de mensagens não entregues no `crtmqm` comando (`crtmqm -u DEAD.LETTER.QUEUE`, por exemplo) ou usando o atributo DEADQ no comando QMALTER para especificar um posterior. Deve-se definir a fila de mensagens não entregues antes de usá-lo.

Uma fila de mensagens não entregues de amostra chamado SYSTEM.DEAD.LETTER.QUEUE está disponível com o produto. Esta fila é criada automaticamente quando você cria o gerenciador de filas. É possível modificar essa definição, se necessário e renomeá-la.

Uma fila de mensagens não entregues não possui requisitos especiais, exceto que:

- Ele deve ser uma fila local
- Seu atributo MAXMSGL (comprimento máximo da mensagem) deve ativar a fila para acomodar as maiores mensagens que o gerenciador de filas possui para manipular **e** o tamanho do cabeçalho de mensagens não entregues (MQDLH)

WebSphere MQ fornece um manipulador de fila de mensagens não entregues que permite especificar como mensagens localizadas em uma fila de mensagens não entregues devem ser processadas ou removidas. Para obter informações adicionais, consulte Manipulação de mensagens não entregues com o manipulador da fila de mensagens não entregues do WebSphere MQ

Exibindo atributos do objeto padrão

É possível usar o comando DISPLAY QUEUE para exibir atributos que foram obtidos do objeto padrão quando um objeto do WebSphere MQ foi definido

Ao definir um objeto WebSphere MQ , ele usa quaisquer atributos que você não especificar a partir do objeto padrão. Por exemplo, ao definir uma fila local, a fila herda quaisquer atributos que você omitir na definição da fila local padrão, que é denominada SYSTEM.DEFAULT.LOCAL.QUEUE. Para ver exatamente quais são esses atributos, utilize o seguinte comando:

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

A sintaxe deste comando é diferente daquele do comando DEFINE correspondente. No comando DISPLAY é possível fornecer somente o nome da fila, enquanto que no comando DEFINE você tem que especificar o tipo da fila ou seja, QLOCAL, QALIAS, QMODEL ou QREMOTE.

É possível exibir seletivamente os atributos especificando-os individualmente. Por exemplo:

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

Este comando exibe os três atributos especificados da seguinte forma:

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

CURDEPTH é a profundidade da fila atual ou seja, o número de mensagens na fila. Este é um atributo útil para exibir, porque, monitorando a profundidade da fila, é possível assegurar que a fila não se torne cheia.

Copiando uma definição de fila local

É possível copiar uma definição de fila utilizando o atributo LIKE no comando DEFINE.

Por exemplo:

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

Esse comando cria uma fila com os mesmos atributos de nossa fila original ORANGE.LOCAL.QUEUE, em vez daqueles da fila local padrão do sistema. Digite o nome da fila a ser copiado ***exatamente*** como foi digitado quando você criou a fila. Se o nome contiver caracteres minúsculos, coloque o nome entre aspas simples.

Também é possível utilizar este formulário do comando DEFINE para copiar uma definição de fila, mas substitua uma ou mais mudanças nos atributos do original. Por exemplo:

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

Este comando copia os atributos da fila ORANGE.LOCAL.QUEUE para a fila THIRD.QUEUE, mas especifica que o comprimento máximo da mensagem na nova fila deve ser 1024 bytes, em vez de 4194304.

Nota:

1. Ao utilizar o atributo LIKE em um comando DEFINE, você está copiando os atributos da fila somente. Você não está copiando as mensagens na fila.
2. Se você definir uma fila local, sem especificar LIKE, é o mesmo que DEFINE LIKE(SYSTEM.DEFAULT.LOCAL.QUEUE).

Mudando atributos de filas locais

É possível mudar os atributos de filas de duas maneiras, usando o comando ALTER QLOCAL ou o comando DEFINE QLOCAL com o atributo REPLACE.

No “Definindo uma fila local” na página 85, a fila chamada ORANGE.LOCAL.QUEUE foi definida. Suponha, por exemplo, que você deseja reduzir o comprimento máximo da mensagem nessa fila para 10.000 bytes.

- Utilizando o comando ALTER:

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

  Esse comando muda um único atributo, aquele do comprimento máximo de mensagem; todos os outros atributos permanecem os mesmos.
- Utilizando o comando DEFINE com a opção REPLACE, por exemplo:

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

  Este comando muda não somente o comprimento máximo da mensagem, mas também todos os outros atributos aos quais são fornecidos seus valores padrão. A fila agora está ativada para put considerando que anteriormente ela estava desativada para put. Ativado para put é o padrão, conforme especificado pela fila SYSTEM.DEFAULT.LOCAL.QUEUE.

  Se você ***diminuir*** o comprimento máximo da mensagem em uma fila existente, as mensagens existentes não serão afetadas. Todas as mensagens novas, no entanto, devem atender aos novos critérios.

Limpando uma Fila Local

É possível utilizar o comando CLEAR para limpar uma fila local.

Para excluir todas as mensagens de uma fila local chamada MAGENTA.QUEUE, utilize o seguinte comando:

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

Nota: Não há nenhum prompt que permita mudar de ideia; uma vez pressionada a tecla Enter as mensagens serão perdidas.

Não é possível limpar uma fila se:

- Há mensagens não confirmadas que foram colocadas na fila sob o ponto de sincronização.
- Um aplicativo possui atualmente a fila aberta.

Excluindo uma fila local

É possível utilizar o comando MQSC DELETE QLOCAL para excluir uma fila local.

Uma fila não pode ser excluída se tiver mensagens não confirmadas nela. No entanto, se a fila tiver uma ou mais mensagens confirmadas e nenhuma mensagem não confirmada, ela poderá ser excluída somente se você especificar a opção PURGE. Por exemplo:

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

Especificando NOPURGE em vez de PURGE assegura que a fila não será excluída se contiver quaisquer mensagens consolidadas.

Procurando filas

O WebSphere MQ fornece um navegador de fila de amostra que pode ser usado para examinar o conteúdo das mensagens em uma fila O navegador é fornecido em ambos os formatos, origem e executável.

O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado.

No WebSphere MQ para Windows, os nomes de arquivos e caminhos para o navegador da fila de amostras são os seguintes:

Fonte
MQ_INSTALLATION_PATH\tools\c\samples\

Executable
MQ_INSTALLATION_PATH\tools\c\samples\bin\amqsbcg.exe

No WebSphere MQ para UNIX and Linux, os nomes de arquivos e caminhos são os seguintes::

Fonte
MQ_INSTALLATION_PATH/samp/amqsbcg0.c

Executable
MQ_INSTALLATION_PATH/samp/bin/amqsbcg

A amostra requer dois parâmetros de entrada, o nome da fila e o nome do gerenciador de filas. Por exemplo:

```
amqsbcg SYSTEM.ADMIN.QMGREVENT.tpp01 saturn.queue.manager
```

Os resultados deste comando são mostrados em Figura 16 na 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. Os resultados do navegador de fila

Ativando Filas Grandes

IBM WebSphere MQ suporta filas com mais de 2 GB.

Em sistemas Windows, suporte para arquivos grandes está disponível sem qualquer ativação adicionais. Nos sistemas AIX, HP-UX, Linuxe Solaris , é preciso ativar explicitamente o suporte de arquivo grande antes de poder criar arquivos de fila maiores que 2 GB... Consulte a documentação do sistema operacional para obter informações sobre como fazer isso.

Alguns utilitários, como o tar, não podem lidar com arquivos com mais de 2 GB. Antes de ativar o suporte a arquivos grandes, verifique a documentação do seu sistema operacional para obter informações sobre as restrições sobre os utilitários que você usa.

Para obter informações sobre como planejar a quantia de armazenamento necessária para filas, visite o website do IBM WebSphere MQ para obter relatórios de desempenho específicos da plataforma:

```
https://www.ibm.com/software/integration/ts/mqseries/
```

Trabalhando com Filas de Alias

É possível definir uma fila de alias para fazer referência indiretamente a outra fila ou tópico.

V 7.5.0.8

Atenção: As listas de distribuição não suportam o uso de filas de alias que apontam para objetos do tópico. A partir da Version 7.5.0, Fix Pack 8, se uma fila de alias apontar para um objeto do tópico em uma lista de distribuição, o IBM WebSphere MQ retornará MQRC_ALIAS_BASE_Q_TYPE_ERROR.

A fila à qual uma fila de alias se refere pode ser qualquer uma das seguintes:

- Uma fila local (consulte “Definindo uma fila local” na página 85).
- Uma definição local de uma fila remota (consulte “Criando uma definição local de uma fila remota” na página 114).
- Um tópico.

Uma fila de alias não é uma fila real, mas uma definição que resolve uma fila real (ou destino) no tempo de execução. A definição da fila de alias especifica a fila de destino. Quando um aplicativo faz uma chamada MQOPEN para uma fila de alias, o gerenciador de filas resolve o alias para o nome de fila de destino.

Uma fila de alias não pode resolver para outro alias da fila definida localmente. No entanto, uma fila de alias podem resolver para as filas de alias que são definidos em outro lugar no clusters do qual o gerenciador de filas locais é um membro. Consulte Resolução de Nome para obter informações adicionais.

As filas de alias são úteis para:

- Fornecer aos diferentes aplicativos diferentes níveis de autoridades de acesso à fila de destino.
- Permitir que os diferentes aplicativos trabalhem com a mesma fila de diferentes maneiras. (Talvez você queira designar diferentes prioridades padrão ou diferentes valores de persistência padrão.)
- Simplificar a manutenção, a migração e o balanceamento de carga de trabalho. (Talvez você queira alterar o nome da fila de destino sem ter que alterar o seu aplicativo, que continua usando o alias.)

Por exemplo, assuma que um aplicativo tenha sido desenvolvido para colocar as mensagens em uma fila denominada MY.ALIAS.QUEUE. Isso especifica o nome desta fila quando ela fizer uma solicitação MQOPEN e, indiretamente, se colocar uma mensagem nesta fila. O aplicativo não reconhece que a fila é uma fila de alias. Para cada chamada MQI que usa este alias, o gerenciador de filas resolve o nome de fila real, que poderia ser uma fila local ou uma fila remota definida neste gerenciador de filas.

Alterando o valor do atributo TARGET, é possível redirecionar chamadas MQI para outra fila, possivelmente em outro gerenciador de filas. Isso é útil para manutenção, migração e balanceamento de carga.

Definindo uma fila de alias

O seguinte comando cria uma fila de alias:

```
DEFINE QALIAS (MY.ALIAS.QUEUE) TARGET (YELLOW.QUEUE)
```

Este comando redireciona as chamadas MQI que especificam o MY.ALIAS.QUEUE para a fila YELLOW.QUEUE. O comando não cria a fila de destino; as chamadas MQI falharão se a fila YELLOW.QUEUE não existir o tempo de execução.

Se você alterar a definição de alias, poderá redirecionar as chamadas MQI para outra fila. Por exemplo:

```
ALTER QALIAS (MY.ALIAS.QUEUE) TARGET (MAGENTA.QUEUE)
```

Este comando redireciona as chamadas MQI para outra fila, MAGENTA.QUEUE.

Também é possível usar as filas de alias para fazer com que uma única fila (fila de destino) pareça ter diferentes atributos para diferentes aplicativos. Isso é feito definindo dois alias, um para cada aplicativo. Imagine que existem dois aplicativos:

- O aplicativo ALPHA pode colocar as mensagens no YELLOW.QUEUE, mas não tem permissão para obter as mensagens dele.
- O aplicativo BETA pode obter as mensagens do YELLOW.QUEUE, mas não tem permissão para colocar as mensagens nele.

O seguinte comando define um alias que é ativado e desativado para o aplicativo ALPHA:

```
DEFINE QALIAS (ALPHAS.ALIAS.QUEUE) +
       TARGET (YELLOW.QUEUE) +
       PUT (ENABLED) +
       GET (DISABLED)
```

O seguinte comando define um alias que é desativado e ativado para o aplicativo BETA:

```
DEFINE QALIAS (BETAS.ALIAS.QUEUE) +
       TARGET (YELLOW.QUEUE) +
       PUT (DISABLED) +
       GET (ENABLED)
```

O ALPHA usa o nome de fila ALPHAS.ALIAS.QUEUE em suas chamadas MQI; BETA usa o nome de fila BETAS.ALIAS.QUEUE. Ambos acessam a mesma fila, mas de diferentes maneiras.

É possível usar os atributos LIKE e REPLACE ao definir os alias de fila, na mesma maneira que você usa esses atributos com as filas locais.

Usando outros comandos com filas de alias

É possível usar os comandos MQSC apropriados para exibir ou alterar os atributos de filas de alias ou excluir o objeto da fila de alias. Por exemplo:

Use o seguinte comando para exibir os atributos da fila de alias:

```
DISPLAY QALIAS (ALPHAS.ALIAS.QUEUE)
```

Use o seguinte comando para alterar o nome da fila base, para o qual o alias é resolvido, em que a opção `force` força a mudança mesmo se a fila estiver aberta:

```
ALTER QALIAS (ALPHAS.ALIAS.QUEUE) TARGET(ORANGE.LOCAL.QUEUE) FORCE
```

Use o seguinte comando para excluir este alias da fila:

```
DELETE QALIAS (ALPHAS.ALIAS.QUEUE)
```

Não é possível excluir uma fila de alias se um aplicativo tiver atualmente a fila aberta. Consulte o Referência MQSC para obter mais informações sobre esse e outros comandos de fila de alias.

Trabalhando com filas modelo

Um gerenciador de filas cria uma *fila dinâmica* se receber uma chamada MQI de um aplicativo especificando um nome de fila que foi definida como uma fila modelo. O nome da fila dinâmica nova é gerada pelo gerenciador de filas quando a fila é criada. Uma *fila modelo* é um modelo que especifica os atributos de quaisquer filas dinâmicas criadas por meio dele. Filas modelo fornece um método conveniente para aplicativos para criar filas conforme necessário.

Definindo uma fila modelo

É possível definir uma fila modelo com um conjunto de atributos da mesma maneira que você definir uma fila local. Filas modelos e filas locais possuem o mesmo conjunto de atributos, exceto que em filas modelos é possível especificar se as filas dinâmicas permanentes ou temporárias são criadas. (As filas permanentes são mantidas nos reinícios do gerenciador de filas, os temporários não são.) Por exemplo:

```
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 comando cria uma definição de fila modelo. A partir do atributo DEFTYPE, é possível ver que as filas reais criadas a partir deste modelo são filas dinâmicas permanentes. Quaisquer atributos não especificados são automaticamente copiados da fila padrão SYSYTEM.DEFAULT.MODEL.QUEUE.

É possível usar os atributos LIKE e REPLACE ao definir filas modelo, da mesma maneira que você os utilize com as filas locais.

Usando outros comandos com filas modelos

É possível usar os comandos MQSC apropriados para exibir ou alterar os atributos de uma fila modelo ou excluir o objeto da fila de modelo. Por exemplo:

Use o seguinte comando para exibir os atributos da fila modelo:

```
DISPLAY QUEUE (GREEN.MODEL.QUEUE)
```

Use o comando a seguir para alterar o modelo para ativar puts em qualquer fila dinâmica criada a partir desse modelo:

```
ALTER QMODEL (BLUE.MODEL.QUEUE) PUT(ENABLED)
```

Use o seguinte comando para excluir esta fila modelo:

```
DELETE QMODEL (RED.MODEL.QUEUE)
```

Trabalhando com tópicos administrativos

Use os comandos MQSC para gerenciar tópicos administrativos.

Consulte Referência MQSC para obter informações detalhadas sobre esses comandos.

Conceitos relacionados

Objetos de Tópico Administrativo

“Definindo um tópico administrativo” na página 93

Use o comando do MQSC **DEFINE TOPIC** para criar um tópico administrativo. Ao definir um tópico administrativo, é possível, opcionalmente, configurar cada atributo do tópico.

"Exibindo atributos do objeto do tópico administrativo" na página 93
Use o comando do MQSC **DISPLAY TOPIC** para exibir um objeto do tópico administrativo.

"Mudando atributos de tópico administrativo" na página 94
É possível alterar os atributos do tópico de duas maneiras, usando o comando **ALTER TOPIC** ou o comando **DEFINE TOPIC** com o atributo **REPLACE**

"Copiando uma definição de tópico administrativo" na página 94
É possível copiar uma definição de tópico utilizando o atributo LIKE no comando **DEFINE**.

"Excluindo uma definição de tópico administrativo" na página 95
É possível utilizar o comando MQSC **DELETE TOPIC** para excluir um tópico administrativo.

Definindo um tópico administrativo

Use o comando do MQSC **DEFINE TOPIC** para criar um tópico administrativo. Ao definir um tópico administrativo, é possível, opcionalmente, configurar cada atributo do tópico.

Qualquer atributo do tópico que não é explicitamente configurado é herdado do tópico administrativo padrão, SYSTEM.DEFAULT.TOPIC, que foi criado quando a instalação do sistema foi instalado.

Por exemplo, o comando **DEFINE TOPIC** que segue, define um tópico chamado **ORANGE.TOPIC** com estas características:

- Resolve para o ORANGE sequência de tópicos. Para obter informações sobre como as sequências de tópicos podem ser usadas, consulte Combinando sequências de tópicos.
- Qualquer atributo que é configurado para ASPARENT utiliza o atributo conforme definido pelo tópico pai deste tópico. Essa ação é repetida a árvore de tópicos tanto quanto o tópico raiz, SYSTEM.BASE.TOPIC está localizado. Para obter mais informações sobre as árvores de tópicos, consulte Árvores de tópicos

```
DEFINE TOPIC (ORANGE.TOPIC) +
       TOPICSTR (ORANGE) +
       DEFPRTY(ASPARENT) +
       NPMSGDLV(ASPARENT)
```

Nota:

- Exceto para o valor da sequência de tópico, todos os valores de atributos mostrados são os valores padrão. Eles são mostrados aqui somente como uma ilustração.É possível omiti-los se você estiver certo de que os padrões são o que você deseja ou não foram mudados. Consulte também "Exibindo atributos do objeto do tópico administrativo" na página 93.
- Se você já tiver um tópico administrativo no mesmo gerenciador de filas com o nome ORANGE.TOPIC, este comando falhará.Use o atributo REPLACE se você deseja sobrescrever a definição existente de um tópico, mas consulte também "Mudando atributos de tópico administrativo" na página 94

Exibindo atributos do objeto do tópico administrativo

Use o comando do MQSC **DISPLAY TOPIC** para exibir um objeto do tópico administrativo.

Para exibir todos os tópicos, utilize:

```
DISPLAY TOPIC(ORANGE.TOPIC)
```

É possível exibir seletivamente os atributos especificando-os individualmente.Por exemplo:

```
DISPLAY TOPIC(ORANGE.TOPIC) +
        TOPICSTR +
        DEFPRTY +
        NPMSGDLV
```

Este comando exibe os três atributos especificados da seguinte forma:

```
AMQ8633: Display topic details.
   TOPIC(ORANGE.TOPIC)                     TYPE(LOCAL)
   TOPICSTR(ORANGE)                        DEFPRTY(ASPARENT)
   NPMSGDLV(ASPARENT)
```

Para exibir os valores do tópico ASPARENT como eles são utilizados no Tempo de execução, use DISPLAY TPSTATUS. Por exemplo, use:

```
DISPLAY TPSTATUS(ORANGE) DEFPRTY NPMSGDLV
```

O comando exibe os seguintes detalhes:

```
AMQ8754: Display topic status details.
   TOPICSTR(ORANGE)                        DEFPRTY(0)
   NPMSGDLV(ALLAVAIL)
```

Ao definir um tópico administrativo, ele assume quaisquer atributos que você não especificar explicitamente do tópico administrativo padrão, que é chamado SYSTEM.DEFAULT.TOPIC. Para ver quais esses atributos padrão são, utilize o seguinte comando:

```
DISPLAY TOPIC (SYSTEM.DEFAULT.TOPIC)
```

Mudando atributos de tópico administrativo

É possível alterar os atributos do tópico de duas maneiras, usando o comando **ALTER TOPIC** ou o comando **DEFINE TOPIC** com o atributo **REPLACE**

Se, por exemplo, você deseja mudar a prioridade padrão de mensagens entregues a um tópico chamado ORANGE.TOPIC, para ser 5, utilize um dos seguintes comandos.

- Usando o comando **ALTER**:

  ```
  ALTER TOPIC(ORANGE.TOPIC) DEFPRTY(5)
  ```

  Esse comando muda um único atributo, que a prioridade padrão de mensagens entregues a este tópico para 5; todos os outros atributos permanecem os mesmos.
- Usando o comando **DEFINE**:

  ```
  DEFINE TOPIC(ORANGE.TOPIC) DEFPRTY(5) REPLACE
  ```

  Este comando muda a prioridade padrão das mensagens entregues para este tópico. Todos os outros atributos recebem valores padrão.

Se você alterar a prioridade de mensagens enviadas para este tópico, as mensagens existentes não serão afetadas. Qualquer nova mensagem, no entanto, utilizar a prioridade especificada se não fornecido pelo aplicativo de publicação.

Copiando uma definição de tópico administrativo

É possível copiar uma definição de tópico utilizando o atributo LIKE no comando **DEFINE**.

Por exemplo:

```
DEFINE TOPIC (MAGENTA.TOPIC) +
       LIKE (ORANGE.TOPIC)
```

Este comando cria um tópico, MAGENTA.TOPIC, com os mesmos atributos que o tópico original, ORANGE.TOPIC, em vez de aqueles do tópico administrativo padrão do sistema. Digite o nome do tópico a ser copiado exatamente como foi digitado quando você criou o tópico. Se o nome contiver caracteres minúsculos, coloque o nome entre aspas simples.

Também é possível utilizar este formulário do comando **DEFINE** para copiar uma definição de tópico, mas faça mudanças aos atributos do original. Por exemplo:

```
DEFINE TOPIC(BLUE.TOPIC) +
       TOPICSTR(BLUE) +
       LIKE(ORANGE.TOPIC)
```

Também é possível copiar os atributos do tópico BLUE.TOPIC para o tópico GREEN.TOPIC e especificar que quando as publicações não podem ser entregues para sua fila de assinantes correta que não são colocados na fila de mensagens não entregues. Por exemplo:

```
DEFINE TOPIC(GREEN.TOPIC) +
       TOPICSTR(GREEN) +
       LIKE(BLUE.TOPIC) +
       USEDLQ(NO)
```

Excluindo uma definição de tópico administrativo

É possível utilizar o comando MQSC **DELETE TOPIC** para excluir um tópico administrativo.

```
DELETE TOPIC(ORANGE.TOPIC)
```

Os aplicativos não estarão mais aptos a abrir o tópico para publicação ou criar novas assinaturas usando o nome do objeto, ORANGE.TOPIC.Aplicativos de publicação que possuem a abertura do tópico estão aptos a continuar publicando a sequência de tópicos resolvidos.Todas as assinaturas já feitas neste tópico continuam recebendo publicações após a exclusão do tópico.

Os aplicativos que não estão referenciando esse objeto do tópico, mas estão utilizando a sequência de tópico resolvida que este objeto do tópico representado, 'ORANGE' neste exemplo, continuam a funcionar. Neste caso eles herdam as propriedades de um objeto do tópico mais alto na árvore de tópicos. Para obter mais informações sobre as árvores de tópicos, consulte Árvores de tópicos

Trabalhando com assinaturas

Use os comandos MQSC para gerenciar assinaturas.

As assinaturas podem ser um dos três tipos, definido no atributo **SUBTYPE**:

ADMIN
Administrativamente definido por um usuário.

PROXY
Uma assinatura criada internamente para roteamento de publicações entre gerenciadores de filas.

API
Criado programaticamente, por exemplo, usando a chamada MQI MQSUB.

Consulte o Referência MQSC para obter informações detalhadas sobre esses comandos.

Conceitos relacionados

“Definindo uma assinatura administrativa” na página 96
Use o comando do MQSC **DEFINE SUB** para criar uma assinatura administrativa. Também é possível utilizar o padrão definido na definição de assinatura local padrão. Ou, é possível modificar as características de assinatura a partir desses da assinatura local padrão, SYSTEM.DEFAULT.SUB que foi criado quando o sistema foi instalado.

“Exibindo atributos de assinaturas” na página 96
É possível utilizar o comando **DISPLAY SUB** para exibir os atributos configurados de qualquer assinatura conhecidos para o gerenciador de filas.

“Mudando atributos de assinatura local” na página 97
É possível alterar os atributos de assinatura de duas maneiras, usando o comando **ALTER SUB** ou o comando **DEFINE SUB** com o atributo **REPLACE**

“Copiando uma definição de assinatura local” na página 98

É possível copiar uma definição de assinatura usando o atributo **LIKE** no comando **DEFINE**.

“Excluindo uma Assinatura” na página 98
É possível utilizar o comando MQSC **DELETE SUB** para excluir uma assinatura local.

Definindo uma assinatura administrativa

Use o comando do MQSC **DEFINE SUB** para criar uma assinatura administrativa. Também é possível utilizar o padrão definido na definição de assinatura local padrão. Ou, é possível modificar as características de assinatura a partir desses da assinatura local padrão, SYSTEM.DEFAULT.SUB que foi criado quando o sistema foi instalado.

Por exemplo, o comando **DEFINE SUB** que a seguir define uma assinatura chamado ORANGE com estas características:

- Assinatura durável, o que significa que ele persistir durante a reinicialização do gerenciador de filas, com expiração ilimitada.
- Receber as publicações feitas na sequência de tópico ORANGE, com as prioridades de mensagens conforme definido pelos aplicativos de publicação.
- Publicações entregues para essa assinatura são enviadas para a fila local SUBQ, essa fila deve ser definida antes da definição da assinatura.

```
DEFINE SUB (ORANGE) +
       TOPICSTR (ORANGE) +
       DESTCLAS (PROVIDED) +
       DEST (SUBQ) +
       EXPIRY (UNLIMITED) +
       PUBPRTY (ASPUB)
```

Nota:

- O nome de sequência de assinatura e tópico não precisam corresponder.
- Exceto para os valores da descrição e da sequência de tópicos, todos os valores de atributo mostrados são os valores padrãoEles são mostrados aqui somente como uma ilustração.É possível omiti-los se você estiver certo de que os padrões são o que você deseja ou não foram mudados. Consulte também “Exibindo atributos de assinaturas” na página 96.
- Se você já tiver uma assinatura local no mesmo gerenciador de filas com o nome TEST, esse comando falhará..Utilize o atributo **REPLACE** se você deseja sobrescrever a definição existente de uma fila, mas consulte também “Mudando atributos de assinatura local” na página 97.
- Se o SUBQ de filas não existir, este comando falhará.

Exibindo atributos de assinaturas

É possível utilizar o comando **DISPLAY SUB** para exibir os atributos configurados de qualquer assinatura conhecidos para o gerenciador de filas.

Por exemplo, use:

```
DISPLAY SUB (ORANGE)
```

É possível exibir seletivamente os atributos especificando-os individualmente. Por exemplo:

```
DISPLAY SUB (ORANGE) +
        SUBID +
        TOPICSTR +
        DURABLE
```

Este comando exibe os três atributos especificados da seguinte forma:

```
AMQ8096: WebSphere MQ subscription inquired.
   SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   SUB(ORANGE)                                TOPICSTR(ORANGE)
   DURABLE(YES)
```

TOPICSTR é a sequência de tópicos resolvidos no qual esse assinante está operando.Quando uma assinatura é definida para usar um objeto do tópico, a sequência de tópicos a partir desse objeto é usada como um prefixo para a sequência de tópicos fornecida ao fazer a assinatura. SUBID é um identificador exclusivo designado pelo gerenciador de filas quando uma assinatura for criada.Este é um atributo útil para exibição porque alguns nomes de assinatura podem ser longos ou estar em um conjuntos de caracteres diferentes para os quais pode se tornar inviável.

Um método alternativo para exibir assinaturas é utilizar o SUBID:

```
DISPLAY SUB +
        SUBID(414D5120414141202020202020202020EE921E4E20002A03) +
        TOPICSTR +
        DURABLE
```

Esse comando fornece a mesma saída de antes:

```
AMQ8096: WebSphere MQ subscription inquired.
 SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   SUB(ORANGE)                                TOPICSTR(ORANGE)
   DURABLE(YES)
```

As assinaturas de proxy em um gerenciador de filas não são exibidas por padrão. Para exibi-los especificar um **SUBTYPE** de PROXY ou ALL.

É possível utilizar o comando DISPLAY SBSTATUS para exibir os atributos de Execução. Por exemplo, utilize o comando:

```
DISPLAY SBSTATUS(ORANGE) NUMMSGS
```

A seguinte saída é exibida:

```
AMQ8099: WebSphere MQ subscription status inquired.
   SUB(ORANGE)
   SUBID(414D5120414141202020202020202020EE921E4E20002A03)
   NUMMSGS(0)
```

Quando você definir uma assinatura administrativa, ele toma quaisquer atributos que você não especificar explicitamente a partir da assinatura padrão, que é chamado SYSTEM.DEFAULT.SUB. Para ver quais esses atributos padrão são, utilize o seguinte comando:

```
DISPLAY SUB (SYSTEM.DEFAULT.SUB)
```

Mudando atributos de assinatura local

É possível alterar os atributos de assinatura de duas maneiras, usando o comando **ALTER SUB** ou o comando **DEFINE SUB** com o atributo **REPLACE**

Se, por exemplo, você deseja mudar a prioridade de mensagens entregues a uma assinatura chamado ORANGE para ser 5, utilize um dos seguintes comandos:

- Utilizando o comando ALTER:

  ```
  ALTER SUB(ORANGE) PUBPRTY(5)
  ```

  Este comando muda um único atributo, aquele da prioridade de mensagens entregues para esta assinatura para 5; todos os outros atributos permanecem os mesmos.

- Utilizando o comando DEFINE:

  ```
  DEFINE SUB (ORANGE) PUBPRTY(5) REPLACE
  ```

  Esse comando muda não só a prioridade de mensagens entregues para esta assinatura, mas todos os outros atributos que recebem valores padrão.

Se você alterar a prioridade de mensagens enviadas para esta assinatura, as mensagens existentes não são afetadas.Todas as mensagens novas, no entanto, têm a prioridade especificada.

Copiando uma definição de assinatura local

É possível copiar uma definição de assinatura usando o atributo **LIKE** no comando **DEFINE**.

Por exemplo:

```
DEFINE SUB (BLUE) +
       LIKE (ORANGE)
```

Também é possível copiar os atributos do sub REAL para o sub THIRD.SUB e especificar que o correlID de publicações entregues é THIRD, em vez do correlID dos publicadores. Por exemplo:

```
DEFINE SUB(THIRD.SUB) +
       LIKE(BLUE) +
       DESTCORL(ORANGE)
```

Excluindo uma Assinatura

É possível utilizar o comando MQSC **DELETE SUB** para excluir uma assinatura local.

```
DELETE SUB(ORANGE)
```

Também é possível excluir uma assinatura utilizando o SUBID:

```
DELETE SUB SUBID(414D5120414141202020202020202020EE921E4E20002A03)
```

Verificando mensagens em uma assinatura

Sobre esta tarefa

Quando uma assinatura é definida, ela é associada a uma fila. As mensagens correspondentes a esta assinatura serão colocadas nesta fila.

Observe que os seguintes comandos **runmqsc** mostram somente as assinaturas que recebem mensagens.

Para verificar se há mensagens atualmente enfileiradas para uma assinatura execute as seguintes etapas:

Procedimento

1. Para verificar as mensagens enfileiradas de um tipo de assinatura **DISPLAY SBSTATUS(<sub_name>) NUMMSGS**, consulte “Exibindo atributos de assinaturas” na página 96.
2. Se o valor **NUMMSGS** for maior que zero, identifique a fila associada à assinatura digitando **DISPLAY SUB(<sub_name>)DEST**.
3. Utilizando o nome de fila retornados, é possível visualizar as mensagens, a técnica descrita no “Procurando filas” na página 88.

Trabalhando com Serviços

Os objetos de serviço são um meio pelo qual processos adicionais podem ser gerenciados como parte de um gerenciador de filas. Com os serviços, é possível definir programas que são iniciados e interrompidos quando o gerenciador de filas inicia e termina. Os serviços do IBM WebSphere MQ são sempre iniciados sob o ID do usuário que iniciou o gerenciador de filas.

Os objetos de serviços podem ser um dos tipos a seguir:

Servidor
Um servidor é um objeto de serviço que possui o parâmetro SERVTYPE especificado como SERVER. Um objeto de serviço do servidor é a definição de um programa que é executado quando um gerenciador de filas especificado é iniciado. Os objetos de serviço do servidor definem programas

que geralmente são executados por um longo tempo. Por exemplo, um objeto de serviço do servidor pode ser usado para executar um processo do monitor acionador, como `runmqtrm`..

Somente uma instância de um objeto de serviço do servidor pode ser executada simultaneamente. O status de objetos de serviço do servidor em execução pode ser monitorado usando o comando do MQSC, DISPLAY SVSTATUS.

Comando:
Um comando é um objeto de serviço que possui o parâmetro SERVTYPE especificado como COMMAND. Os objetos de serviço de comando são semelhantes aos objetos de serviço do servidor, no entanto, diversas instâncias de um objeto de serviço de comando podem ser executadas simultaneamente e seus status não podem ser monitorados usando o comando do MQSC DISPLAY SVSTATUS.

Se o comando do MQSC, STOP SERVICE, for executado, nenhuma verificação será feita para determinar se o programa foi iniciado pelo comando do MQSC, START SERVICE, ainda está ativo antes de executar o programa de parada.

Definindo um objeto de serviço

Você define um objeto de serviço com vários atributos.

Os atributos são os seguintes:

SERVTYPE
Define o tipo do objeto de serviço. Os valores possíveis são os seguintes:

server
Um objeto de serviço do servidor.

Somente uma instância de um objeto de serviço do servidor pode ser executada por vez. O status de objetos de serviço do servidor pode ser monitorado usando o comando do MQSC, DISPLAY SVSTATUS.

COMANDO
Um objeto de serviço de comando.

Várias instâncias de um objeto de serviço de comando podem ser executadas simultaneamente. O status de um objeto de serviço de comando não pode ser monitorado.

STARTCMD
O programa que é executado para iniciar o serviço. Um caminho completo para o programa deve ser especificado.

STARTARG
Argumentos transmitidos para o programa de início.

STDERR
Especifica o caminho para um arquivo para o qual o erro padrão (stderr) do programa de serviço deve ser redirecionado.

STDOUT
Especifica o caminho para um arquivo para o qual a saída padrão (stdout) do programa de serviço deve ser redirecionado.

STOPCMD
O programa que é executado para parar o serviço. Um caminho completo para o programa deve ser especificado.

STOPARG
Argumentos passados para o programa de parada.

CONTROLE
Especifica como o serviço deve ser iniciado e parado:

MANUAL
O serviço não deve ser iniciado automaticamente ou parado automaticamente. Ele é controlado pelo uso dos comandos START SERVICE e STOP SERVICE. Esse é o valor-padrão.

QMGR
O serviço que está sendo definido deve ser iniciado e parado ao mesmo tempo que o gerenciador de filas é iniciado e parado.

STARTONLY
O serviço deve ser iniciado ao mesmo tempo que o gerenciador de filas é iniciado, mas não é solicitado a parar quando o gerenciador de filas é parado.

Conceitos relacionados

“Serviço gerenciado” na página 100
Utilizando o parâmetro CONTROL, uma instância de um objeto de serviço pode ser iniciado e parado automaticamente pelo gerenciador de filas ou iniciado e parado utilizando os comandos MQSC START SERVICE e STOP SERVICE.

Serviço gerenciado

Utilizando o parâmetro CONTROL, uma instância de um objeto de serviço pode ser iniciado e parado automaticamente pelo gerenciador de filas ou iniciado e parado utilizando os comandos MQSC START SERVICE e STOP SERVICE.

Quando uma instância de um objeto de serviço é iniciado, uma mensagem será gravada no log de erros do gerenciador de filas que contém o nome do objeto de serviço e o ID de processo do processo iniciado. Uma entrada de log de exemplo para um objeto de serviço do servidor inicial da seguinte forma:

```
  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.
```

Uma entrada de log de exemplo para um início de objeto de serviço de comando da seguinte forma:

```
  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.
```

Quando um serviço de instância do servidor para, uma mensagem será gravada nos logs de erros do gerenciador de filas que contém o nome do serviço e o ID do processo final. Uma entrada de log de exemplo para um objeto de serviço do servidor de parada a seguir:

```
  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.
```

Referências relacionadas

“Variáveis de ambiente adicionais” na página 101
Quando um serviço é iniciado, o ambiente no qual o processo de serviço é iniciado é herdado do ambiente do gerenciador de filas. É possível definir variáveis de ambiente adicionais a serem configuradas no

ambiente do processo de serviço incluindo as variáveis que você deseja definir em um dos arquivos de substituição de ambiente `service.env`.

Variáveis de ambiente adicionais

Quando um serviço é iniciado, o ambiente no qual o processo de serviço é iniciado é herdado do ambiente do gerenciador de filas. É possível definir variáveis de ambiente adicionais a serem configuradas no ambiente do processo de serviço incluindo as variáveis que você deseja definir em um dos arquivos de substituição de ambiente `service.env`.

Nota:

Há dois arquivos possíveis nos quais você pode incluir variáveis de ambiente:

- O arquivo `service.env` do escopo de máquina, que está localizado em `/var/mqm` em sistemas UNIX and Linux ou no diretório de dados selecionado durante a instalação em sistemas Windows.
- O gerenciador de filas escopo `service.env` arquivo, que está localizado no diretório de dados do gerenciador de filas. Por exemplo, o local do arquivo de substituição do ambiente para um gerenciador de filas denominado QMNAME é:
  - Em sistemas UNIX and Linux, `/var/mqm/qmgrs/QMNAME/service.env`
  - Em sistemas Windows, `C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\service.env`

Ambos os arquivos são processados, se disponível, com definições no arquivo do escopo do gerenciador de filas que têm precedência sobre as definições no arquivo do escopo de máquina.

Qualquer variável de ambiente pode ser especificada em `service.env`. Por exemplo, se o serviço IBM WebSphere MQ executa vários comandos, pode ser útil definir a variável de usuário PATH no arquivo `service.env`. Os valores para os quais você configura a variável não podem ser variáveis de ambiente; por exemplo, CLASSPATH=*%CLASSPATH%* está incorreto. Da mesma forma, em Linux PATH=*$PATH*`:/opt/mqm/bin` forneceria resultados inesperados

CLASSPATH deve ser capitalizado e a instrução do caminho da classe pode conter somente literais. Alguns serviços (de telemetria, por exemplo) configuram seu próprio caminho da classe. O CLASSPATH definido em `service.env` é incluído nele.

O formato das variáveis definidas no arquivo,`service.env` é uma lista de pares de variáveis de nome e valor. Cada variável deve ser definida em uma nova linha e cada variável é obtida como é explicitamente definida, incluindo espaço em branco. Um exemplo do arquivo,`service.env` é o seguinte:

```
#********************************************************************#
#*                                                                  *#
#* <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
```

Referências relacionadas

“Inserções substituíveis em definições de serviço” na página 102

Na definição de um objeto de serviço, é possível substituir os tokens. Tokens que são substituídos são automaticamente substituídos por seu texto expandido quando o programa de serviço é executado. Tokens substitutos podem ser obtidos da seguinte lista de tokens comuns ou a partir de quaisquer variáveis que são definidas no arquivo, service.env.

Inserções substituíveis em definições de serviço

Na definição de um objeto de serviço, é possível substituir os tokens. Tokens que são substituídos são automaticamente substituídos por seu texto expandido quando o programa de serviço é executado. Tokens substitutos podem ser obtidos da seguinte lista de tokens comuns ou a partir de quaisquer variáveis que são definidas no arquivo, service.env.

Os seguintes são tokens comuns que podem ser utilizados para substituir tokens na definição de um objeto de serviço:

MQ_INSTALL_PATH
O local em que o WebSphere MQ está instalado

MQ_DATA_PATH
O local do diretório de dados do WebSphere MQ :

- Em sistemas UNIX and Linux , o local do diretório de dados WebSphere MQ é `/var/mqm/`
- Em sistemas Windows , o local do diretório de dados do WebSphere MQ é o diretório de dados selecionado durante a instalação do WebSphere MQ

QMNAME
O nome do gerenciador de filas atual.

MQ_SERVICE_NAME
O nome do serviço.

MQ_SERVER_PID
Esse token pode ser utilizado somente pelos argumentos STOPARG e STOPCMD.

Para objetos de serviço do servidor esse token é substituído pelo id de processo do processo iniciado pelos argumentos STARTCMD e STARTARG. Caso contrário, esse token é substituído por 0.

MQ_Q_MGR_DATA_PATH
O local do diretório de dados do gerenciador de filas.

MQ_Q_MGR_DATA_NAME
O nome transformado do gerenciador de filas. Para obter mais informações sobre transformação de nome, consulte Entendendo WebSphere MQ nomes de arquivos.

Para usar inserções substituíveis, insira o token dentro dos caracteres + em qualquer uma das sequências STARTCMD, STARTARG, STOPCMD, STOPARG, STDOUT ou STDERR. Para obter exemplos disso, consulte “Exemplos sobre como utilizar objetos de serviço” na página 102.

Exemplos sobre como utilizar objetos de serviço

Os serviços nesta seção são gravados com caracteres separadores de caminho de estilo UNIX , exceto onde indicado de outra forma...

Usando um objeto de serviço do servidor

Este exemplo mostra como definir, utilizar e alterar, um objeto de serviço do servidor para iniciar um monitor de acionador.

1. Um objeto de serviço do servidor é definido, utilizando o seguinte comando 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+')
```

Em que:

+MQ_INSTALL_PATH+ é um token que representa o diretório de instalação.

+QMNAME+ é um token que representa o nome do gerenciador de filas.

ACCOUNTS.INITIATION.QUEUE é a fila de inicialização.

amqsstop é um programa de amostra fornecido com o WebSphere MQ que solicita que o gerenciador de filas interrompa todas as conexões para o ID do processo amqsstop gera comandos PCF, portanto, o servidor de comandos deve estar em execução.

+MQ_SERVER_PID+ é um token que representa o ID do processo transmitido para o programa de parada

Consulte "Inserções substituíveis em definições de serviço" na página 102 para uma lista dos tokens comuns.

2. Uma instância do objeto de serviço do servidor será executada quando o gerenciador de filas for iniciado da próxima vez. No entanto, vamos iniciar uma instância do objeto de serviço do servidor imediatamente com o comando MQSC a seguir:

```
START SERVICE(S1)
```

3. O status do processo do serviço do servidor é exibido, usando o comando MQSC a seguir:

```
DISPLAY SVSTATUS(S1)
```

4. Este exemplo agora mostra como alterar o objeto de serviço do servidor e ter as atualizações selecionadas reiniciando manualmente o processo de serviço do servidor. O objeto de serviço do servidor é alterado para que a fila de iniciação seja especificada como JUPITER.INITIATION.QUEUE O comando MQSC a seguir é utilizado:

```
ALTER SERVICE(S1) +
      STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

Nota: Um serviço em execução não selecionará todas as atualizações para sua definição de serviço até que seja reiniciado.

5. O processo de serviço do servidor é reiniciado para que a alteração seja selecionada, utilizando os seguintes comandos do MQSC:

```
STOP SERVICE(S1)
```

Seguido por:

```
START SERVICE(S1)
```

O processo de serviço do servidor é reiniciado e selecionará as alterações feitas no "4" na página 103.

Nota: O comando do MQSC, STOP SERVICE somente pode ser utilizado se um argumento STOPCMD está especificado na definição de serviço.

Usando um objeto de serviço de comando

Este exemplo mostra como definir um objeto de serviço de comando para iniciar um programa que grava entradas no log do sistema do sistema operacional quando um gerenciador de filas é iniciado ou parado.

1. O objeto de serviço do comando é definido, utilizando o comando MQSC a seguir:

```
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')
```

Em que:

`logger` é o sistema UNIX and Linux fornecido pelo comando para gravar no log do sistema.
+QMNAME+ é um token que representa o nome do gerenciador de filas.

Usando um objeto de serviço de comando quando um gerenciador de filas termina somente

Este exemplo mostra como definir um objeto de serviço de comando para iniciar um programa que grava entradas no log do sistema do sistema operacional quando um gerenciador de filas está somente parado.

1. O objeto de serviço do comando é definido, utilizando o comando MQSC a seguir:

```
DEFINE SERVICE(S3) +
       CONTROL(QMGR) +
       SERVTYPE(COMMAND) +
       STOPCMD('/usr/bin/logger') +
       STOPARG('Queue manager +QMNAME+ stopping')
```

Em que:

`logger` é um programa de amostra fornecido com o WebSphere MQ que pode gravar entradas no log do sistema operacional.
+QMNAME+ é um token que representa o nome do gerenciador de filas.

Mais sobre transmissão de argumentos

Este exemplo mostra como definir um objeto de serviço do servidor para iniciar um programa chamado `runserv` quando um gerenciador de filas é iniciado.

Este exemplo é gravado com caracteres separadores de caminho de estilo do Windows

Um dos argumentos que serão transmitidos ao programa inicial é uma sequência que contém um espaço. Esse argumento precisa ser transmitido como uma sequência única. Para conseguir isso, aspas duplas são utilizadas conforme mostrado no comando a seguir para definir o objeto de serviço de comando:

1. O objeto de serviço do servidor é definido, utilizando o seguinte comando MQSC:

```
   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')
```

Em que:

+QMNAME+ é um token que representa o nome do gerenciador de filas.
`"C:\Program Files\Tools\"` é uma sequência que contém um espaço, que será transmitido como uma única sequência de caracteres

Inicialização automática de um serviço

Este exemplo mostra como definir um objeto de serviço do servidor que pode ser utilizado para iniciar automaticamente o monitor acionador quando o gerenciador de filas é iniciado.

1. O objeto de serviço do servidor é definido, utilizando o seguinte comando MQSC:

```
DEFINE SERVICE(TRIG_MON_START) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('runmqtrm') +
       STARTARG('-m +QMNAME+ -q +IQNAME+')
```

Em que:

+QMNAME+ é um token que representa o nome do gerenciador de filas.

+IQNAME+ é uma variável de ambiente definida pelo usuário em um dos arquivos service.env que representa o nome da fila de inicialização.

Gerenciando os Objetos para Acionamento

WebSphere MQ permite iniciar um aplicativo automaticamente quando determinadas condições em uma fila são atendidas. Por exemplo, talvez você queira iniciar um aplicativo quando o número de mensagens em uma fila atingir um número especificado. Este recurso é denominado *acionamento*. Você precisa definir os objetos que suportam o acionamento.

Acionamento descrito em detalhes em Iniciando WebSphere MQ aplicativos usando acionadores

Definindo uma Fila do Aplicativo para o Acionamento

Uma fila do aplicativo é uma fila local que é usada pelos aplicativos para sistema de mensagens, por meio do MQI. O acionamento requer que inúmeros atributos de fila sejam definidos na fila do aplicativo.

O próprio acionamento é ativado pelo atributo *Trigger* (TRIGGER em comandos MQSC). Neste exemplo, um evento acionador deve ser gerado quando houver 100 mensagens de prioridade 5 ou mais na fila local MOTOR.INSURANCE.QUEUE, da seguinte maneira:

```
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)
```

em que:

QLOCAL (MOTOR.INSURANCE.QUEUE)
É o nome da fila do aplicativo sendo definida.

PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
É o nome da definição de processo que define o aplicativo a ser iniciado por um programa do monitor acionador.

MAXMSGL (2000)
É o comprimento máximo das mensagens na fila.

DEFPSIST (YES)
Especifica que as mensagens nesta fila são persistentes por padrão.

INITQ (MOTOR.INS.INIT.QUEUE)
É o nome da fila de inicialização na qual o gerenciador de filas deve colocar a mensagem do acionador.

TRIGGER
É o valor de atributo do acionado.

TRIGTYPE (DEPTH)
Especifica que um evento acionador é gerado quando o número de mensagens da propriedade necessária (TRIGMPRI) atinge o número especificado em TRIGDPTH.

TRIGDPTH (100)
É o número de mensagens necessárias para gerar um evento acionador.

TRIGMPRI (5)
É a prioridade de mensagens que devem ser contadas pelo gerenciador de filas ao decidir se deve gerar um evento acionador. Apenas as mensagens com prioridade 5 ou mais são contadas.

Definindo uma Fila de Inicialização

Quando ocorrer um evento do acionador, o gerenciador de filas colocará uma mensagem do acionador na fila de inicialização especificada na definição da fila do aplicativo. As filas de inicialização não possuem configurações especiais, mas é possível usar a seguinte definição na fila local MOTOR.INS.INIT.QUEUE para orientação:

```
DEFINE QLOCAL(MOTOR.INS.INIT.QUEUE) +
       GET (ENABLED) +
       NOSHARE +
       NOTRIGGER +
       MAXMSGL (2000) +
       MAXDEPTH (1000)
```

Definindo um Processo

Use o comando DEFINE PROCESS para criar uma definição de processo. Uma definição de processo define o aplicativo a ser usado para processar as mensagens da fila do aplicativo. A definição da fila do aplicativo nomeia o processo a ser usado e portanto associa a fila de aplicativos ao aplicativo a ser usado para processar suas mensagens. Isso é feito por meio do atributo PROCESS na fila do aplicativo MOTOR.INSURANCE.QUEUE. O seguinte comando MQSC define o processo necessário, MOTOR.INSURANCE.QUOTE.PROCESS, identificado neste exemplo:

```
DEFINE PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
               DESCR ('Insurance request message processing') +
               APPLTYPE (UNIX) +
               APPLICID ('/u/admin/test/IRMP01') +
               USERDATA ('open, close, 235')
```

Em que:

MOTOR.INSURANCE.QUOTE.PROCESS
É o nome da definição de processo.

DESCR ('Insurance request message processing')
Descreve o programa de aplicativo ao qual desta definição se relaciona. Este texto é exibido ao usar o comando DISPLAY PROCESS. Isso pode ajudá-lo a identificar o qual o processo faz. Se você usar os espaços na sequência, deve colocar a sequência entre aspas simples.

APPLTYPE (UNIX)
É o tipo de aplicativo a ser iniciado.

APPLICID ('/u/admin/test/IRMP01')
É o nome do arquivo executável do aplicativo, especificado como um nome completo do arquivo. Em sistemas Windows , um valor típico de APPLICID seria `c:\appl\test\irmp01.exe`.

USERDATA ('open, close, 235')
São dados definidos pelo usuário, que podem ser usados pelo aplicativo.

Exibindo Atributos de uma Definição de Processo

Use o comando DISPLAY PROCESS para examinar os resultados de sua definição. Por exemplo:

```
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)
```

Também é possível usar o comando ALTER PROCESS do MQSC para alterar uma definição de processo existente e o comando DELETE PROCESS para excluir uma definição de processo.

Administrando objetos remotos do IBM WebSphere MQ

Esta seção informa como administrar objetos do IBM WebSphere MQ em um gerenciador de filas remotas usando comandos MQSC e como utilizar objetos de fila remota para controlar o destino de mensagens e mensagens de resposta.

Esta seção descreve:

Canais, clusters e enfileiramento remoto

Um gerenciador de filas se comunica com outro gerenciador de filas enviando uma mensagem e, se necessário, recebendo de volta uma resposta. O gerenciador de fila de recebimento poderia ser:

- Na mesma máquina
- Em outra máquina o mesmo local (ou mesmo no outro lado do mundo)
- Executando na mesma plataforma que o gerenciador de filas locais
- Executando em outra plataforma suportada pelo WebSphere MQ

Essas mensagens podem se originar de:

- Programas de aplicativo gravados pelo usuário que transferem dados de um nó para outro
- Aplicativos de administração gravados pelo usuário que usam comandos PCF ou a MQAI
- O IBM WebSphere MQ Explorer.
- Gerenciadores de filas enviando:
  - Mensagens de eventos de instrumentação para outro gerenciador de filas
  - Comandos MQSC emitidos a partir de um comando `runmqsc` no modo indireto (no qual os comandos são executados em outro gerenciador de filas)

Antes de uma mensagem poder ser enviada para um gerenciador de filas remotas, o gerenciador de filas locais precisa de um mecanismo para detectar a chegada de mensagens e transportá-las consistindo em:

- Pelo menos um canal
- Uma fila de transmissão
- Um iniciador de canal

Para um gerenciador de filas remotas receber uma mensagem, um listener é necessário.

Um canal é um link de comunicação unidirecional entre dois gerenciadores de filas e pode transportar mensagens destinadas para qualquer número de filas no gerenciador de filas remotas.

Cada extremidade do canal possui uma definição separada. Por exemplo, se um final é um emissor ou um servidor, a outra extremidade deve ser um receptor ou um solicitante. Um canal simples consiste em uma *definição do canal emissor* no final do gerenciador de filas locais e uma *definição de canal receptor* no final do gerenciador de filas remotas. As duas definições devem ter o mesmo nome e juntas constituir um canal de mensagens único.

Se você deseja que o gerenciador de filas remotas responda às mensagens enviadas pelo gerenciador de filas locais, configure um segundo canal para enviar respostas de volta ao gerenciador de filas locais.

Use o comando DEFINE CHANNEL do MQSC para definir os canais. Nesta seção, os exemplos relacionados aos canais usam os atributos do canal padrão a menos que seja especificado de outra forma.

Há um agente do canal de mensagens (MCA) em cada extremidade de um canal, controlando o envio e o recebimento de mensagens. O MCA obtém as mensagens da fila de transmissão e coloca-as no link de comunicação entre os gerenciadores de filas.

Uma fila de transmissão é uma fila local especializada que mantém temporariamente as mensagens antes que o MCA as selecione e as envie para o gerenciador de filas remotas. Você especifica o nome da fila de transmissão em um *definição de fila remota*.

É possível permitir que um MCA transfira mensagens utilizando diversos encadeamentos. Esse processo é conhecido como *enfileiramento*. O enfileirando permite que o MCA transfira mensagens com mais eficiência, aprimorando o desempenho do canal. Consulte Atributos de canais para obter detalhes de como configurar um canal para utilizar o enfileiramento.

"Preparando canais e filas de transmissão para administração remota" na página 110 informa como utilizar essas definições para configurar a administração remota.

Para obter mais informações sobre como configurar o enfileiramento distribuído em geral, consulte Componentes de enfileiramento distribuído.

Administração remota utilizando clusters

Em uma rede WebSphere MQ usando enfileiramento distribuído, cada gerenciador de filas é independente. Se um gerenciador de filas precisar enviar as mensagens para outro gerenciador de filas, ele deve definir uma fila de transmissão, um canal para o gerenciador de filas remotas e uma definição de fila remota para cada fila para a qual deseja enviar as mensagens.

Um *cluster* é um grupo de gerenciadores de filas configurados de tal modo que os gerenciadores de filas podem se comunicar diretamente entre si através de uma rede única sem fila de transmissão complexa, canal e definições de fila. Os clusters podem ser configurados facilmente e geralmente contêm gerenciadores de filas que estão logicamente relacionados de alguma maneira e precisam compartilhar dados ou aplicativos. Mesmo o menor cluster reduz os custos de administração do sistema.

Estabelecer uma rede de gerenciadores de filas em um cluster envolve menos definições do que estabelecer um ambiente de enfileiramento distribuído tradicional. Com menos definições a fazer, é possível configurar ou alterar sua rede mais rápida e facilmente e reduzir o risco de cometer um erro em suas definições.

Para configurar um cluster, você precisa de um emissor de cluster (CLUSSDR) e a definição de um receptor de cluster (CLUSRCVR) para cada gerenciador de filas. Você não precisa de nenhuma definição de fila de transmissão ou definições de fila remota. Os princípios da administração remota são os mesmos quando utilizados dentro de um cluster, mas as próprias definições são bem mais simplificadas.

Para obter mais informações sobre clusters, seus atributos e como configurá-los, consulte Clusters do Gerenciador de Filas.

Administração Remota de um Gerenciador de Filas Locais

Esta seção informa como administrar um gerenciador de filas remotas a partir de um gerenciador de filas locais usando comandos MQSC e PCF.

Preparar filas e canais é essencialmente igual para ambos os comandos MQSC e PCF. Nesta seção, os exemplos mostram comandos MQSC, porque são mais fáceis de serem entendidos. Para obter informações adicionais sobre a gravação de programas de administração usando comandos PCF, consulte "Usando formatos de comando programáveis" na página 10.

Envie comandos MQSC para um gerenciador de filas remotas interativamente ou a partir de um arquivo de texto que contenha os comandos. O gerenciador de filas remotas pode estar na mesma máquina ou, mais normalmente, em uma máquina diferente. É possível administrar remotamente gerenciadores de filas em outros ambientes do WebSphere MQ , incluindo sistemas UNIX and Linux , sistemas Windows , IBM ie z/OS

Para implementar a administração remota, você deve criar objetos específicos. A menos que tenha requisitos especializados, os valores padrão (por exemplo, para o comprimento máximo da mensagem) são suficientes.

Preparando gerenciadores de filas para administração remota

Como utilizar os comandos do MQSC para preparar gerenciadores de filas para administração remota.

Figura 17 na página 109 mostra a configuração de gerenciadores de filas e canais que você precisa para administração remota utilizando o comando **runmqsc**. O objeto `source.queue.manager` é o gerenciador de filas de origem do qual é possível emitir comandos MQSC e para o qual os resultados desses comandos (mensagens do operador) são retornados. O objeto `target.queue.manager` é o nome do gerenciador de filas de destino, que processa os comandos e gera as mensagens do operador.

Nota: Se você estiver usando **runmqsc** com a opção -w, `source.queue.manager` ***deve*** ser o gerenciador de filas padrão. Para obter informações adicionais sobre a criação de um gerenciador de filas, consulte crtmqm.

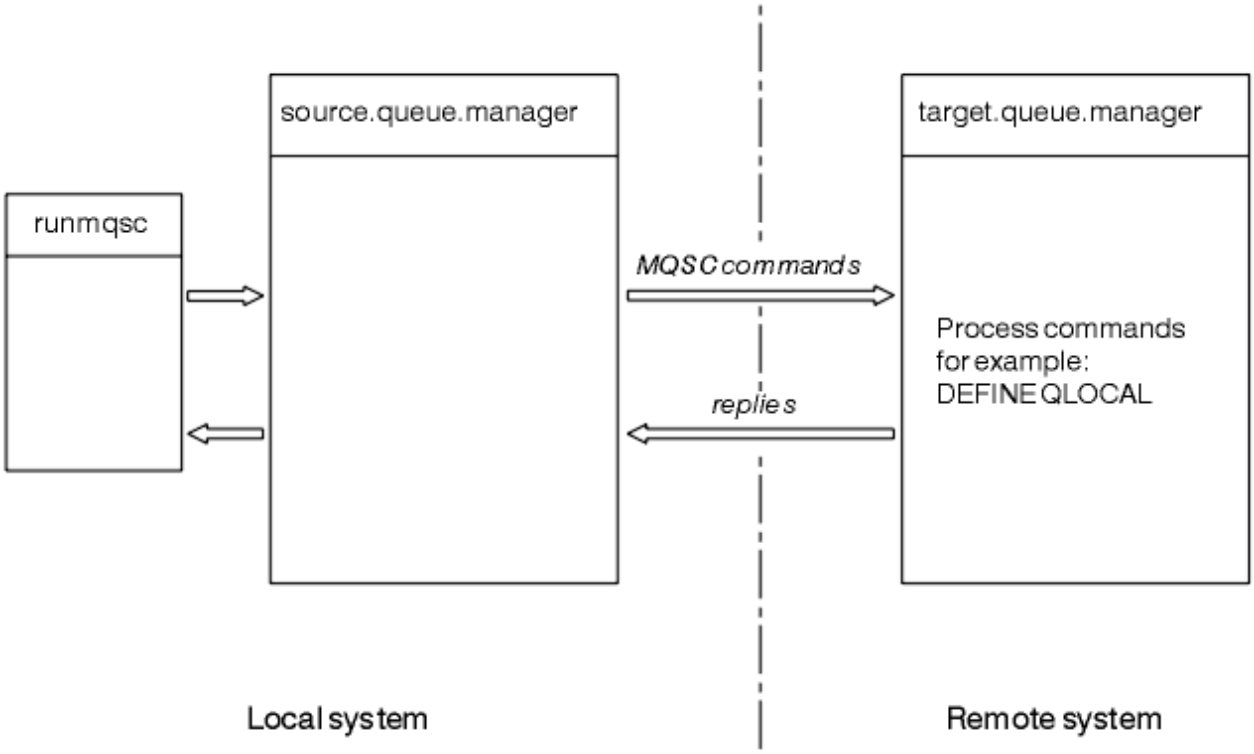


Figura 17. Administração remota usando comandos MQSC

Em ambos os sistemas, se você ainda não tiver feito isso:

- Crie o gerenciador de filas e os objetos padrão, utilizando o comando `crtmqm`.
- Inicie o gerenciador de filas, usando o comando `strmqm`.

No gerenciador de filas de destino:

- A fila de comandos, SYSTEM.ADMIN.COMMAND.QUEUE, deve estar presente. Esta fila é criada por padrão quando um gerenciador de filas é criado.

É necessário executar esses comandos localmente ou através de uma instalação de rede, como Telnet.

Preparando canais e filas de transmissão para administração remota

Como utilizar os comandos do MQSC para preparar canais e filas de transmissão para administração remota.

Para executar comandos MQSC remotamente, configure dois canais, um para cada direção e suas filas de transmissão associadas. Este exemplo presume que você esteja utilizando o TCP/IP como o tipo de transporte e que você saiba o endereço TCP/IP envolvido.

O canal `source.to.target` é para enviar comandos MQSC do gerenciador de filas de origem para o gerenciador de filas de destino. O emissor está em `source.queue.manager` e seu receptor está em `target.queue.manager`. O canal `target.to.source` é para retornar a saída dos comandos e quaisquer mensagens do operador que são gerados para o gerenciador de filas de origem. Também deve-se definir uma fila de transmissão para cada canal. Esta fila é uma fila local que recebe o nome do gerenciador de filas de recebimento. O nome XMITQ deve corresponder ao nome do gerenciador de filas remotas para que a administração remota funcione, a menos que você esteja utilizando um alias de gerenciador de filas.Figura 18 na página 110 resume essa configuração.

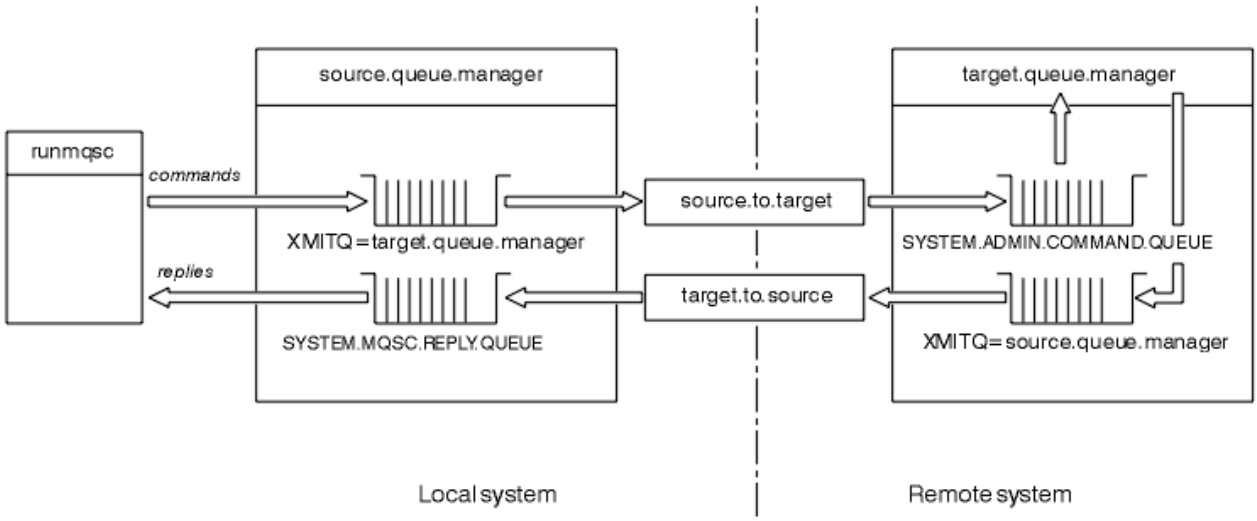


Figura 18. Configurando canais e filas para administração remota

Consulte Conectando aplicativos usando enfileiramento distribuído para obter mais informações sobre como configurar canais.

Definindo canais, listeners e filas de transmissão

No gerenciador de filas de origem (`source.queue.manager`), emita os comandos MQSC a seguir para definir os canais, o listener e a fila de transmissão:

1. Defina o canal emissor no gerenciador de filas de origem:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(SDR) +
       CONNAME (RHX5498) +
       XMITQ ('target.queue.manager') +
       TRPTYPE(TCP)
```

2. Defina o canal receptor no gerenciador de filas de origem:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Defina o listener no gerenciador de filas de origem:

```
DEFINE LISTENER ('source.queue.manager') +
       TRPTYPE (TCP)
```

4. Defina a fila de transmissão no gerenciador de filas de origem:

```
DEFINE QLOCAL ('target.queue.manager') +
       USAGE (XMITQ)
```

Emita os seguintes comandos no gerenciador de filas de destino (`target.queue.manager`), para criar os canais, o listener e a fila de transmissão:

1. Defina o canal emissor no gerenciador de filas de destino:

```
DEFINE CHANNEL ('target.to.source') +
       CHLTYPE(SDR) +
       CONNAME (RHX7721) +
       XMITQ ('source.queue.manager') +
       TRPTYPE(TCP)
```

2. Defina o canal receptor no gerenciador de filas de destino:

```
DEFINE CHANNEL ('source.to.target') +
       CHLTYPE(RCVR) +
       TRPTYPE(TCP)
```

3. Defina o listener no gerenciador de filas de destino:

```
DEFINE LISTENER ('target.queue.manager') +
       TRPTYPE (TCP)
```

4. Defina a fila de transmissão no gerenciador de filas de destino:

```
DEFINE QLOCAL ('source.queue.manager') +
       USAGE (XMITQ)
```

Nota: Os nomes de conexão TCP/IP especificados para o atributo CONNAME nas definições de canal emissor são para ilustração somente. Este é o nome da rede da máquina na *outra* extremidade da conexão. Use os valores apropriados para sua rede.

Iniciando os listeners e os canais

Como utilizar os comandos do MQSC para iniciar listeners e canais.

Inicie ambos os listeners usando os comandos MQSC a seguir:

1. Inicie o listener no gerenciador de filas de origem, `source.queue.manager`, emitindo o comando MQSC a seguir:

```
START LISTENER ('source.queue.manager')
```

2. Inicie o listener no gerenciador de filas de destino, `target.queue.manager`, emitindo o comando MQSC a seguir:

```
START LISTENER ('target.queue.manager')
```

Inicie os canais do emissor usando os comandos MQSC a seguir:

1. Inicie o canal emissor no gerenciador de filas de origem, `source.queue.manager`, emitindo o comando MQSC a seguir:

```
START CHANNEL ('source.to.target')
```

2. Inicie o canal emissor no gerenciador de filas de destino, `target.queue.manager`, emitindo o comando MQSC a seguir:

```
START CHANNEL ('target.to.source')
```

Definição automática de canais
Você ativa a definição automática das definições de conexão de receptor e servidor atualizando o objeto do gerenciador de filas usando o comando MQSC, ALTER QMGR (ou o comando Mudar gerenciador de filas do PCF).

Se o WebSphere MQ receber um pedido de conexão de entrada e não puder localizar um canal receptor ou de conexão do servidor apropriado, ele criará um canal automaticamente. As definições automáticas são baseadas em duas definições padrão fornecidas com o WebSphere MQ: SYSTEM.AUTO.RECEIVER e SYSTEM.AUTO.SVRCONN.

Para obter mais informações sobre como criar definições de canal automaticamente, consulte Preparando canais. Para obter informações sobre como definir automaticamente os canais para clusters, consulte Definição automática de canais de clusters

Gerenciando o servidor de comandos para administração remota

Como iniciar, parar e exibir o status do servidor de comandos. Um servidor de comandos é obrigatório para toda a administração que envolve os comandos PCF, a MQAI e também para administração remota.

Cada gerenciador de filas pode ter um servidor de comandos associado a ele. Um servidor de comandos processa qualquer comando de entrada a partir de gerenciadores de filas remotas ou comandos PCF a partir de aplicativos. Ele apresenta os comandos para o gerenciador de filas para processamento e retorna um código de conclusão ou mensagem do operador dependendo da origem do comando.

Nota: Para administração remota, certifique-se de que o gerenciador de filas de destino esteja em execução. Caso contrário, as mensagens que contêm comandos não podem deixar o gerenciador de filas a partir do qual elas são emitidas. Em vez disso, essas mensagens são enfileiradas na fila de transmissão local que serve ao gerenciador de filas remotas. Evite esta situação.

Há comandos de controle separados para iniciar e parar o servidor de comandos. Desde que o servidor de comando esteja em execução, os usuários do WebSphere MQ para Windows ou WebSphere MQ para Linux (plataformasx86 e x86-64) podem executar as operações descritas nas seguintes seções usando o IBM WebSphere MQ Explorer. Para obter mais informações, consulte “Administração usando o IBM WebSphere MQ Explorer” na página 57.

Iniciando o Servidor de Comandos

Dependendo do valor do atributo do gerenciador de filas, *SCMDSERV*, o servidor de comandos é iniciado automaticamente quando o gerenciador de filas é iniciado ou deve ser iniciado manualmente. O valor do atributo do gerenciador de filas pode ser alterado usando o comando MQSC `ALTER QMGR` especificando o parâmetro SCMDSERV. Por padrão, o servidor de comandos é iniciado automaticamente.

Se *SCMDSERV* é configurado como MANUAL, inicie o servidor de comando utilizando o comando:

```
strmqcsv saturn.queue.manager
```

em que `saturn.queue.manager` é o gerenciador de filas para o qual o servidor de comandos está sendo iniciado.

Exibindo o status do servidor de comandos

Para administração remota, assegure que o servidor de comandos no gerenciador de filas de destino esteja em execução. Se ele não estiver em execução, os comandos remotos não podem ser processados. Todas as mensagens contendo comandos são enfileiradas na fila de comandos do gerenciador de filas de destino.

Para exibir o status do servidor de comandos para um gerenciador de filas, emita o comando MQSC a seguir:

```
DISPLAY QMSTATUS CMDSERV
```

Parando um servidor de comandos

Para finalizar o servidor de comandos iniciado pelo exemplo anterior, use o comando a seguir:

```
endmqcsv saturn.queue.manager
```

É possível parar o servidor de comandos de duas maneiras:

- Para obter uma parada controlada, use o comando `endmqcsv` com o sinalizador -c, que é o padrão.
- Para obter uma parada imediata, use o comando `endmqcsv` com o sinalizador -i.

Nota: Parar um gerenciador de filas também encerra o servidor de comandos associado a ele.

Emitindo comandos MQSC em um gerenciador de filas remotas

É possível utilizar um formulário específico do comando **runmqsc** para executar comandos MQSC em um gerenciador de filas remotas.

O comando do servidor ***deve*** ser executado no gerenciador de filas de destino, se ele irá processar comandos MQSC remotamente. (Isso não é necessário no gerenciador de filas de origem). Para obter informações sobre como iniciar o servidor de comandos em um gerenciador de filas, consulte “Gerenciando o servidor de comandos para administração remota” na página 112.

No gerenciador de filas de origem, é possível, então, executar comandos MQSC interativamente no modo indireto digitando:

```
runmqsc -w 30 target.queue.manager
```

Este formato do comando `runmqsc`, com o sinalizador -w, executa os comandos MQSC no modo indireto, no qual os comandos são colocados (em um formato modificado) na fila de entrada de comando do servidor e executados em ordem.

Ao digitar em um comando do MQSC, ele é redirecionado para o gerenciador de filas remotas, neste caso, `target.queue.manager`. O tempo limite é configurado para 30 segundos; se uma resposta não é recebida dentro de 30 segundos, a mensagem a seguir é gerada no gerenciador de filas locais (origem):

```
AMQ8416: MQSC timed out waiting for a response from the command server.
```

Ao parar de emitir comandos do MQSC, o gerenciador de filas local exibe quaisquer respostas de tempo limite que chegaram e descarta quaisquer respostas adicionais.

Os padrões do gerenciador de filas de origem para o gerenciador de filas locais padrão. Se você especificar a opção `-m` *LocalQmgrName* no comando **runmqsc**, é possível direcionar os comandos a serem emitidos por meio de qualquer gerenciador de filas locais.

No modo indireto, também é possível executar um arquivo de comando MQSC em um gerenciador de filas remotas. Por exemplo:

```
runmqsc -w 60 target.queue.manager < mycomds.in > report.out
```

em que `mycomds.in` é um arquivo que contém comandos MQSC e `report.out` é o arquivo de relatório.

Método sugerido para emitir comandos remotamente

Quando você estiver emitindo comandos em um gerenciador de filas remotas, considere utilizar a seguinte abordagem:

1. Coloque os comandos MQSC a serem executados no sistema remoto em um arquivo de comando.
2. Verifique seus comandos MQSC localmente, especificando o sinalizador -v no comando `runmqsc`.

   Não é possível usar `runmqsc` para verificar os comandos MQSC em outro gerenciador de filas.
3. Verifique se o arquivo de comando é executado localmente sem erro.
4. Execute o arquivo de comando no sistema remoto.

Se você tiver problemas utilizando comandos MQSC remotamente

Se você tiver dificuldade em executar comandos MQSC remotamente, certifique-se de que você tenha:

- Iniciado o servidor de comandos no gerenciador de filas de destino.
- Definido uma fila de transmissão válido.
- Definido as duas extremidades dos canais de mensagens para ambos:
  - O canal ao longo do qual os comandos estão sendo enviados.
  - O canal ao longo do qual as respostas devem ser retornadas.
- Especificado o nome correto de conexão (CONNAME) na definição de canal.
- Iniciado os listeners antes de iniciar os canais de mensagens.
- Verificado se o intervalo de desconexão não expirou, por exemplo, se um canal foi iniciado mas, em seguida, encerrado após algum tempo. Isso é especialmente importante se você iniciar os canais manualmente.
- Enviado solicitações de um gerenciador de filas de origem que não fazem sentido para o gerenciador de filas de destino (por exemplo, as solicitações que incluem parâmetros que não são suportados no gerenciador de filas remotas).

Consulte também “Resolvendo problemas com os comandos do MQSC” na página 82.

Criando uma definição local de uma fila remota

Uma definição local de uma fila remota é uma definição em um gerenciador de filas locais que se refere a uma fila em um gerenciador de filas remotas.

Você não precisa definir uma fila remota a partir de uma posição local, mas a vantagem de fazer isso é que os aplicativos podem referir-se à fila remota por seu nome definido localmente em vez de ter que especificar um nome que é qualificado pelo ID do gerenciador de filas no qual a fila remota está localizada.

Entendendo como as definições locais de filas remotas funcionam

Um aplicativo conecta-se a um gerenciador de filas locais e, em seguida, emite uma chamada MQOPEN. Na chamada aberta, o nome da fila especificado é aquele de uma definição de fila remota no gerenciador de filas locais. A definição de fila remota fornece os nomes da fila de destino, o gerenciador de filas de destino e, opcionalmente, uma fila de transmissão. Para colocar uma mensagem na fila remota, o aplicativo emite uma chamada MQPUT, especificando o identificador retornado da chamada MQOPEN. O gerenciador de filas usa o nome da fila remota e o nome do gerenciador de filas remotas em um cabeçalho de transmissão no início da mensagem. Estas informações são usadas para rotear a mensagem para seu destino correto na rede.

Como administrador, é possível controlar o destino da mensagem, mudando a definição de fila remota.

O exemplo a seguir mostra como um aplicativo põe uma mensagem em uma fila pertencente a um gerenciador de filas remotas. O aplicativo se conecta a um gerenciador de filas, por exemplo, `saturn.queue.manager`. A fila de destino pertence a outro gerenciador de filas.

Na chamada MQOPEN, o aplicativo especifica estes campos:

Valor do Campo	Descrição
ObjectName CYAN.REMOTE.QUEUE	Especifica o nome local do objeto de fila remota. Define a fila de destino e o gerenciador de filas de destino.
ObjectType (Queue)	Identifica esse objeto como uma fila.
ObjectQmgrName Blank or `saturn.queue.manager`	Este campo é opcional. Se estiver em branco, o nome do gerenciador de fila local será assumido. (Este é o gerenciador de filas no qual a definição de fila remota existe.)

Depois disso, o aplicativo emite uma chamada MQPUT para colocar uma mensagem nessa fila.

No gerenciador de filas locais, é possível criar uma definição local de uma fila remota utilizando os seguintes comandos do 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)
```

em que:

QREMOTE (CYAN.REMOTE.QUEUE)
Especifica o nome local do objeto de fila remota. Esse é o nome que os aplicativos conectados a este gerenciador de filas devem especificar na chamada MQOPEN para abrir a fila AUTOMOBILE.INSURANCE.QUOTE.QUEUE no gerenciador de filas remotas do `jupiter.queue.manager`.

DESCR ('Queue for auto insurance requests from the branches')
Fornece texto adicional que descreve o uso da fila.

RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)
Especifica o nome da fila de destino no gerenciador de filas remotas. Esta é a fila de destino real para mensagens enviadas por aplicativos que especificam o nome da fila CYAN.REMOTE.QUEUE. A fila AUTOMOBILE.INSURANCE.QUOTE.QUEUE deve ser definida como uma fila local no gerenciador de filas remotas.

RQMNAME (jupiter.queue.manager)
Especifica o nome do gerenciador de filas remotas que tem a fila de destino AUTOMOBILE.INSURANCE.QUOTE.QUEUE.

XMITQ (INQUOTE.XMIT.QUEUE)
Especifica o nome da fila de transmissão. Isso é opcional; se o nome de uma fila de transmissão não for especificado, uma fila com o mesmo nome que o gerenciador de filas remoto é utilizada.

Em qualquer um dos casos, a fila de transmissão apropriada deve ser definida como uma fila local com um atributo *Usage* que especifica que é uma fila de transmissão (USAGE(XMITQ) em comandos MQSC).

Uma maneira alternativa de colocar mensagens em uma fila remota

Utilizar uma definição local de uma fila remota não é a única maneira de colocar mensagens em uma fila remota. Os aplicativos podem especificar o nome da fila completo, incluindo o nome do gerenciador de filas remotas, como parte da chamada MQOPEN. Neste caso, você não precisa de uma definição local de uma fila remota. No entanto, isto significa que os aplicativos devem conhecer ou ter acesso ao nome do gerenciador de filas remotas no tempo de execução.

Usando outros comandos com filas remotas

É possível utilizar os comandos do MQSC para exibir ou alterar os atributos de um objeto de fila remoto ou é possível excluir o objeto de fila remota. Por exemplo:

- Para exibir os atributos da fila remota:

```
DISPLAY QUEUE (CYAN.REMOTE.QUEUE)
```

- Para mudar a fila remota para ativar puts. Isso não afeta a fila de destino, somente os aplicativos que especificam esta fila remota:

```
ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)
```

- Para excluir esta fila remota. Isso não afeta a fila de destino, somente sua definição de local:

```
  DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

Nota: Quando você exclui uma fila remota, você excluir somente a representação local da fila remota. Você não exclui a fila remota em si ou quaisquer mensagens nela.

Definindo uma fila de transmissão

Uma fila de transmissão é uma fila local que é usada quando um gerenciador de filas encaminha mensagens para um gerenciador de filas remotas através de um canal de mensagens.

O canal fornece um link unidirecional para o gerenciador de filas remotas. As mensagens são enfileiradas na fila de transmissão até que o canal possa aceitá-las. Ao definir um canal, deve-se especificar um nome da fila de transmissão na extremidade de envio do canal de mensagem.

O atributo USAGE do comando MQSC define se uma fila é uma fila de transmissão ou uma fila normal.

Filas de transmissão padrão

Quando um gerenciador de filas envia mensagens para um gerenciador de filas remotas, ele identifica a fila de transmissão utilizando a seguinte sequência:

1. A fila de transmissão nomeada no atributo XMITQ da definição local de uma fila remota.
2. Uma fila de transmissão com o mesmo nome que o gerenciador de filas de destino. (Esse valor é o valor padrão no XMITQ da definição local de uma fila remota.)
3. A fila de transmissão nomeada no atributo DEFXMITQ do gerenciador de fila local.

Por exemplo, o comando MQSC a seguir cria uma fila de transmissão padrão no `source.queue.manager` para mensagens que vão para `target.queue.manager`:

```
DEFINE QLOCAL ('target.queue.manager') +
       DESCR ('Default transmission queue for target qm') +
       USAGE (XMITQ)
```

Os aplicativos podem colocar mensagens diretamente em uma fila de transmissão ou indiretamente por meio de uma definição de fila remota. Consulte também “Criando uma definição local de uma fila remota” na página 114.

Utilizando definições de filas remotas como aliases

Além de localizar uma fila em outro gerenciador de filas, também é possível utilizar uma definição local de uma fila remota para aliases do gerenciador de filas e aliases da fila de resposta. Ambos os tipos de alias são resolvidas através da definição local de uma fila remota. Deve-se configurar os canais apropriados para a mensagem chegar em seu destino.

Aliases do gerenciador de filas

Um alias é o processo pelo qual o nome do gerenciador de filas de destino, conforme especificado em uma mensagem, é mudado por um gerenciador de filas na rota da mensagem.aliases do gerenciador de filas são importantes porque é possível utilizá-las para controlar o destino de mensagens em uma rede de gerenciadores de filas.

Você faz isso mudando a definição de fila remota no gerenciador de filas no ponto de controle. O aplicativo de envio não está ciente de que o nome do gerenciador de filas especificado é um alias.

Para obter mais informações sobre aliases do gerenciador de filas, consulte O que são aliases?.

Aliases da Fila de Resposta

Opcionalmente, um aplicativo pode especificar o nome de uma fila de resposta quando ela coloca uma *mensagem de solicitação* em uma fila.

Se o aplicativo que processa a mensagem extrai o nome da fila de resposta, ele saiba onde enviar as *mensagem de resposta*, se necessário.

Um alias da fila de resposta é o processo pelo qual uma fila de resposta, conforme especificado em uma mensagem de solicitação, for alterado por um gerenciador de filas na rota da mensagem. O aplicativo de envio não está ciente de que o nome da fila de resposta especificado é um alias.

Um alias da fila de resposta permite que você altere o nome da fila de resposta e, opcionalmente, seu gerenciador de filas. Isso, por sua vez, permite que você controle qual rota é utilizada para mensagens de resposta.

Para obter mais informações sobre mensagem de solicitação, mensagens de resposta e filas de resposta, consulte Tipos de mensagem e Fila de resposta e gerenciador de filas.

Para obter mais informações sobre os aliases da fila de resposta, consulte Aliases e clusters da fila de resposta

Conversão de dados entre conjuntos de caracteres codificados

Os dados da mensagem nos formatos definidos do WebSphere MQ (também conhecidos como *formatos integrados*) podem ser convertidos pelo gerenciador de filas de um conjunto de caracteres codificados para outro, desde que ambos os conjuntos de caracteres estejam relacionados a um único idioma ou a um grupo de idiomas semelhantes.

Por exemplo, a conversão entre conjuntos de caracteres codificados com identificadores (CCSIDs) 850 e 500 é suportada porque ambos se aplicam aos idiomas europeus ocidentais.

Para conversões de caracteres EBCDIC newline (NL) para ASCII, consulte Todos os gerenciadores de filas .

Conversões suportadas são definidas em Conversão de dados.

Quando um gerenciador de filas não puder converter mensagens em formatos integrados

O gerenciador de filas não pode converter mensagens em formatos integrados automaticamente se seus CCSIDs representam diferentes grupos de idioma nacional. Por exemplo, a conversão entre o CCSID 850 e o CCSID 1025 (que é um conjunto de caracteres EBCDIC para idiomas codificados utilizando script cirílico) não é suportada porque muitos dos caracteres em um conjunto de caracteres codificado

não podem ser representados no outro. Se você tiver uma rede de gerenciadores de filas funcionando em diferentes idiomas nacionais e a conversão de dados entre alguns dos conjuntos de caracteres codificados não for suportada, é possível ativar uma conversão padrão. Conversão de dados padrão é descrita em “Conversão de dados padrão” na página 118.

Arquivo ccsid.tbl

O arquivo ccsid.tbl é utilizado para as finalidades a seguir:

- No WebSphere MQ para Windows , ele registra todos os conjuntos de códigos suportados
- No AIX e plataformas HP-UX, os conjuntos de códigos suportados são mantidos internamente pelo sistema operacional.
- Para todas as outras plataformas do UNIX and Linux , os conjuntos de códigos suportados são mantidos nas tabelas de conversão fornecidas pelo WebSphere MQ
- Ele especifica quaisquer conjuntos de códigos adicionais. Para especificar conjuntos de códigos adicionais, você precisa editar ccsid.tbl (orientação sobre como fazer isso é fornecida no arquivo).
- Ele especifica qualquer conversão de dados padrão.

É possível atualizar as informações registradas no ccsid.tbl; é possível desejar fazer isto se, por exemplo, uma liberação futura do seu sistema operacional suportar conjuntos de caracteres codificados adicionais.

Em WebSphere MQ para Janelas, ccsid.tbl está localizado no diretório `C:\Program Files\IBM\WebSphere MQ\conv\table` por padrão..

Em WebSphere MQ para UNIX and Linux sistemas, ccsid.tbl está localizado no diretório `/var/mqm/conv/table`..

Conversão de dados padrão

Se você configurar canais entre duas máquinas nas quais a conversão de dados normalmente não é suportada, deve-se ativar a conversão de dados padrão para os canais funcionarem.

Para ativar a conversão de dados padrão, edite o arquivo ccsid.tbl para especificar um EBCDIC CCSID padrão e ASCII CCSID padrão. Instruções sobre como fazer isto estão incluídas no arquivo. Deve-se fazer isso em todas as máquinas que serão conectadas utilizando os canais. Reinicie o gerenciador de filas para que a alteração entre em vigor.

O padrão de conversão de dados do processo é o seguinte:

- Se a conversão entre os CCSIDs de origem e destino não é suportada, mas os CCSIDs dos ambientes de origem e destino são ambos EBCDIC ou ASCII, os dados de caracteres são transmitidos para o aplicativo de destino sem conversão.
- Se um CCSID representar um conjunto de caracteres codificados ASCII e o outro representar um conjunto de caracteres codificados EBCDIC, o WebSphere MQ converterá os dados usando os CCSIDs de conversão de dados padrão definidos em ccsid.tbl.

Nota: Tente restringir os caracteres que estão sendo convertidos àqueles que têm os mesmos valores de código no conjunto de caracteres codificados especificado para a mensagem e no conjunto de caracteres codificados padrão. Se você usar apenas o conjunto de caracteres que é válido para nomes de objetos do WebSphere MQ (conforme definido em Nomenclatura IBM WebSphere MQ objetos), você, em geral, atenderá a este requisito. Exceções ocorrem com os CCSIDs do EBCDIC 290, 930, 1279 e 5026 utilizados no Japão, onde os caracteres minúsculos possuem códigos diferentes daqueles utilizados em outros CCSIDs do EBCDIC.

Convertendo mensagens em formatos definidos pelo usuário

O gerenciador de filas não pode converter mensagens em formatos definidos pelo usuário a partir de um conjunto de caracteres codificado para outro. Se você precisar converter os dados em um formato definido pelo usuário, deve-se fornecer uma saída de conversão de dados para cada formato desse. Não utilize CCSIDs padrão para converter dados de caractere em formatos definidos pelo usuário. Para obter

mais informações sobre a conversão de dados em formatos definidos pelo usuário e sobre a gravação de saídas de conversão de dados, consulte Gravando saídas de conversão de dados.

Mudando o CCSID do gerenciador de filas

Quando você tiver utilizado o atributo CCSID do comando ALTER QMGR para mudar o CCSID do gerenciador de filas, pare e reinicie o gerenciador de filas para garantir que todos os aplicativos em execução, incluindo o servidor de comandos e os programas de canal, sejam parados e reiniciados.

Isso é necessário porque todos os aplicativos que estão em execução quando o gerenciador de filas CCSID é mudado continuam a utilizar o CCSID existente.

Administrar o IBM WebSphere MQ Telemetry

IBM WebSphere MQ Telemetry é administrado usando IBM WebSphere MQ Explorer ou em uma linha de comandos. Use o Explorer para configurar canais de telemetria, controlar o serviço de telemetria e monitorar os clientes MQTT que estão conectados ao IBM WebSphere MQ. Configure a segurança do IBM WebSphere MQ Telemetry usando JAAS, SSL e o gerenciador de autoridade de objeto do IBM WebSphere MQ.

Administrando usando o IBM WebSphere MQ Explorer

Use o Explorer para configurar canais de telemetria, controlar o serviço de telemetria e monitorar os clientes MQTT que estão conectados ao IBM WebSphere MQ. Configure a segurança do IBM WebSphere MQ Telemetry usando JAAS, SSL e o gerenciador de autoridade de objeto do IBM WebSphere MQ.

Administrando Usando a Linha de Comandos

IBM WebSphere MQ Telemetry pode ser administrado completamente na linha de comandos usando os comandos do IBM WebSphere MQ MQSC.

A documentação IBM WebSphere MQ Telemetry também possui scripts de amostra que demonstram o uso básico do aplicativo Cliente MQ Telemetry Transport v3 .

Leia e entenda as amostras em Programas de amostra do IBM WebSphere MQ Telemetry na seção Desenvolvendo aplicativos para o IBM WebSphere MQ Telemetry antes de usá-las.

Conceitos relacionados

WebSphere MQ Telemetry

“Configurando Enfileiramento Distribuído para Enviar Mensagens para Clientes MQTT” na página 123
Os aplicativos IBM WebSphere MQ podem enviar mensagens de clientes MQTT v3 publicando na assinatura criada por um cliente ou enviando uma mensagem diretamente. Qualquer que seja o método usado, a mensagem é colocada em `SYSTEM.MQTT.TRANSMIT.QUEUE`e enviada ao cliente pelo serviço de telemetria (MQXR). Há várias maneiras de colocar uma mensagem em `SYSTEM.MQTT.TRANSMIT.QUEUE`.

“Identificação, Autorização e Autenticação de Cliente MQTT” na página 126

“Autenticação de Canal de Telemetria Usando SSL” na página 133

“Privacidade de publicação em canais de telemetria” na página 135

“Configuração de SSL dos Clientes MQTT e Canais de Telemetria” na página 135

“Configuração JAAS do Canal de Telemetria” na página 140
Configure JAAS para autenticar o `Username` enviado pelo cliente.

“IBM WebSphere MQ Daemon de telemetria para conceitos de dispositivos” na página 142
O IBM WebSphere MQ Daemon de telemetria para dispositivos é um aplicativo cliente MQTT V3 avançado Use-o para armazenamento e encaminhamento de mensagens de outros clientes MQTT. Ele se conecta ao IBM WebSphere MQ como um cliente MQTT, mas também é possível conectar outros clientes MQTT a ele..

Tarefas relacionadas

“Configurando um gerenciador de filas para telemetria em Linux e AIX” na página 120

Siga estas etapas manuais para configurar um gerenciador de filas para executar o IBM WebSphere MQ Telemetry. É possível executar um procedimento automatizado para configurar uma configuração mais simples usando o suporte ao IBM WebSphere MQ Telemetry para o IBM WebSphere MQ Explorer.

"Configurando um gerenciador de filas para telemetria no Windows" na página 121
Siga estas etapas manuais para configurar um gerenciador de filas para executar o IBM WebSphere MQ Telemetry. É possível executar um procedimento automatizado para configurar uma configuração mais simples usando o suporte ao IBM WebSphere MQ Telemetry para o IBM WebSphere MQ Explorer.

Referências relacionadas

Propriedades MQXR

Configurando um gerenciador de filas para telemetria em Linux e AIX

Siga estas etapas manuais para configurar um gerenciador de filas para executar o IBM WebSphere MQ Telemetry. É possível executar um procedimento automatizado para configurar uma configuração mais simples usando o suporte ao IBM WebSphere MQ Telemetry para o IBM WebSphere MQ Explorer.

Antes de começar

1. Consulte Instalando IBM WebSphere MQ Telemetry para obter informações sobre como instalar o IBM WebSphere MQe o recurso IBM WebSphere MQ Telemetry .
2. Crie e inicie um gerenciador de filas. O gerenciador de filas é referido como *qMgr* nesta tarefa.
3. Como parte desta tarefa você configurar o serviço de telemetria (MQXR). As configurações de propriedade MQXR são armazenadas em um arquivo de propriedades específicas da plataforma: `mqxr_unix.properties`. Normalmente, não é necessário editar o arquivo de propriedades MQXR diretamente, porque quase todas as configurações podem ser configuradas por meio de comandos do administrador MQSC ou do MQ Explorer. Se você decidir editar o arquivo diretamente, pare o gerenciador de filas antes de fazer suas mudanças. Consulte MQXR propriedades.

Sobre esta tarefa

O suporte IBM WebSphere MQ Telemetry para IBM WebSphere MQ Explorer inclui um assistente e um procedimento de comando de amostra `sampleMQM`. Eles instalam uma configuração inicial usando o ID do usuário guest; consulte Verificando a instalação do IBM WebSphere MQ Telemetry usando o IBM WebSphere MQ Explorer e Programas de amostra do IBM WebSphere MQ Telemetry.

Siga as etapas desta tarefa para configurar o IBM WebSphere MQ Telemetry manualmente usando esquemas de autorização diferentes.

Procedimento

1. Abra uma janela de comando no diretório de amostra de telemetria.

   O diretório de amostra de telemetria é `/opt/mqm/mqxr/samples`.
2. Crie a fila de transmissão de telemetria.

   ```
   echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc
   qMgr
   ```

   Quando o serviço de telemetria (MQXR) é iniciado pela primeira vez, ele cria `SYSTEM.MQTT.TRANSMIT.QUEUE`.

   Ele é criado manualmente nesta tarefa, porque `SYSTEM.MQTT.TRANSMIT.QUEUE` deve existir antes que o serviço de telemetria (MQXR) seja iniciado, para autorizar o acesso a ele.
3. Configure a fila de transmissão padrão

   Quando o serviço de telemetria (MQXR) é iniciado pela primeira vez, ele não altera o gerenciador de filas para tornar `SYSTEM.MQTT.TRANSMIT.QUEUE` a fila de transmissão padrão.

Para transformar a SYSTEM.MQTT.TRANSMIT.QUEUE na fila de transmissão padrão, altere a propriedade da fila de transmissão padrão. Altere a propriedade usando IBM WebSphere MQ Explorer ou com o comando no exemplo a seguir:

```
echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
```

A alteração da fila de transmissão padrão pode interferir na configuração existente. O motivo para alterar a fila de transmissão padrão para SYSTEM.MQTT.TRANSMIT.QUEUE é facilitar o envio de mensagens diretamente para clientes MQTT. Sem alterar a fila de transmissão padrão, você deve incluir uma definição de fila remota ara cada cliente que recebe mensagens do IBM WebSphere MQ; consulte “Enviando uma mensagem para um cliente diretamente” na página 125.

4. Siga um procedimento em “Autorizando clientes MQTT a acessar objetos do WebSphere MQ” na página 127 para criar um ou mais IDs de usuário. Os IDs do usuário têm autoridade para publicar, assinar e enviar publicações para clientes MQTT.
5. Instalar o serviço de telemetria (MQXR)

   ```
   cat /opt/<install_dir>/mqxr/samples/installMQXRService_unix.mqsc | runmqsc qMgr
   ```

   Consulte também o código de exemplo em Figura 19 na página 121
6. Iniciar o serviço

   ```
   echo "START SERVICE(SYSTEM.MQXR.SERVICE)" | runmqsc qMgr
   ```

   O serviço de telemetria (MQXR) é iniciado automaticamente quando o gerenciador de filas é iniciado.

   Ele é iniciado manualmente nesta tarefa porque o gerenciador de filas já está em execução.
7. Usando IBM WebSphere MQ Explorer, configure canais de telemetria para aceitar conexões de clientes MQTT.

   Os canais de telemetria devem ser configurados de modo que suas identidades sejam um dos IDs de usuário definidos na etapa 4.

   Consulte também DEFINE CHANNEL (MQTT).
8. Verifique a configuração executando o cliente de amostra.

   Para o cliente de amostra trabalhar com seu canal de telemetria, o canal deve autorizar o cliente a publicar, subscrever-se e receber publicações. O cliente de amostra se conecta ao canal de telemetria na porta 1883 por padrão. Consulte também Programas de amostra do IBM WebSphere MQ Telemetry.

Exemplo

Figura 19 na página 121 mostra o comando **runmqsc** para criar SYSTEM.MQXR.SERVICE manualmente em 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

Configurando um gerenciador de filas para telemetria no Windows

Siga estas etapas manuais para configurar um gerenciador de filas para executar o IBM WebSphere MQ Telemetry. É possível executar um procedimento automatizado para configurar uma configuração mais simples usando o suporte ao IBM WebSphere MQ Telemetry para o IBM WebSphere MQ Explorer.

Antes de começar

1. Consulte Instalando IBM WebSphere MQ Telemetry para obter informações sobre como instalar o IBM WebSphere MQe o recurso IBM WebSphere MQ Telemetry .
2. Crie e inicie um gerenciador de filas. O gerenciador de filas é referido como *qMgr* nesta tarefa.
3. Como parte desta tarefa você configurar o serviço de telemetria (MQXR). As configurações de propriedade MQXR são armazenadas em um arquivo de propriedades específicas da plataforma: `mqxr_win.properties`. Normalmente, não é necessário editar o arquivo de propriedades MQXR diretamente, porque quase todas as configurações podem ser configuradas por meio de comandos do administrador MQSC ou do MQ Explorer. Se você decidir editar o arquivo diretamente, pare o gerenciador de filas antes de fazer suas mudanças. Consulte MQXR propriedades.

Sobre esta tarefa

O suporte IBM WebSphere MQ Telemetry para IBM WebSphere MQ Explorer inclui um assistente e um procedimento de comando de amostra `sampleMQM`. Eles instalam uma configuração inicial usando o ID do usuário guest; consulte Verificando a instalação do IBM WebSphere MQ Telemetry usando o IBM WebSphere MQ Explorer e Programas de amostra do IBM WebSphere MQ Telemetry.

Siga as etapas desta tarefa para configurar o IBM WebSphere MQ Telemetry manualmente usando esquemas de autorização diferentes.

Procedimento

1. Abra uma janela de comando no diretório de amostra de telemetria.

   O diretório de amostras de telemetria é *WMQ program installation directory*`\mqxr\samples`.
2. Crie a fila de transmissão de telemetria.

   ```
   echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
   ```

   Quando o serviço de telemetria (MQXR) é iniciado pela primeira vez, ele cria `SYSTEM.MQTT.TRANSMIT.QUEUE`.

   Ele é criado manualmente nesta tarefa, porque `SYSTEM.MQTT.TRANSMIT.QUEUE` deve existir antes que o serviço de telemetria (MQXR) seja iniciado, para autorizar o acesso a ele.
3. Configure a fila de transmissão padrão

   ```
   echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
   ```

   Figura 20. Configurando a Fila de Transmissão Padrão

   Quando o serviço de telemetria (MQXR) é iniciado pela primeira vez, ele não altera o gerenciador de filas para tornar `SYSTEM.MQTT.TRANSMIT.QUEUE` a fila de transmissão padrão.

   Para transformar a `SYSTEM.MQTT.TRANSMIT.QUEUE` na fila de transmissão padrão, altere a propriedade da fila de transmissão padrão. Altere a propriedade usando o IBM WebSphere MQ Explorer ou com o comando em Figura 20 na página 122.

   A alteração da fila de transmissão padrão pode interferir na configuração existente. O motivo para alterar a fila de transmissão padrão para `SYSTEM.MQTT.TRANSMIT.QUEUE` é facilitar o envio de mensagens diretamente para clientes MQTT. Sem alterar a fila de transmissão padrão, você deve incluir uma definição de fila remota ara cada cliente que recebe mensagens do IBM WebSphere MQ; consulte “Enviando uma mensagem para um cliente diretamente” na página 125.
4. Siga um procedimento em “Autorizando clientes MQTT a acessar objetos do WebSphere MQ” na página 127 para criar um ou mais IDs de usuário. Os IDs do usuário têm autoridade para publicar, assinar e enviar publicações para clientes MQTT.
5. Instalar o serviço de telemetria (MQXR)

```
type
installMQXRService_win.mqsc | runmqsc qMgr
```

6. Iniciar o serviço

```
echo START SERVICE(SYSTEM.MQXR.SERVICE) | runmqsc qMgr
```

   O serviço de telemetria (MQXR) é iniciado automaticamente quando o gerenciador de filas é iniciado.

   Ele é iniciado manualmente nesta tarefa porque o gerenciador de filas já está em execução.

7. Usando IBM WebSphere MQ Explorer, configure canais de telemetria para aceitar conexões de clientes MQTT.

   Os canais de telemetria devem ser configurados de modo que suas identidades sejam um dos IDs de usuário definidos na etapa 4.

   Consulte também DEFINE CHANNEL (MQTT).

8. Verifique a configuração executando o cliente de amostra.

   Para o cliente de amostra trabalhar com seu canal de telemetria, o canal deve autorizar o cliente a publicar, subscrever-se e receber publicações. O cliente de amostra se conecta ao canal de telemetria na porta 1883 por padrão. Consulte também Programas de amostra do IBM WebSphere MQ Telemetry.

Criando SYSTEM.MQXR.SERVICE Manualmente

Figura 21 na página 123 mostra o comando **runmqsc** para criar SYSTEM.MQXR.SERVICE manualmente no 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

Configurando Enfileiramento Distribuído para Enviar Mensagens para Clientes MQTT

Os aplicativos IBM WebSphere MQ podem enviar mensagens de clientes MQTT v3 publicando na assinatura criada por um cliente ou enviando uma mensagem diretamente. Qualquer que seja o método usado, a mensagem é colocada em SYSTEM.MQTT.TRANSMIT.QUEUEe enviada ao cliente pelo serviço de telemetria (MQXR). Há várias maneiras de colocar uma mensagem em SYSTEM.MQTT.TRANSMIT.QUEUE.

Publicando uma mensagem em resposta a uma assinatura do cliente MQTT

O serviço de telemetria (MQXR) cria uma subscrição em nome do cliente MQTT. O cliente é o destino para quaisquer publiou ações que correspondem à assinatura enviada pelo cliente. Os serviços de telemetria encaminham publicações correspondentes de volta para o cliente.

Um cliente MQTT é conectado ao WebSphere MQ como um gerenciador de filas, com seu nome do gerenciador de filas configurado para seu ClientIdentifier O destino para publicações serem enviadas para o cliente é uma fila de transmissão, SYSTEM.MQTT.TRANSMIT.QUEUE. O serviço de telemetria encaminha mensagens no SYSTEM.MQTT.TRANSMIT.QUEUE para os clientes MQTT, usando o nome do gerenciador de fila de destino como a chave para um cliente específico

O serviço de telemetria (MQXR) abre a fila de transmissão usando ClientIdentifier como o nome do gerenciador de filas. O serviço de telemetria (MQXR) transmite o identificador de objeto da fila

para o MQSUB chamada, para encaminhar as publicações que correspondem à assinatura do cliente. Na resolução de nome de objeto, o `ClientIdentifier` é criado como o nome do gerenciador de filas remotas e a fila de transmissão deve ser resolvida para SYSTEM.MQTT.TRANSMIT.QUEUE. Usando a resolução do nome do objeto padrão do WebSphere MQ , o *ClientIdentifier* é resolvido da seguinte forma; consulte Tabela 6 na página 124

1. *ClientIdentifier* corresponde a nada.

   ClientIdentifier é um nome do gerenciador de filas remotas. Ele não corresponde ao nome do gerenciador de filas locais, ao alias do gerenciador de filas ou a um nome da fila de transmissão.

   O nome da fila não é definido. Atualmente, o serviço de telemetria (MQXR) configura SYSTEM.MQTT.PUBLICATION.QUEUE como o nome da fila. Um cliente MQTT v3 não suporta filas, portanto o nome da fila resolvida é ignorado pelo cliente.

   A propriedade do gerenciador de filas locais, `Fila de transmissão padrão`, deve ser configurada como SYSTEM.MQTT.TRANSMIT.QUEUE, para que a publicação seja colocada em SYSTEM.MQTT.TRANSMIT.QUEUE para ser enviada para o cliente.

2. *ClientIdentifier* corresponde a um alias de gerenciador de filas chamado *ClientIdentifier*.

   ClientIdentifier é um nome do gerenciador de filas remotas. Ele corresponde ao nome de um alias de gerenciador de filas.

   O alias do gerenciador de filas deve ser definida com *ClientIdentifier* como o nome do gerenciador de filas remotas.

   Durante a configuração do nome da fila de transmissão na definição de alias do gerenciador de filas, não é necessário que a transmissão padrão seja configurada como SYSTEM.MQTT.TRANSMIT.QUEUE.

Tabela 6. Resolução de nome de um alias do gerenciador de filas MQTT

	Entrada		**Saída**		
ClientIdentifier	**Nome do gerenciador de filas**	**Nome da fila**	**Nome do gerenciador de filas**	**Nome da fila**	**Fila de transmissão**
Não corresponde a nada	*ClientIdentifier*	*undefined*	*ClientIdentifier*	*undefined*	Fila de transmissão padrão. SYSTEM.MQTT.TRANSMIT.QUEUE
Corresponde a um alias do gerenciador de filas denominado *ClientIdentifier*	*ClientIdentifier*	*undefined*	*ClientIdentifier*	*undefined*	SYSTEM.MQTT.TRANSMIT.QUEUE

Para obter mais informações sobre a resolução de nome, consulte Resolução de Nome.

Qualquer programa WebSphere MQ pode publicar no mesmo tópico. A publicação é enviada para seus assinantes, incluindo os clientes MQTT v3 que possuem uma assinatura para o tópico

Se um tópico administrativo for criado em um cluster, com o atributo CLUSTER(*clusterName*), qualquer aplicativo no cluster poderá publicar no cliente; por exemplo:

```
echo DEFINE TOPIC('MQTTExamples') TOPICSTR('MQTT Examples') CLUSTER(MQTT) REPLACE | runmqsc qMgr
```

Figura 22. Definindo um tópico de cluster no Windows

Nota: Não forneça ao SYSTEM.MQTT.TRANSMIT.QUEUE um atributo de cluster

Assinantes e publicadores do cliente MQTT podem se conectar a diferentes gerenciadores de filas. Os assinantes e os publiou adores podem fazer parte do mesmo ou luster ou ser conectados por uma hierarquia de publiou ação/assinatura. A publicação é entregue do publicador para o assinante usando o WebSphere MQ.

Enviando uma mensagem para um cliente diretamente

Uma alternativa para um cliente criar uma assinatura e receber uma publicação que corresponda ao tópico de assinatura, envie uma mensagem para um cliente MQTT v3 diretamente... Os aplicativos clientes MQTT V3 não podem enviar mensagens diretamente, mas outros aplicativos, como WebSphere MQ , podem.

O aplicativo do WebSphere MQ deve conhecer o `ClientIdentifier` do cliente MQTT v3 Como os clientes MQTT v3 não têm filas, o nome da fila de destino é transmitido para o método MQTT v3 application client `messageArrived` como um nome de tópico. Por exemplo, em um programa MQI, crie um descritor de objeto com o cliente como `ObjectQmgrName`:

```
MQOD.ObjectQmgrName = ClientIdentifier;
MQOD.ObjectName = name;
```

Figura 23. Descritor de Objeto MQI para enviar uma mensagem para um destino do cliente MQTT v3

Se o aplicativo for gravado usando JMS, crie um destino ponto a ponto; por exemplo:

```
javax.jms.Destination jmsDestination =
                      (javax.jms.Destination)jmsFactory.createQueue
                      ("queue://ClientIdentifier/name");
```

Figura 24. Destino JMS para enviar uma mensagem para um cliente MQTT v3

Para enviar uma mensagem não solicitada para um cliente MQTT use uma definição de fila remota. O nome do gerenciador de filas remotas deve ser resolvido para `ClientIdentifier` do cliente. A fila de transmissão deve resolver para SYSTEM.MQTT.TRANSMIT.QUEUE; consulte Tabela 7 na página 125. O nome da fila remota pode ser qualquer ou oisa. O cliente o reou ebe como uma sequência de tópicos.

Tabela 7. Resolução de nome de uma definição da fila remota do cliente MQTT

Entrada		Saída		
Nome da fila	**Nome do gerenciador de filas**	**Nome da fila**	**Nome do gerenciador de filas**	**Fila de transmissão**
Nome da definição de fila remota	Em branco ou nome do gerenciador de filas locais	Nome da fila remota usado como uma sequência de tópicos	*ClientIdentifier*	SYSTEM.MQTT.TRANSMIT.QUEUE

Se o cliente estiver conectado, a mensagem será enviada diretamente ao cliente MQTT, que chama o método `messageArrived` ; consulte messageArrived method.

Se o cliente tiver desconectado com uma sessão persistente, a mensagem será armazenada em SYSTEM.MQTT.TRANSMIT.QUEUE; consulte sessões stateless e stateful do MQTT . Ela é encaminhada para o cliente quando ele se reconecta novamente à sessão.

Se você enviar uma mensagem não persistente, ela será enviada para o cliente com qualidade de serviço "no máximo uma vez", QoS=0. Se você enviar uma mensagem persistente diretamente para um cliente, por padrão, ela será enviada com qualidade de serviço "exatamente uma vez", QoS=2. Como o cliente pode não ter um mecanismo de persistência, o cliente pode diminuir a qualidade de serviço que aceita para mensagens enviadas diretamente. Para abaixar a qualidade de serviço para mensagens enviadas diretamente a um cliente, faça uma assinatura do tópiou o DEFAULT.QoS. Especifique a qualidade de serviço máxima que o cliente pode suportar.

Identificação, Autorização e Autenticação de Cliente MQTT

O serviço de telemetria (MQXR) publica ou assina os tópicos do WebSphere MQ em nome de clientes MQTT usando canais do MQTT. O administrador do WebSphere MQ configura a identidade do canal MQTT que é usado para autorização do WebSphere MQ. O administrador pode definir uma identidade comum para o canal ou usar Username ou ClientIdentifier de um cliente conectado ao canal.

O serviço de telemetria (MQXR) pode autenticar o cliente usando o Username fornecido pelo cliente ou usando um certificado de cliente. O Username é autentiou do usando uma senha fornecida pelo cliente.

Resumindo: Identificação de cliente é a seleção da identidade do cliente. Dependendo do contexto, o cliente é identifiou do por ClientIdentifier, Username, uma identidade de cliente comum criada pelo administrador ou um certifiou do de cliente. O identifiou dor de cliente usado para verifiou ção de autenticidade não precisa ser o mesmo identifiou dor que é usado para autorização.

Programas clientes MQTT configuram Username e Password que são enviados para o servidor com o uso de um canal MQTT. Eles também podem configurar as propriedades SSL necessárias para criptografar e autentiou r a conexão. O administrador decide se irá autenticar o canal MQTT e como irá autenticar.

Para autorizar um cliente MQTT a acessar objetos do WebSphere MQ, autorize o ClientIdentifier ou Username do cliente ou autorize uma identidade de cliente comum. Para permitir que um cliente se conecte ao WebSphere MQ, autentique Username ou use um certificado de cliente. Configure JAAS para autentiou r Username e configure SSL para autentiou r um certifiou do de cliente.

Se você configurar Password no cliente, criptografe a conexão usando VPN ou configure o canal MQTT para usar SSL para manter a senha particular.

É difícil gerenciar certificados de cliente. Por essa razão, se os riscos associados à autenticação de senha forem aceitáveis, a autenticação de senha será usada na maioria das vezes para autenticar clientes.

Se houver uma maneira segura de gerenciar e armazenar o certificado de cliente, é possível contar com a autenticação de certificado. Porém, raramente os certificados podem ser gerenciados com segurança nos tipos de ambientes em que a telemetria é usada. Em vez disso, a autenticação dos dispositivos usando certificados de cliente é complementada pela autenticação de senhas de clientes no servidor. Devido a essa complexidade adicional, o uso de certifiou dos de cliente é restrito a apliou tivos altamente sensíveis. O uso de duas formas de autentiou ção é chamado de autentiou ção de dois fatores. Deve-se conhecer um dos fatores, como uma senha, e ter o outro, como um certifiou do.

Em um aplicativo altamente sensível, como um dispositivo chip-and-pin, o dispositivo é bloqueado durante a fabricação para evitar violação com hardware e software internos. Um certificado de cliente confiável com tempo limitado é copiado no dispositivo. O dispositivo é implementado no loou l onde deve ser usado. Outras autentiou ções são feitas ou da vez que o dispositivo é usado, por meio de senha ou de outro certifiou do de smart ou rd.

Identidade e Autorização de Cliente MQTT

Use ClientIdentifier, Username ou uma identidade de cliente comum para obter autorização para acessar objetos do WebSphere MQ.

O administrador do WebSphere MQ tem três opções para selecionar a identidade do canal MQTT. O administrador faz a escolha ao definir ou modificar o canal MQTT usado pelo cliente. A identidade é usada para autorizar o acesso aos tópicos do WebSphere MQ. As opções são:

1. O identificador de cliente.
2. Uma identidade que o administrador fornece para o canal.
3. O `Username` passado do cliente MQTT.

`Username` é um atributo da classe MqttConnectOptions. Ele deve ser configurado antes de o cliente se conectar ao serviço. Seu valor padrão é nulo.

Use o comando **setmqaut** do WebSphere MQ para selecionar quais objetos, e quais ações, estão autorizados para serem usados pela identidade associada ao canal MQTT. Por exemplo, para autorizar uma identidade de canal, `MQTTClient`, fornecida pelo administrador do gerenciador de filas, `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
```

Autorizando clientes MQTT a acessar objetos do WebSphere MQ

Siga estas etapas para autorizar os clientes MQTT a publicar e assinar os Objetos do WebSphere MQ . As etapas seguem quatro padrões de controle de acesso alternativos.

Antes de começar

Os clientes MQTT estão autorizados a acessar objetos no WebSphere MQ ao serem designados a uma identidade quando se conectam a um canal de telemetria. O administrador do WebSphere MQ configura o canal de telemetria usando o WebSphere MQ Explorer para fornecer a um cliente um dos três tipos de identidade:

1. `ClientIdentifier`
2. `Nome de usuário`
3. Um nome que o administrador designa ao canal.

Qualquer que seja o tipo utilizado, a identidade deve ser definida como WebSphere MQ como um principal pelo serviço de autorização instalado. O serviço de autorização padrão no Windows ou Linux é chamado de Object Authority Manager (OAM). Se você estiver usando o OAM, a identidade deverá ser definida como um ID do usuário.

Use a identidade para fornecer a um cliente ou coleção de clientes permissão para publicar ou assinar tópicos definidos no WebSphere MQ. Se um cliente MQTT tiver assinado um tópico, use a identidade para dar a ele permissão para receber as publicações resultantes.

É difícil gerenciar um sistema com dezenas de milhares de clientes MQTT, cada um exigindo permissões de acesso individuais. Uma solução é definir identidades comuns e associar clientes MQTT individuais a uma das identidades comuns. Defina quantas identidades comuns forem necessárias para definir diferentes combinações de permissões. Outra solução é gravar seu próprio serviço de autorização que possa lidar com milhares de usuários com mais facilidade do que o sistema operacional.

É possível combinar clientes MQTT em identidades comuns de duas maneiras, usando o OAM:

1. Defina vários canais de telemetria, cada um com um ID de usuário diferente que o administrador aloca usando o WebSphere MQ Explorer. Clientes que se conectam usando números de porta TCP/IP diferentes são associados a diferentes canais de telemetria e recebem identidades diferentes.
2. Defina um único canal de telemetria, mas cada cliente deve selecionar um `Username` de um pequeno conjunto de IDs de usuário. O administrador configura o canal de telemetria para selecionar o cliente `Username` como sua identidade.

Nesta tarefa, a identidade do canal de telemetria é chamada de *`mqttUser`*, independentemente de como ele está definido. Se as coleções de clientes utilizam diferentes identidades, utilize vários do *`mqttUsers`*, um para cada coleção de clientes. Como a tarefa usa o OAM, cada *`mqttUser`* deve ser um ID de usuário.

Sobre esta tarefa

Nesta tarefa, você tem quatro padrões de controle de acesso que podem ser padronizados para requisitos específicos. Os padrões são diferentes em termos de granularidade de controle de acesso.

- “Nenhum Controle de Acesso” na página 128
- “Controle de Acesso de Alta Granularidade” na página 128
- “Controle de Acesso de Média Granularidade” na página 128
- “Controle de acesso de baixa granularidade” na página 128

O resultado dos modelos é designar *mqttUsers* conjuntos de permissões para publicar e assinar o WebSphere MQe receber publicações do WebSphere MQ..

Nenhum Controle de Acesso
Os clientes MQTT recebem autoridade administrativa do WebSphere MQ e podem executar qualquer ação em qualquer objeto.

Procedimento

1. Crie um ID do usuário *mqttUser* para agir como a identidade de todos os clientes do MQTT.
2. Inclua *mqttUser* no grupo `mqm` ; consulte Incluindo um usuário em um grupo no Janelas ..ou Incluindo um usuário em um grupo no Linux ..

Controle de Acesso de Alta Granularidade
Os clientes MQTT têm autoridade para publicar e assinar e para enviar mensagens para clientes MQTT. Eles não têm autoridade para executar outras ações ou para acessar outros objetos.

Procedimento

1. Crie um ID do usuário *mqttUser* para agir como a identidade de todos os clientes do MQTT.
2. Autorize o *mqttUser* a publicar e assinar todos os tópicos e enviar publicações para 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
```

Controle de Acesso de Média Granularidade
Os clientes MQTT são divididos em grupos diferentes para publicar e assinar diferentes conjuntos de tópicos e para enviar mensagens para os clientes MQTT

Procedimento

1. Crie IDs de usuários múltiplos, *mqttUsers*e vários tópicos administrativos na árvore de tópicos de publicação/assinatura.
2. Autorize diferente *mqttUsers* para diferentes tópicos.

```
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. Crie um grupo *mqtt* e inclua todos os *mqttUsers* no grupo.
4. Autorize o *mqtt* para enviar tópicos para os clientes MQTT

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Controle de acesso de baixa granularidade
Os clientes MQTT são incorporados em um sistema existente de controle de acesso, que autoriza grupos a executar ações em objetos.

Sobre esta tarefa

Um ID do usuário é designado a um ou mais grupos de sistemas operacionais, dependendo das autorizações que requer. Se os aplicativos do WebSphere MQ estiverem publicando e assinando o mesmo espaço de tópico que os clientes MQTT, use este modelo Os grupos são referidos como Publish*X*, Subscribe*Y*e `mqtt`

Publish*X*
Membros de grupos Publish*X* podem publicar em *topicX*.

Subscribe*Y*
Membros de grupos do Subscribe*Y* podem assinar *topicY*.

mqtt
Membros do grupo *mqtt* podem enviar publicações para clientes MQTT.

Procedimento

1. Crie diversos grupos Publish*X* e Subscribe*Y* que são alocados para diversos tópicos administrativos na árvore de tópicos de publicação / assinatura.
2. Crie um grupo `mqtt`.
3. Crie IDs de usuários múltiplos, *mqttUserse* inclua os usuários a qualquer um dos grupos, dependendo do que eles estão autorizados a fazer.
4. Autorize diferentes grupos Publish*X* e Subscribe*X* para diferentes tópicos e autorize o grupo *mqtt* para enviar mensagens para 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
```

Autenticação de Cliente MQTT Usando uma Senha

Autentique `Username` usando a senha do cliente. É possível autenticar o cliente usando uma identidade diferente da usada para autorizar o cliente para publicar e assinar tópicos.

O serviço de telemetria (MQXR) usa JAAS para autenticar o cliente `Username`. JAAS usa o `Password` fornecido pelo cliente MQTT.

O administrador do WebSphere MQ decide se deve autenticar `Username`, ou não autenticar nada, configurando o canal MQTT ao qual um cliente se conecta. Clientes podem ser designados a diferentes canais, e cada canal pode ser configurado para autenticar seus clientes de maneiras diferentes. Usando JAAS, é possível configurar quais métodos devem autenticar o cliente e quais podem opcionalmente autenticar o cliente.

A opção de identidade para autenticação não afeta a escolha da identidade para autorização. É possível querer configurar uma identidade comum para autorização para comodidade administrativa, mas autenticar cada usuário para usar essa identidade. O procedimento a seguir descreve as etapas para autenticar usuários individuais para usarem uma identidade comum:

1. O administrador do WebSphere MQ configura a identidade do canal MQTT para qualquer nome, como `MQTTClientUser`, usando WebSphere MQ Explorer.
2. O administrador do WebSphere MQ autoriza o `MQTTClient` a publicar e se subscrever para qualquer tópico:

```
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. O desenvolvedor de aplicativos clientes MQTT cria um objeto `MqttConnectOptions` e configura `Username` e `Password` antes de conectar-se ao servidor
4. O desenvolvedor de segurança cria um `LoginModule` JAAS para autenticar `Username` com `Password` e incluí-lo no arquivo de configuração JAAS.

5. O administrador do WebSphere MQ configura o canal MQTT para autenticar o `UserName` do cliente usando JAAS.

Autenticação de Cliente MQTT Usando SSL

Conexões entre o cliente MQTT e o gerenciador de filas são sempre iniciadas pelo cliente MQTT. O cliente MQTT sempre o cliente SSL. A autenticação de cliente para o servidor e a autenticação de servidor para o cliente MQTT são opcionais.

Ao fornecer ao cliente um certificado digital assinado privado, é possível autenticar o cliente MQTT no IBM WebSphere MQ. O Administrador do IBM WebSphere MQ pode forçar os clientes MQTT a se autenticarem no gerenciador de filas usando SSL. Só é possível solicitar autenticação de cliente como parte da autenticação mútua.

Como uma alternativa para o uso de SSL, alguns tipos de Virtual Private Network (VPN), como IPsec, autenticam os terminais de uma conexão TCP/IP. A VPN criptografa cada pacote IP que flui pela rede. Após uma conexão VPN ser estabelecida, você estabeleceu uma rede confiável. É possível conectar clientes MQTT a canais de telemetria usando TCP/IP sobre a rede VPN.

A autenticação de cliente usando SSL acredita que o cliente tenha um segredo. O segredo é a chave privada do cliente no caso de um certificado autoassinado ou uma chave fornecida por uma autoridade de certificação. A chave é usada para assinar o certificado digital do cliente. Qualquer pessoa em posse de uma chave pública correspondente pode verificar o certificado digital. Certificados podem ser confiáveis ou, se estiverem em cadeia, rastreados de volta por uma cadeia de certificados para um certificado de raiz confiável. A verificação de cliente envia todos os certificados na cadeia de certificados fornecida pelo cliente para o servidor. O servidor verifica a cadeia de certificados até localizar um certificado de sua confiança. O certificado confiável é o certificado público gerado a partir de um certificado autoassinado, ou um certificado raiz normalmente emitido por uma autoridade de certificação. Como etapa final opcional, o certificado confiável pode ser comparado com uma lista de revogação de certificado de "produção".

O certificado confiável pode ter sido emitido por uma autoridade de certificação e já estar incluído no armazenamento de certificados JRE. Ele pode ser um certificado autoassinado ou qualquer certificado que tenha sido incluído no keystore de canal de telemetria como um certificado confiável.

Nota: O canal de telemetria tem um keystore/armazenamento confiável combinados que retêm chaves privadas para um ou mais canais de telemetria e quaisquer certificados públicos necessários para autenticar clientes. Como um canal SSL deve ter um keystore, e esse é o mesmo arquivo que do armazenamento confiável de canal, o armazenamento de certificados JRE nunca é usado como referência. A implicação é que se a autenticação de um cliente exigir um certificado raiz de CA, você deverá colocar o certificado raiz no keystore para o canal, mesmo se esse certificado raiz de CA já estiver no armazenamento de certificados JRE. O armazenamento de certificados JRE nunca é usado como referência.

Pense nas ameaças que a autenticação de cliente deverá contar e nas funções que o cliente e o servidor desempenham na contagem de ameaças. A autenticação de um certificado de cliente sozinha é insuficiente para evitar acesso não autorizado a um sistema. Se alguma outra pessoa tiver retido o dispositivo do cliente, o dispositivo do cliente não estará agindo necessariamente com a autoridade do portador do certificado. Nunca conte com apenas uma defesa contra ataques indesejados. Use pelo menos uma abordagem de autenticação de dois fatores e tenha um suplemento de certificado com conhecimento de informações particulares. Por exemplo, use JAAS e autentique o cliente usando uma senha emitida pelo servidor.

A principal ameaça para o certificado de cliente é cair nãos mãos erradas. O certificado fica retido em um keystore protegido por senha no cliente. Como ele é colocado no keystore? Como o cliente MQTT consegue a senha para o keystore? Até que ponto a proteção da senha é segura? Os dispositivos de telemetria costumam ser fáceis de remover e, por isso, podem ser hackeados em particular. O hardware de dispositivo deve ser à prova de violação? A distribuição e a proteção de certificados do lado do cliente são reconhecidas como difíceis; elas são chamadas de problema de gerenciamento de chaves.

Uma segunda ameaça é o mau uso do dispositivo para acessar servidores de maneiras indesejadas. Por exemplo, se o aplicativo MQTT estiver corrompido, talvez seja possível usar um ponto fraco da configuração do servidor usando a identidade do cliente autenticado.

Para autenticar um cliente MQTT usando SSL, configure o canal de telemetria e o cliente.

•
•

Configuração do canal de telemetria para autenticação de cliente MQTT usando o SSL

O administrador IBM WebSphere MQ configura canais de telemetria no servidor. Cada canal é configurado para aceitar uma conexão TCP/IP em um número de porta diferente. Canais SSL são configurados com acesso protegido por passphrase a arquivos-chave. Se um canal SSL for definido sem passphrase ou arquivo-chave, ele não aceitará conexões SSL.

Configure a propriedade `com.ibm.mq.MQTT.ClientAuth` de um canal de telemetria SSL para `REQUIRED` para forçar todos os clientes que se conectam nesse canal a fornecer prova de que eles verificaram certificados digitais. Os certificados de cliente são autenticados usando certificados de autoridades de certificação, levando a uma certificado de raiz confiável. Se o certificado de cliente for autoassinado ou for assinado por um certificado de uma autoridade de certificação, os certificados assinados publicamente do cliente ou da autoridade de certificação deverão ser armazenados de forma segura no servidor.

Coloque o certificado de cliente assinado publicamente ou o certificado da autoridade de certificação no keystore do canal de telemetria. No servidor, os certificados assinados publicamente são armazenados no mesmo arquivo-chave que os certificados assinados particularmente, e não em um armazenamento confiável separado.

O servidor verifica a assinatura de todos os certificados de cliente enviados usando todos os certificados públicos e conjuntos de criptografia que possui. O servidor verifica a ou ia de chaves. O gerenciador de filas pode ser configurado para testar o certificado com relação à lista de revogação de certificado. A propriedade da lista de nomes de revogação do gerenciador de filas é `SSLCRLNL`.

Se algum dos certificados enviado por um cliente for verificado por um certificado no keystore do servidor, o cliente será autenticado.

O administrador do WebSphere MQ pode configurar o mesmo canal de telemetria para usar JAAS para verificar o `UserName` ou `ClientIdentifier` do cliente com a `Senhado` cliente.

É possível usar o mesmo keystore para diversos canais de telemetria.

A verificação de pelo menos um certificado digital no keystore do cliente protegido por senha no dispositivo autentica o cliente para o servidor. O certificado digital é usado somente para autenticação pelo WebSphere MQ Ele não é usado para verificar o endereço TCP/IP do cliente ou configurar a identidade do cliente para autorização ou contabilidade. A identidade do cliente adotada pelo servidor é o `Nome do usuário` ou `ClientIdentifier` do cliente ou uma identidade criada pelo administrador do WebSphere MQ

Também é possível usar conjuntos de criptografia SSL para autenticação de cliente.Aqui está uma lista alfabética dos conjuntos de cifras SSL que são atualmente suportados:

- `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 Se você planeja usar as suites cifradas SHA-2 , consulte Requisitos do sistema para usar as suites cifradas SHA-2 com canais MQTT.

Conceitos relacionados

“Configuração do canal de telemetria para autenticação de canal usando SSL” na página 133
O administrador IBM WebSphere MQ configura canais de telemetria no servidor. Cada canal é configurado para aceitar uma conexão TCP/IP em um número de porta diferente. Canais SSL são configurados com acesso protegido por passphrase a arquivos-chave. Se um canal SSL for definido sem passphrase ou arquivo-chave, ele não aceitará conexões SSL.

CipherSpecs e CipherSuites

Referências relacionadas

DEFINE CHANNEL (MQTT)

ALTER CHANNEL (MQTT)

Autenticação de Canal de Telemetria Usando SSL

Conexões entre o cliente MQTT e o gerenciador de filas são sempre iniciadas pelo cliente MQTT. O cliente MQTT sempre o cliente SSL. A autenticação de cliente para o servidor e a autenticação de servidor para o cliente MQTT são opcionais.

O cliente sempre tenta autenticar o servidor, a menos que o cliente esteja configurado para usar um CipherSpec que suporte conexão anônima. Se a autenticação falhar, a conexão não será estabelecida.

Como uma alternativa para o uso de SSL, alguns tipos de Virtual Private Network (VPN), como IPsec, autenticam os terminais de uma conexão TCP/IP. A VPN criptografa cada pacote IP que flui pela rede. Após uma conexão VPN ser estabelecida, você estabeleceu uma rede confiável. É possível conectar clientes MQTT a canais de telemetria usando TCP/IP sobre a rede VPN.

A autenticação de servidor usando SSL autentica o servidor para o qual você está prestes a enviar informações confidenciais. O cliente executa as verificações correspondentes aos certificados enviados do servidor em relação aos certificados colocados em seu armazenamento confiável ou em seu armazenamento `cacerts` JRE.

O armazenamento de certificados JRE é um arquivo JKS `cacerts`. Ele está localizado em JRE `InstallPath\lib\security\`. Ele é instalado com a senha padrão `changeit`. É possível armazenar certificados de sua confiança no armazenamento de certificados JRE ou no armazenamento confiável do cliente. Não é possível usar os dois armazenamentos. Use o armazenamento confiável do cliente se você deseja manter os certificados públicos nos quais o cliente confia separados dos certificados que são usados por outros aplicativos Java. Use o armazenamento de certificados JRE se você deseja usar um armazenamento de certificados comum para todos os aplicativos Java em execução no cliente. Se você decidir usar o armazenamento de certificados JRE, revise os certificados que ele contém para se certificar de que confia neles.

É possível modificar a configuração JSSE fornecendo um provedor de confiança diferente. É possível customizar um provedor de confiança para executar diferentes verificações em um certificado. Em alguns ambientes OGSi que usaram o cliente MQTT, o ambiente fornece um provedor de confiança diferente.

Para autenticar o canal de telemetria usando SSL, configure o servidor e o cliente.

-
-

Configuração do canal de telemetria para autenticação de canal usando SSL

O administrador IBM WebSphere MQ configura canais de telemetria no servidor. Cada canal é configurado para aceitar uma conexão TCP/IP em um número de porta diferente. Canais SSL são configurados com acesso protegido por passphrase a arquivos-chave. Se um canal SSL for definido sem passphrase ou arquivo-chave, ele não aceitará conexões SSL.

Armazene o certificado digital do servidor, assinado com sua chave privada, no armazenamento de chaves que o canal de telemetria for utilizar no servidor. Armazene quaisquer certificados na ou ia de chaves no keystore, caso queira transmitir a ou ia de chaves para o cliente. Configure o canal de telemetria usando o WebSphere MQ Explorer para usar SSL. Forneça a ele o caminho para o keystore e o passphrase para acessar o keystore. Se você não configurar o número da porta TCP/IP do canal, o número da porta do canal de telemetria SSL será padronizado como 8883.

Também é possível usar conjuntos de criptografia SSL para autenticação de canal.Aqui está uma lista alfabética dos conjuntos de cifras SSL que são atualmente suportados:

- `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 Se você planeja usar as suites cifradas SHA-2 , consulte Requisitos do sistema para usar as suites cifradas SHA-2 com canais MQTT.

Conceitos relacionados

“Configuração do canal de telemetria para autenticação de cliente MQTT usando o SSL” na página 131

O administrador IBM WebSphere MQ configura canais de telemetria no servidor. Cada canal é configurado para aceitar uma conexão TCP/IP em um número de porta diferente. Canais SSL são configurados com acesso protegido por passphrase a arquivos-chave. Se um canal SSL for definido sem passphrase ou arquivo-chave, ele não aceitará conexões SSL.

CipherSpecs e CipherSuites

Referências relacionadas

DEFINE CHANNEL (MQTT)

ALTER CHANNEL (MQTT)

Privacidade de publicação em canais de telemetria

A privacidade de publicações MQTT enviadas em qualquer direção em canais de telemetria é protegida usando o SSL para criptografar transmissões através da conexão.

Os clientes MQTT que se conectam aos canais de telemetria usam o SSL para proteger a privacidade de publicações transmitidas no canal usando a criptografia de chave simétrica. Como os terminais não são autenticados, não é possível confiar somente na criptografia do canal. Combine a proteção da privacidade com a autenticação mútua ou de servidor.

Como uma alternativa para o uso de SSL, alguns tipos de Virtual Private Network (VPN), como IPsec, autenticam os terminais de uma conexão TCP/IP. A VPN criptografa cada pacote IP que flui pela rede. Após uma conexão VPN ser estabelecida, você estabeleceu uma rede confiável. É possível conectar clientes MQTT a canais de telemetria usando TCP/IP sobre a rede VPN.

Para uma configuração típica, que criptografa o canal e autentica o servidor, consulte “Autenticação de Canal de Telemetria Usando SSL” na página 133.

Criptografar conexões SSL sem autenticar o servidor expõe a conexão com ataques man-in-the-middle. Embora as informações que você troca estejam protegidas contra interceptação, você não sabe com quem está trocando-as. A menos que você controle a rede, você está exposto a alguém que intercepte suas transmissões de IP e se disfarce como o terminal.

É possível criar uma conexão SSL criptografada, sem autenticar o servidor, utilizando uma CipherSpec de troca de chave Diffie-Hellman anônimo que suporta SSL. O segredo principal, compartilhado entre o cliente e o servidor e utilizado para criptografar transmissões SSL é estabelecido sem a troca de um certificado do servidor assinado em particular.

Como as conexões anônimas são precárias, a maioria das implementações SSL não assumem como padrão o uso de CipherSpecs anônimo. Se uma solicitação de cliente para conexão SSL é aceita por um canal de telemetria, o canal deve ter um keystore protegido por uma passphrase. Por padrão, desde que as implementações de SSL não usem CipherSpecs anônimo, o keystore deve conter um certificado assinado em particular que o cliente possa autenticar.

Se você usar CipherSpecs anônimo, o armazenamento de chaves do servidor deve existir, mas ele não precisa conter quaisquer certificados assinados particularmente.

Outra maneira de estabelecer uma conexão criptografada é substituir o provedor de confiança no cliente por sua própria implementação. O provedor de confiança não autenticaria o certificado do servidor, mas a conexão seria criptografada.

Configuração de SSL dos Clientes MQTT e Canais de Telemetria

Os clientes MQTT e o serviço WebSphere MQ Telemetry (MQXR) usam Java Secure Socket Extension (JSSE) para conectar canais de telemetria usando SSL. O daemon de telemetria IBM WebSphere MQ para dispositivos não suporta SSL.

Configure o SSL para autenticar o canal de telemetria, o cliente MQTT e criptografar a transferência de mensagens entre clientes e o canal de telemetria.

Como uma alternativa para o uso de SSL, alguns tipos de Virtual Private Network (VPN), como IPsec, autenticam os terminais de uma conexão TCP/IP. A VPN criptografa cada pacote IP que flui pela rede.

Após uma conexão VPN ser estabelecida, você estabeleceu uma rede confiável. É possível conectar clientes MQTT a canais de telemetria usando TCP/IP sobre a rede VPN.

É possível configurar a conexão entre um cliente do MQTT Java e um canal de telemetria para usar o protocolo SSL sobre TCP/IP. O que será assegurado depende de como você configura SSL para usar JSSE. Começando com a configuração mais segura, é possível configurar três níveis diferentes de segurança:

1. Permita que apenas clientes MQTT confiáveis se conectem. Conecte um cliente MQTT apenas a um canal de telemetria confiável. Criptografe mensagens entre o cliente e o gerenciador de filas; consulte "Autenticação de Cliente MQTT Usando SSL" na página 130
2. Conecte um cliente MQTT apenas a um canal de telemetria confiável. Criptografe mensagens entre o cliente e o gerenciador de filas; consulte "Autenticação de Canal de Telemetria Usando SSL" na página 133.
3. Criptografe mensagens entre o cliente e o gerenciador de filas; consulte "Privacidade de publicação em canais de telemetria" na página 135.

Parâmetros de Configuração de JSSE

Modifique os parâmetros de JSSE para alterar a maneira como a conexão SSL é configurada. Os parâmetros de configuração de JSSE são organizados em três conjuntos:

1. IBM WebSphere MQ Canal de telemetria
2. Cliente MQTT Java
3. JRE

Configure os parâmetros de canal de telemetria usando IBM WebSphere MQ Explorer. Configure os parâmetros do MQTT Java Client no atributo `MqttConnectionOptions.SSLProperties` Modifique os parâmetros de segurança do JRE editando arquivos no diretório de segurança do JRE no cliente e no servidor.

IBM WebSphere MQ Canal de telemetria

Configure todos os parâmetros SSL do canal de telemetria usando WebSphere MQ Explorer.

ChannelName

`ChannelName` é um parâmetro necessário em todos os canais.

O nome do canal identifica o canal associado a um determinado número de porta. Nomeie canais para ajudá-lo a administrar conjuntos de clientes MQTT.

PortNumber

`PortNumber` é um parâmetro opcional em todos os canais. Ele é padronizado como 1883 para canais TCP e 8883 para canais SSL.

O número da porta TCP/IP associado a esse canal. Clientes MQTT são conectados a um canal especificando a porta definida para o canal. Se o canal tiver propriedades SSL, o cliente deverá se conectar usando o protocolo SSL; por exemplo:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

KeyFileName

`KeyFileName` é um parâmetro necessário para canais SSL. Ele deve ser omitido para canais TCP.

`KeyFileNome` é o caminho para o Java keystore contendo certificados digitais que você fornece. Use JKS, JCEKS ou PKCS12 como tipo de keystore no servidor.

Identifique o tipo de keystore usando uma das seguintes extensões de arquivo:

```
.jks
.jceks
.p12
```

`.pkcs12`

Um keystore com qualquer outra extensão de arquivo é assumido como um keystore JKS.

É possível combinar um tipo de keystore no servidor com outros tipos de keystore no cliente.

Coloque o certificado particular do servidor no keystore. O certificado é conhecido como o certificado do servidor. O certificado pode ser autoassinado ou fazer parte de uma sequência de certificados que é assinada por uma autoridade de assinatura.

Se você estiver usando uma sequência de certificados, coloque os certificados associados no keystore do servidor.

O certificado do servidor, e quaisquer certificados na ou de certificados, é enviado para os clientes autenticarem a identidade do servidor.

Se você tiver configurado `ClientAuth` como `Required`, o keystore deverá conter quaisquer certificados necessários para autenticar o cliente. O cliente envia um certificado autoassinado ou uma sequência de certificados e o cliente é autenticado pela primeira verificação desse material a um certificado no keystore. Usando uma sequência de certificados, um certificado pode verificar muitos clientes, mesmo se eles forem emitidos com diferentes certificados de cliente.

PassPhrase

`PassPhrase` é um parâmetro necessário para canais SSL. Ele deve ser omitido para canais TCP.

O passphrase é usado para proteger o keystore.

ClientAuth

`ClientAuth` é um parâmetro SSL opcional. Ele é padronizado para não autenticação de cliente. Ele deve ser omitido para canais TCP.

Configure `ClientAuth` se quiser que o serviço de telemetria (MQXR) autentique o cliente antes de permitir que o cliente se conecte ao canal de telemetria.

Se você configurar `ClientAuth`, o cliente deverá se conectar ao servidor usando SSL e autenticar o servidor. Em resposta à configuração de `ClientAuth`, o cliente envia seu certificado digital para o servidor e quaisquer outros certificados em seu keystore. Seu certificado digital é conhecido como certificado de cliente. Esses certificados são autenticados com relação aqueles retidos no keystore do canal e no armazenamento `cacerts` do JRE.

CipherSuite

`CipherSuite` é um parâmetro SSL opcional. Ele é padronizado para tentar todos os CipherSpecs ativados. Ele deve ser omitido para canais TCP.

Se quiser usar um determinado CipherSpec, configure `CipherSuite` para o nome do CipherSpec que deve ser usado para estabelecer a conexão SSL.

O serviço de telemetria e o cliente MQTT negociam um CipherSpec comum de todos os CipherSpecs ativados em cada extremidade. Se um CipherSpec específico for especificado em ambas as extremidades da conexão, ele deverá corresponder ao CipherSpec na outra extremidade.

Instale cifras adicionais incluindo provedores adicionais no JSSE.

Federal Information Processing Standards (FIPS)

FIPS é uma configuração opcional. Por padrão, ela não é configurada.

No painel de propriedades do gerenciador de filas ou usando o comando **runmqsc**, configure SSLFIPS. SSLFIPS especifica se apenas algoritmos certificados por FIPS devem ser usados.

Lista de nomes de revogação

A lista de nomes de revogação é uma configuração opcional. Por padrão, ela não é configurada.

No painel de propriedades do gerenciador de filas ou usando o comando **runmqsc**, configure SSLCRLNL. SSLCRLNL especifica uma lista de nomes de objetos de informações sobre autenticação que são usados para fornecer locais de revogação de certificado.

Nenhum outro parâmetro de gerenciador de filas que configura propriedades de SSL é usado.

cliente MQTT Java

Configure as propriedades SSL para o cliente Java em `MqttConnectionOptions.SSLProperties`; por exemplo:

```
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);
```

Os nomes e valores das propriedades específicas são descritos na classe `MqttConnectOptions`.Para obter links para a documentação da API do cliente para as bibliotecas do cliente MQTT, consulte Referência de programação do cliente MQTT..

Protocolo

`Protocolo` é opcional.

O protocolo é selecionado na negociação com o servidor de telemetria. Se você requerer um protocolo específico, é possível selecionar um. Se o servidor de telemetria não suportar o protocolo, a conexão falhará.

ContextProvider

`ContextProvider` é opcional.

KeyStore

`KeyStore` é opcional. Configure-o se `ClientAuth` estiver configurado no servidor para forçar a autenticação do cliente.

Coloque o certificado digital do cliente, assinado com o uso de sua chave privada, no keystore. Especifique a senha e o caminho do keystore. O tipo e o provedor são opcionais. JKS é o tipo padrão e IBMJCE é o provedor padrão.

Especifique um provedor de keystore diferente para fazer referência a uma classe que inclui um novo provedor de keystore. Passe o nome do algoritmo usado pelo provedor de keystore para instanciar `KeyManagerFactory` configurando o nome do gerenciador de chave.

TrustStore

`TrustStore` é opcional. É possível colocar todos os certificados nos quais você confia no armazenamento `cacerts` do JRE.

Configure o armazenamento confiável se quiser ter um armazenamento confiável diferente para o cliente. É possível não configurar o armazenamento confiável se o servidor estiver usando um certificado emitido por uma CA conhecida que já tem seu certificado raiz armazenado em `cacerts`.

Inclua o certificado assinado publicamente do servidor ou o certificado raiz do armazenamento confiável e especifique o caminho e a senha do armazenamento confiável. JKS é o tipo padrão e IBMJCE é o provedor padrão.

Especifique um provedor de armazenamento confiável diferente para fazer referência a uma classe que inclui um novo provedor de armazenamento confiável. Passe o nome do algoritmo usado pelo provedor de armazenamento confiável para instanciar `TrustManagerFactory` configurando o nome do gerenciador de confiança.

JRE

Outros aspectos da segurança Java que afetam o comportamento do SSL no cliente e no servidor são configurados no JRE. Os arquivos de configuração no Windows estão no *Java Installation*

Directory\jre\lib\security Se você estiver usando o JRE fornecido com o IBM WebSphere MQ, o caminho será igual ao mostrado na seguinte tabela:

Tabela 8. Caminhos de Arquivos por Plataforma para Arquivos de Configuração SSL do JRE	
Plataforma	**Caminho de arquivo..**
Windows	*WMQ Installation Directory*\java\jre\lib\security
Outras plataformas UNIX and Linux	*WMQ Installation Directory*/java/jre64/jre/lib/security

Autoridades de certificação conhecidas

O arquivo `cacerts` contém os certificados raiz de autoridades de certificação conhecidas. O `cacerts` é usado por padrão, a menos que você especifique o armazenamento confiável. Se usar o armazenamento `cacertse` não fornecer um armazenamento confiável, você deverá revisar e editar a lista de assinantes no `cacerts` para atender aos requisitos de segurança.

É possível abrir o `cacerts` usando o comando WebSphere MQ `strmqikm`.que executa o utilitário IBM Key Management Abra o `cacerts` como um arquivo JKS usando a senha `changeit`. Modifique a senha para assegurar o arquivo.

Configurando classes de segurança

Use o arquivo `java.security` para registrar provedores de segurança adicionais e outras propriedades de segurança padrão.

Permissões

Use o arquivo `java.policy` para modificar as permissões concedidas a recursos. `javaws.policy` concede permissões para `javaws.jar`

Grau de Intensidade da Criptografia

Alguns JREs são fornecidos com segurança de criptografia reduzida. Se você não puder importar as chaves para os keystores, a criptografia de força reduzida poderá ser a causa. Tente iniciar o **ikeyman** usando o comando **strmqikm** ou fazer o download de arquivos de jurisdição fortes, mas limitados, em IBM Developer Kits, Informações de Segurança.

Importante: Seu país de origem pode ter restrições quanto à importação, à posse, ao uso ou à reexportação para outro país de um software de criptografia. Antes de fazer download ou usar os arquivos de políticas irrestritas, você deve verificar as leis do seu país. Verifique seus regulamentos e suas políticas com relação à importação, à posse, ao uso e à reexportação do software de criptografia para determinar se essas ações são permitidas.

Modificando o Provedor de Confiança para Permitir que o Cliente se Conecte a qualquer Servidor

O exemplo ilustra como incluir um provedor de confiança e fazer referência a ele a partir do código do cliente MQTT. O exemplo não faz autenticação do cliente ou servidor. A conexão SSL resultante é criptografada sem ser autenticada.

O fragmento de código em Figura 25 na página 139 configura o provedor de confiança e o gerenciador de confiança `AcceptAllProviders` para o 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 do 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

Configuração JAAS do Canal de Telemetria

Configure JAAS para autenticar o `Username` enviado pelo cliente.

O administrador do WebSphere MQ configura quais canais do MQTT requerem autenticação de cliente usando JAAS Especifique o nome de uma configuração JAAS para cada canal que deve executar a autenticação JAAS. Todos os canais podem usar a mesma configuração JAAS ou podem usar diferentes configurações JAAS. As configurações são definidas no *WMQData directory*`\qmgrs\`*qMgrName*`\mqxr\jaas.config`

O arquivo `jaas.config` é organizado pelo nome da configuração JAAS. Sob cada nome de configuração há uma lista de configurações de login; consulte Figura 29 na página 141.

O JAAS fornece quatro módulos de login padrão. Os módulos de Login padrão NT e UNIX são de valor limitado.

JndiLoginModule
Autentica com relação a um serviço de diretório configurado em JNDI (Java Naming and Directory Interface).

Krb5LoginModule
Autentica usando protocolos Kerberos.

NTLoginModule
Autentica usando informações de segurança de NT para o usuário atual.

UnixLoginModule
Autentica usando as informações de segurança UNIX para o usuário atual.

O problema com o uso de `NTLoginModule` ou `UnixLoginModule` é que o serviço de telemetria (MQXR) é executado com a identidade `mqm` e não com a identidade de canal do MQTT. `mqm` é a identidade passada para `NTLoginModule` ou `UnixLoginModule` para autenticação e não a identidade do cliente.

Para superar esse problema, grave seu próprio módulo de login ou use os outros módulos de login padrão. Uma amostra `JAASLoginModule.java` é fornecida com WebSphere MQ Telemetry. Ele é uma implementação da interface `javax.security.auth.spi.LoginModule`. Use-o para desenvolver seu próprio método de autenticação.

Quaisquer novas classes LoginModule que você fornecer deverão estar no caminho da classe do serviço de telemetria (MQXR). Não coloque suas classes nos diretórios do WebSphere MQ que estão no caminho da classe Crie seus próprios diretórios e defina o caminho da classe inteiro para o serviço de telemetria (MQXR).

É possível aumentar o caminho da classe usado pelo serviço de telemetria (MQXR) configurando o caminho da classe no arquivo `service.env`. `CLASSPATH` deve ser capitalizado e a instrução do caminho da classe só pode conter literais. Não é possível usar variáveis em CLASSPATH; por exemplo, `CLASSPATH=%CLASSPATH%` está incorreto. O serviço de telemetria (MQXR) configura seu próprio caminho de classe. O CLASSPATH definido em `service.env` é incluído nele.

O serviço de telemetria (MQXR) fornece dois retornos de chamada que retornam `Username` e o `Password` para um cliente conectado ao canal do MQTT. O `Nome do Usuário` e a `Senha` são configurados no objeto `MqttConnectOptions` Consulte Figura 30 na página 142 para obter um exemplo de como acessar `Username` e `Password`.

Examples

Exemplo de um Arquivo de Configuração JAAS com uma Configuração Nomeada, `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. Arquivo `jaas.config` de Amostra

Um exemplo de módulo de login JAAS codificado para receber o `Username` e o `Password` fornecidos por um 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 Amostra

IBM WebSphere MQ Daemon de telemetria para conceitos de dispositivos

O IBM WebSphere MQ Daemon de telemetria para dispositivos é um aplicativo cliente MQTT V3 avançado Use-o para armazenamento e encaminhamento de mensagens de outros clientes MQTT. Ele se conecta ao IBM WebSphere MQ como um cliente MQTT, mas também é possível conectar outros clientes MQTT a ele..

O daemon é um broker de publicação / assinatura Clientes MQTT V3 se conectam a ele para publicar e assinar tópicos, usando sequências de tópicos para publicar e filtros de tópicos para assinar. A sequência de tópicos é hierárquica com níveis de tópicos divididos por /. Os filtros de tópicos são sequências de tópicos que pode incluir curingas + de nível único e um curinga # de vários níveis como a última parte da sequência de tópicos.

Nota: Os curingas no daemon seguem as regras mais restritivas do WebSphere Message Broker, v6 IBM WebSphere MQ é diferente.. Ele suporta vários curingas de vários níveis; curinga pode representar qualquer número de níveis da hierarquia, em qualquer lugar na sequência de tópicos.

Vários clientes MQTT v3 se conectam ao daemon usando uma porta do listener. A porta do listener padrão é modificável. É possível definir várias portas listener e alocar namespaces diferentes para elas, consulte “Daemon do WebSphere MQ Telemetry para portas do listener de dispositivos” na página 150. O próprio daemon é um cliente MQTT v3. Configure uma conexão de ponte de daemon para conectar o daemon à porta do listener de outro daemon ou a um serviço de Telemetry (MQXR) do WebSphere MQ .

É possível configurar várias pontes do daemon de telemetria do WebSphere MQ para dispositivos. Use as pontes para conectar juntamente a uma rede de daemons que pode trocar publicações.

Cada ponte pode publicar e assinar tópicos em seu daemon local. Também é possível publicar e assinar tópicos em outro daemon, um broker de publicação/assinatura do WebSphere MQ ou qualquer outro broker MQTT v3 ao qual está conectado. Usando um filtro de tópicos, é possível selecionar as publicações para serem propagadas de um broker para outro. É possível propagar as publicações em qualquer direção. É possível propagar publicações do daemon local para cada um de seus brokers remotos conectados ou de qualquer um dos brokers conectados para o daemon local; consulte “IBM WebSphere MQ daemon de telemetria para pontes de dispositivos” na página 143.

IBM WebSphere MQ daemon de telemetria para pontes de dispositivos

Um IBM WebSphere MQ daemon de telemetria para dispositivos ponte conecta dois brokers de publicação / assinatura usando o protocolo MQTT v3 . A ponte propaga publicações de um broker para outro, em qualquer direção. Em uma extremidade há um daemon de telemetria do WebSphere MQ para a conexão de ponte de dispositivos e na outra pode ser um gerenciador de filas ou outro daemon Um gerenciador de filas é conectado à conexão de ponte usando um canal de telemetria. Um daemon está conectado à conexão de ponte usando um listener do daemon.

IBM WebSphere MQ O Daemon de telemetria para dispositivos suporta uma ou mais conexões simultâneas com outros brokers As conexões do daemon são chamadas de pontes e são definidas pelas entradas de conexão no arquivo de configuração do daemon. As conexões com o IBM WebSphere MQ são feitas usando canais de telemetria do IBM WebSphere MQ , conforme mostrado na figura a seguir:

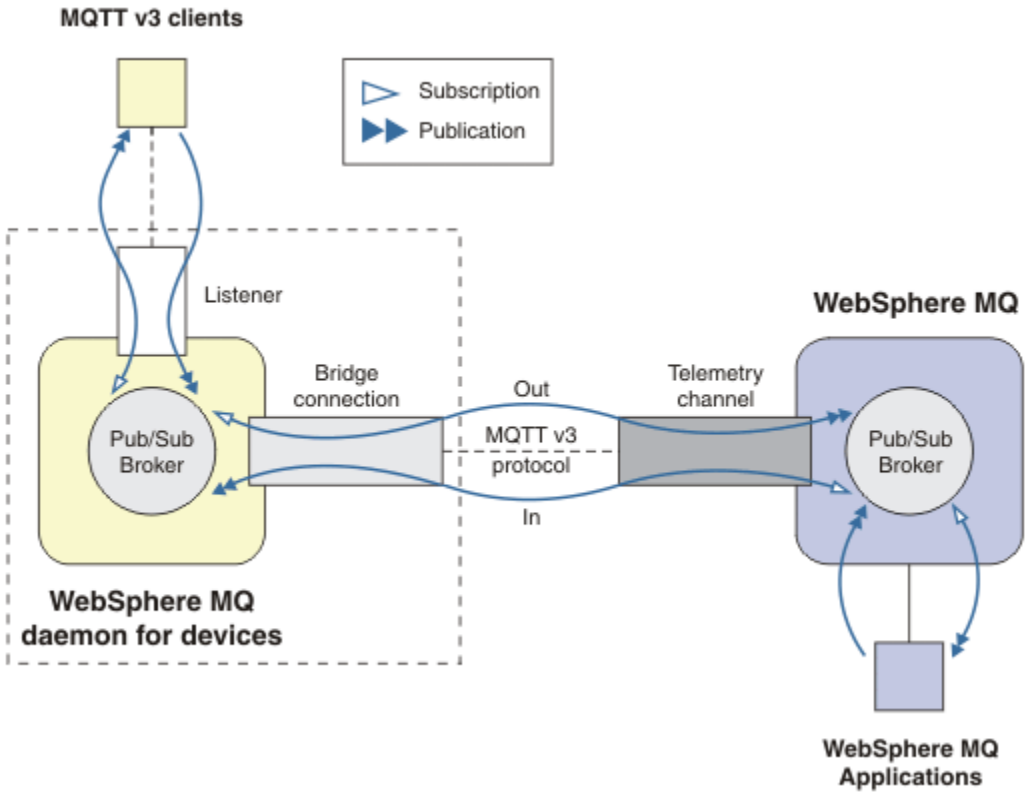


Figura 31. Conectando o IBM WebSphere MQ Telemetry daemon for devices a IBM WebSphere MQ

Uma ponte conecta o daemon em outro broker como um cliente MQTT v3. Os parâmetros bridge espelham os atributos de um cliente MQTT v3 .

Uma ponte é mais de uma conexão. Ela atua como um agente de publicação e assinatura situado entre dois brokers de publicação/assinatura. O broker local é o IBM WebSphere MQ daemon de Telemetria para dispositivos e o broker remoto é qualquer broker de publicação / assinatura que suporte o protocolo MQTT v3 Geralmente o broker remoto é outro daemon ou IBM WebSphere MQ.

A tarefa da ponte é propagar publicações entre os dois brokers. A ponte é bidirecional. Ela propaga publicações em qualquer direção. O Figura 31 na página 143 ilustra a maneira como a ponte conecta o daemon do IBM WebSphere MQ Telemetry para dispositivos ao IBM WebSphere MQ O “Configurações de tópico de exemplo para a ponte” na página 144 usa exemplos para ilustrar como usar o parâmetro topic para configurar a ponte.

As setas Entrada e Saída em Figura 31 na página 143 indicam a bidirecionalidade da ponte Em uma extremidade da seta, uma assinatura é criada. As publicações que correspondem à assinatura são

publicadas para o broker na extremidade oposta da seta. A seta é rotulada de acordo com o fluxo de publicações. Fluxo de publicações In para o daemon e Out a partir do daemon. A importância dos rótulos é que eles são usados na sintaxe de comando. Lembre-se de que In e Out se referem ao lugar em que o fluxo de publicações está e não ao lugar em que a assinatura é enviada.

Outros clientes, aplicativos ou brokers podem estar conectados ao IBM WebSphere MQ ou ao daemon de telemetria do WebSphere MQ para dispositivos Eles publicam e assinam tópicos no broker ao qual eles estão conectados. Se o broker for IBM WebSphere MQ, os tópicos poderão ser armazenados em cluster ou distribuídos e não serão explicitamente definidos no gerenciador de filas locais.

Usos de pontes

Conectar daemons juntos usando conexões de ponte e listeners. Conecte daemons e gerenciadores de filas juntos usando conexões de ponte e canais de telemetria. Ao conectar vários brokers juntos, será possível criar loops. Cuidado: publicações podem circular incessantemente ao redor de um loop de brokers, não detectado.

Algumas das razões para usar daemons vinculados ao IBM WebSphere MQ são as seguintes:

Reduza o número de conexões do cliente MQTT para WebSphere MQ
Usando uma hierarquia de daemons é possível conectar muitos clientes ao WebSphere MQ; mais clientes do que o número que um único gerenciador de filas pode conectar de uma vez.

Armazenar e encaminhar mensagens entre clientes MQTT e WebSphere MQ
Você pode usar armazenamento e encaminhamento para evitar manter conexões contínuas entre clientes e IBM WebSphere MQ, se os clientes não tiverem seu próprio armazenamento. É possível usar diversos tipos de conexões entre o cliente MQTT e o WebSphere MQ. Consulte Conceitos e cenários de telemetria para monitoramento e controle

Filtrar as publicações trocadas entre clientes MQTT e WebSphere MQ
Geralmente, as publicações se dividem em mensagens processadas localmente e mensagens que envolvem outros aplicativos. As publicações locais podem incluir fluxos de controle entre sensores e atuadores e as publicações remotas incluem solicitações para leituras, status e comandos de configuração.

Mude os espaços de tópicos de publicações
Evite sequências de tópicos a partir de clientes conectados às portas de listener diferentes de uma colisão entre si. O exemplo usa o daemon para rotular as leituras do medidor provenientes de diferentes construções; consulte Separando os espaços de tópico de diferentes grupos de clientes

Configurações de tópico de exemplo para a ponte

Publique tudo para o broker remoto - usando padrões

A direção padrão é chamada out e a ponte publica os tópicos para o broker remoto. O parâmetro topic controla quais tópicos são propagados usando filtros de tópicos.

A ponte usa o parâmetro topic no Figura 32 na página 144 para assinar tudo publicado para o daemon local pelos clientes MQTT ou por outros brokers. A ponte publica os tópicos para o broker remoto conectado pela ponte.

```
connection Daemon1
topic #
```

Figura 32. Publique tudo para o broker remoto

Publique tudo para o broker remoto - explícito

A configuração topic no fragmento de código a seguir fornece o mesmo resultado que é usado pelos padrões. A única diferença é que o parâmetro **direction** é explícito. Use a direção `out` para assinar o broker local, o daemon e publicar no broker remoto. As publicações criadas no daemon local que a ponte foi assinada, são publicadas no broker remoto.

```
connection Daemon1
topic # out
```

Figura 33. Publique tudo para o broker remoto - explícito

Publique tudo para o broker local

Em vez de usar a direção `out`, é possível configurar a direção oposta, `in`. O fragmento de código a seguir configura a ponte para assinar tudo publicado no broker remoto conectado pela ponte. A ponte publica os tópicos para o broker local, o daemon.

```
connection Daemon1
topic # in
```

Figura 34. Publique tudo para o broker local

Publique tudo a partir do tópico de exportação no broker local para o tópico de importação no broker remoto

Use dois parâmetros do tópicos adicionais, **`local_prefix`** e **`remote_prefix`**, para modificar o filtro de tópicos, `#` nos exemplos anteriores. Um parâmetro é usado para modificar o filtro de tópicos usado na assinatura e o outro parâmetro é usado para modificar o tópico ao qual a publicação é publicada. O efeito é para substituir o início da sequência de tópicos usada em um broker com outra sequência de tópicos no outro broker.

Dependendo da direção do comando de tópicos, o significado de **`local_prefix`** e **`remote_prefix`** será revertido. Se a direção for `out`, o padrão, **`local_prefix`** será usado como parte da assinatura do tópico e **`remote_prefix`** substituirá a parte da sequência de tópicos **`local_prefix`** na publicação remota. Se a direção for `in`, **`remote_prefix`** se tornará parte da assinatura remota e **`local_prefix`** substituirá a parte da sequência de tópicos **`remote_prefix`**.

A primeira parte de uma sequência de tópicos é muitas vezes considerada como a definição de um espaço de tópico. Use os parâmetros adicionais para mudar o espaço de tópico ao qual um tópico é publicado. Você pode fazer isto para evitar que o tópico que está sendo propagado colida com outro ao qual o tópico está no broker de destino ou para remover uma sequência de tópicos do ponto de montagem.

Como um exemplo, no fragmento de código a seguir, todas as publicações para a sequência de tópicos `export/#` no daemon são republicadas para `import/#` no broker remoto.

```
topic # out export/ import/
```

Figura 35. Publique tudo a partir do tópico de exportação no broker local para o tópico de importação no broker remoto

Publique tudo para o tópico de importação no broker local do tópico de exportação no broker remoto

O fragmento de código a seguir mostra a configuração revertida; a ponte assina tudo que foi publicado com a sequência de tópicos `export/#` no broker remoto e o publica para `import/#` no broker local.

```
connection Daemon1
topic # in import/ export/
```

Figura 36. Publique tudo para o tópico de importação no broker local do tópico de exportação no broker remoto

Publique tudo a partir do ponto de montagem 1884/ para o broker remoto com a sequência de tópicos original

No fragmento de código a seguir, a ponte assina tudo que foi publicado pelos clientes conectados ao ponto de montagem 1884/ no daemon local. A ponte publica tudo que foi publicado para o ponto de montagem no broker remoto. A sequência do ponto de montagem 1884/ é removida a partir dos tópicos publicados no broker remoto. O *local_prefix* é o mesmo que a sequência do ponto de montagem 1884/ e o *remote_prefix* é uma sequência em branco.

```
listener 1884
mount_point 1884/
connection Daemon1
topic # out 1884/ ""
```

Figura 37. Publique tudo a partir do ponto de montagem 1884/ para o broker remoto com as sequências de tópicos originais.

Separando os espaços de tópicos de clientes diferentes conectados a diferentes daemons

Um aplicativo é gravado para medidores de energia elétrica para publicar leituras do medidor para um prédio. As leituras são publicados usando os clientes MQTT em um daemon hospedado na mesma construção. O tópico selecionado para as publicações é `power`. O mesmo aplicativo é implementado em um número de construções em um complexo. Para o site de monitoramento e o armazenamento de dados, as leituras a partir de todas as construção são agregadas usando conexões de ponte. As conexões vinculam os daemons de construção ao WebSphere MQ em um local central..

Os aplicativos clientes em cada construção são idênticos, mas os dados devem ser diferenciados por construção Cada leitura tem um tópico `power` e deve ser prefixado com o número da construção para distingui-la... A ponte da primeira construção no complexo usa o prefixo `meters/building01/`, a partir da construção dois, o prefixo é `meters/building02/`. As leituras a partir de outras construções seguem o mesmo padrão. O WebSphere MQ recebe as leituras com tópicos como `meters/building01/power`

O exemplo é inventado; na prática, o espaço de tópico no qual o aplicativo publica é provavelmente configurável.

O arquivo de configuração para cada daemon tem uma instrução de tópico que segue o padrão no fragmento de código a seguir:

```
connection Daemon1
topic power out "" meters/building01/
```

Figura 38. Separe os espaços de tópicos de clientes conectados a diferentes daemons

Especifique uma sequência vazia como um item temporário para o parâmetro local_prefix não usado.

Separe os espaços de tópicos de clientes conectados ao mesmo daemon

Suponha que um daemon único é usado para conectar todos os medidores de energia. Supondo que no aplicativo pode ser configurado para se conectar a portas diferentes, você pode distinguir as construções, conectando os medidores a partir de construções diferentes a diferentes portas de listener, como no fragmento de código a seguir. Novamente, o exemplo é complicado; ele ilustra como os pontos de montagem podem ser usados.

```
listener 1884
mount_point meters/building01/
listener 1885
mount_point meters/building02/
connection Daemon1
topic meters/+/power out
```

Figura 39. Separe os espaços de tópicos de clientes conectados ao mesmo daemon

Remapear diferentes tópicos para publicações fluem em ambas as direções

Na configuração no fragmento de código a seguir, a ponte assina o único tópico b no broker remoto e encaminha publicações sobre b ao daemon local, mudando o tópico para a. A ponte também assina o único tópico x no broker local e encaminha publicações sobre x para o broker remoto, mudando o tópico para y.

```
connection Daemon1
topic "" in a b
topic "" out x y
```

Figura 40. Remapear diferentes tópicos para publicações fluem em ambas as direções

Um ponto importante sobre este exemplo é que diferentes tópicos são assinados e publicados em ambos os brokers. Os espaços de tópicos em ambos os brokers são desconectados.

Remapear os mesmos tópicos para publicações fluem em ambas as direções (em loop)

Diferente do exemplo anterior, a configuração no Figura 41 na página 147, em geral, resulta em um loop. Na instrução de tópico `topic "" in a b`, a ponte assina b remotamente e publica a localmente. Na outra instrução de tópico, a ponte assina a localmente e publica b remotamente. A mesma configuração pode ser gravada como mostrado em Figura 42 na página 147.

O resultado geral é que, se um cliente publicar no b remotamente, a publicação será transferida para o daemon local como uma publicação no tópico a No entanto, ao ser publicado pela ponte para o daemon local no tópico a, a publicação corresponde à assinatura feita pela ponte para o tópico local a. A assinatura é `topic "" out a b`.. Como resultado, a publicação é transferida de volta para o broker remoto como uma publicação no tópico b A ponte agora está inscrita no tópico remoto be o ciclo começa novamente.

Alguns brokers implementam a detecção de loop para evitar que o loop aconteça. Mas o mecanismo de detecção do loop deverá funcionar quando diferentes tipos de brokers forem ligados. A detecção de loop não funcionará se o WebSphere MQ estiver vinculado ao daemon do WebSphere MQ Telemetry para dispositivos Ele funcionará se dois daemon de Telemetria do IBM WebSphere MQ para dispositivos forem vinculados juntos. Por padrão, a detecção de loop está ativada. Consulte try_private

```
connection Daemon1
topic "" in a b
topic "" out a b
```

Figura 41. !Remapear os mesmos tópicos para publicações que fluem nas duas direções

```
connection Daemon1
topic "" both a b
```

Figura 42. !Remapeie os mesmos tópicos para publicações fluindo nas duas direções, usando `both`.

A configuração no Figura 40 na página 147 é a mesma que Figura 41 na página 147.

Disponibilidade de conexões de ponte IBM WebSphere MQ Telemetry daemon for devices

Configure diversos endereços de conexão de ponte IBM WebSphere MQ Telemetry daemon for devices para se conectar ao primeiro broker remoto disponível. Se o broker for um gerenciador de filas com várias instâncias, forneça ambos os endereços TCP/IP. Configure uma conexão primária para se conectar ou se reconectar ao servidor principal, quando ele estiver disponível.

O parâmetro de ponte de conexão, addresses é uma lista de endereços de soquete TCP/IP. A ponte tenta se conectar ao endereço de cada vez, até que ele faça uma conexão bem-sucedida. Os parâmetros de conexão round_robin e start_type controlam como os endereços serão usados após uma conexão bem-sucedida ser feita.

Se start_type for auto, manual ou lazy então, se a conexão falhar, a ponte tentará se reconectar. Ele usa cada endereço, por vez, com cerca de 20 segundos de atraso entre cada tentativa de conexão. Se `start_type` for once, então, se a conexão falhar, a ponte não tentará se reconectar automaticamente.

Se round_robin for true, as tentativas de conexão de ponte começarão com o primeiro endereço na lista e tentará cada endereço na lista em ordem. Ele iniciará no primeiro endereço novamente, quando a lista estiver esgotada. Se houver apenas um endereço na lista, ele tentará novamente a cada 20 segundos.

Se round_robin for false, o primeiro endereço na lista, que é chamado de servidor principal, será dado preferência. Se a primeira tentativa de se conectar ao servidor principal falhar, a ponte continuará a tentativa de se reconectar ao servidor principal em segundo plano. Ao mesmo tempo, a ponte tenta a conexão usando outros endereços na lista. Quando o segundo plano tentar se conectar ao servidor principal com sucesso, a ponte se desconectará da conexão atual e alternará para a conexão do servidor principal.

Se uma conexão for desconectada voluntariamente, por exemplo, emitindo um comando **`connection_stop`**, em seguida, se a conexão for reiniciada, ela tentará usar o mesmo endereço novamente. Se a conexão for desconectada devido a uma falha ao se conectar ou para o broker remoto eliminando a conexão, a ponte aguardará 20 segundos. Ela tentará se conectar ao próximo endereço na lista ou ao mesmo endereço, se houver apenas um endereço na lista.

Conectando-se a um gerenciador de filas com várias instâncias

Em uma configuração do gerenciador de filas com várias instâncias, o gerenciador de filas é executado em dois servidores diferentes com endereços IP diferentes. Geralmente, os canais de telemetria estão configurados sem um endereço IP específico. Eles são configurados apenas com um número de porta. Quando o canal de telemetria for iniciado, por padrão, ele selecionará o primeiro endereço de rede disponível no servidor local.

Configure o parâmetro addresses da conexão de ponte com os dois endereços IP usados pelo gerenciador de filas. Configure round_robin como true..

Se a instância ativa do gerenciador de filas falhar, o gerenciador de filas alternará para a instância em espera. O daemon detecta que a conexão com a instância ativa foi interrompida e tenta se reconectar à instância em espera. Ele usa o outro endereço IP na lista de endereços configurados para a conexão de ponte.

O gerenciador de filas ao qual a ponte se conecta ainda é o mesmo gerenciador de filas. O gerenciador de filas recupera seu próprio estado. Se cleansession for configurado como false, a sessão de conexão de ponte será restaurada para o mesmo estado que antes do failover A conexão será retomada após um atraso. As mensagens com a qualidade de serviço "pelo menos uma vez" ou "no máximo uma vez" não são perdidas e as assinaturas continuam a funcionar.

O tempo de reconexão depende do número de canais e clientes reiniciados ao iniciar a instância em espera e quantas mensagens estavam em andamento. A conexão de ponte poderá tentar se reconectar a ambos os endereços IP um número de vezes antes da conexão ser restabelecida.

Não configure um canal de telemetria do gerenciador de filas com várias instâncias com um endereço IP específico. O endereço IP é válido apenas em um servidor.

Se você estiver usando uma solução de alta disponibilidade alternativa, que gerencia o endereço IP, isso poderá estar correto para configurar um canal de telemetria com um endereço IP específico.

cleansession

Uma conexão de ponte é uma sessão de cliente MQTT v3. É possível controlar se uma conexão inicia uma nova sessão ou se restaura uma sessão existente. Se ela restaurar uma sessão existente, a conexão de ponte preservará as assinaturas e as publicações retidas da sessão anterior.

Não configure cleansession como false se endereços listar vários endereços IP e os endereços IP se conectarem a canais de telemetria hospedados por diferentes gerenciadores de filas, ou a diferentes daemons de telemetria O estado da sessão não é transferido entre os gerenciadores de filas ou os daemons. Ao tentar reiniciar uma sessão existente em um gerenciador de filas ou daemon diferente resultará em uma nova sessão sendo iniciada. Em mensagens indeterminadas são perdidas e as assinaturas podem não se comportar conforme esperado.

notificações

Um aplicativo pode acompanhar se a conexão de ponte está em execução usando as notificações. Uma notificação é uma publicação que tem o valor `1` conectado ou `0` desconectado. Ela é publicada para *topicString* definido pelo parâmetro `notification_topic`. O valor padrão de *topicString* é $SYS/`broker/connection/` *clientIdentifier*`/state`. O padrão *topicString* contém o prefixo $SYS Assine os tópicos iniciados com $SYS, definindo um filtro de tópicos iniciados com $SYS. O filtro de tópicos *#*, assina a tudo, não assina os tópicos iniciados com $SYS no daemon. Pense em $SYS como definindo um espaço de tópico do sistema especial diferente do espaço de tópico do aplicativo.

Notificações ativam IBM WebSphere MQ Telemetry daemon for devices para notificar clientes MQTT quando uma ponte está conectada ou desconectada.

keepalive_interval

O parâmetro de conexão de ponte keepalive_interval configura o intervalo entre a ponte enviando um ping do TCP/IP para o servidor remoto. O intervalo padrão é de 60 segundos. O ping impede que a sessão TCP/IP seja fechada pelo servidor remoto ou por um firewall, que detecta um período de inatividade na conexão.

CLIENTID

Uma conexão de ponte é uma sessão do cliente MQTT v3 e possui um `clientIdentifier` configurado pelo parâmetro de conexão de ponte clientid. Se você pretende que as reconexões retomem a uma sessão anterior configurando o parâmetro cleansession como false, o `clientIdentifier` usado em cada sessão deverá ser o mesmo. O valor padrão de clientid é *hostname.connectionName*, que permanece o mesmo.

Instalação, verificação, configuração e controle do daemon do WebSphere MQ Telemetry para dispositivos

A instalação, configuração e o controle do daemon são baseados em arquivos.

Instale o daemon copiando o Software Development Kit para o dispositivo no qual você executará o daemon .

Como exemplo, execute o utilitário do cliente MQTT e conecte-se ao daemon de telemetria do WebSphere MQ para dispositivos como o broker de publicação / assinação; consulte Publicar uma mensagem para um cliente MQTT v3 específico .

Configure o daemon criando um arquivo de configuração; consulte WebSphere MQ Telemetry daemon para o arquivo de configuração de dispositivos.

Controle um daemon em execução, criando comandos no arquivo, `amqtdd.upd`. A cada 5 segundos, o daemon lê o arquivo, executa os comandos e exclui o arquivo; consulte o WebSphere MQ Telemetry para o arquivo de comando de dispositivos

Daemon do WebSphere MQ Telemetry para portas do listener de dispositivos

Conecte os clientes MQTT V3 ao daemon WebSphere MQ Telemetry para dispositivos que usam portas listener. É possível qualificar uma porta do listener com um ponto de montagem e um número máximo de conexões.

Um porta do listener deve corresponder ao número da porta especificado no método `connect(serverURI)` do cliente MQTT de um cliente conectado a essa porta. Ela é padronizada no cliente e no daemon para 1883.

É possível mudar a porta padrão para o daemon, configurando a definição global `port` no arquivo de configuração do daemon. É possível configurar as portas específicas, incluindo uma definição `listener` para o arquivo de configuração do daemon.

Para cada porta do listener, diferente da porta padrão, é possível especificar um ponto de montagem para isolar os clientes. Os clientes conectados a uma porta com um ponto de montagem são isolados de outros clientes; consulte “Daemon do WebSphere MQ Telemetry para pontos de montagem de dispositivos” na página 150.

É possível limitar o número de clientes que pode se conectar a qualquer porta. Configure a definição global `max_connections` para limitar as conexões com a porta padrão ou qualificar cada porta do listener com `max_connections`.

exemplo

Um exemplo de um arquivo de configuração que muda a porta padrão de 1883 para 1880 e limita as conexões com a porta 1880 para 10000. As conexões à porta 1884 são limitadas a 1000. Os clientes conectados à porta 1884 são isolados dos clientes conectados a outras portas.

```
port 1880
max_connections 10000
listener 1884
mount_point 1884/
max_connections 1000
```

Daemon do WebSphere MQ Telemetry para pontos de montagem de dispositivos

É possível associar um ponto de montagem a uma porta listener usada pelos clientes MQTT para se conectar a um daemon do WebSphere MQ Telemetry para dispositivos. Um ponto de montagem isola as publicações e assinaturas trocadas pelos clientes MQTT usando uma porta do listener a partir de clientes MQTT conectados a uma porta do listener diferente.

Os clientes conectados a uma porta do listener com um ponto de montagem nunca podem trocar diretamente os tópicos com clientes conectados a quaisquer outras portas do listener. Os clientes conectados a uma porta do listener sem um ponto de montagem podem publicar ou assinar tópicos de qualquer cliente. Os clientes não estão cientes se eles estão conectados por meio de um ponto de montagem ou não; isso não faz diferença para as sequências de tópicos criadas pelos clientes.

Um ponto de montagem é uma sequência de texto prefixada na sequência de tópicos de publicações e assinaturas. Ela é prefixada para todas as sequências de tópicos criadas por clientes conectados a porta do listener com um ponto de montagem. A sequência de texto é removida de todas as sequências de tópicos enviadas para clientes conectados à porta do listener.

Se uma porta do listener não tiver nenhum ponto de montagem, as sequências de tópicos das publicações e assinaturas criadas e recebidas por clientes conectados à porta não serão alteradas.

Criar sequências de ponto de montagem com um final `/`. Dessa maneira o ponto de montagem é o tópico-pai da árvore de tópicos para o ponto de montagem.

exemplo

Um arquivo de configuração contém as seguintes portas listener:

```
listener 1883
mount_point 1883/
listener 1884 127.0.0.1
mount_point 1884/
listener 1885
```

Um cliente, conectado à porta 1883, cria uma assinatura para `MyTopic`. O daemon registra a assinatura, como `1883/MyTopic`. Outro cliente conectado à porta 1883 publica uma mensagem no tópico, `MyTopic`. O daemon muda a sequência de tópicos para `1883/MyTopic` e procura as assinaturas correspondentes. O assinante na porta 1883 recebe a publicação com a sequência de tópicos original `MyTopic`. O daemon removeu o prefixo do ponto de montagem da sequência de tópicos.

Outro cliente, conectado à porta 1884, também publica no tópico `MyTopic`. Desta vez, a daemon registra o tópico como `1884/MyTopic`. O assinante na porta 1883 não recebe a publicação, porque o ponto de montagem diferente resulta em uma assinatura com uma sequência de tópicos diferente.

Um cliente, conectado à porta 1885, publica no tópico, `1883/MyTopic`. O daemon não muda a sequência de tópicos. O assinante na porta 1883 recebe a publicação para `MyTopic`.

Daemon do WebSphere MQ Telemetry para qualidade de serviço de dispositivos, assinaturas duráveis e publicações retidas..

As configurações de qualidade de serviço se aplicam apenas a um daemon em execução. Se um daemon parar, seja em um modo controlado ou por causa de uma falha, o estado de mensagens em andamento será perdido. A entrega de uma mensagem pelo menos uma vez ou no máximo uma vez, não poderá ser garantida se o daemon parar. O daemon WebSphere MQ Telemetry para dispositivos suporta persistência limitada. Configure o parâmetro de configuração **`retained_persistence`** para salvar as publicações e assinaturas retidas quando o daemon for encerrado.

Diferente do WebSphere MQ, o daemon do WebSphere MQ Telemetry para dispositivos não registra dados persistentes. O estado de sessão, o estado da mensagem e as publicações retidas não serão salvos de forma transacional. Por padrão, o daemon descartará todos os dados, quando ele parar. É possível configurar uma opção para as publicações retidas e assinaturas de ponto de verificação periodicamente. O status da mensagem sempre será perdido quando o daemon parar. Todas as publicações não retidas são perdidas.

Configure a opção de configuração do daemon, `Retained_persistence` como `true`, para salvar as publicações retidas periodicamente para um arquivo. Quando o daemon for reiniciado, as publicações retidas que foram salvas automaticamente por último serão restabelecidas. Por padrão, as mensagens retidas criadas por clientes não serão restabelecidas quando o daemon for reiniciado.

Configure a opção de configuração do daemon, `Retained_persistence` como `true`, para salvar as assinaturas criadas em uma sessão persistente periodicamente para um arquivo. Se `Retained_persistence` for configurado como `true`, as assinaturas que os clientes criam em uma sessão com `CleanSession` configurado como `false`, uma "sessão persistente", serão restauradas. O daemon restaurará as assinaturas quando for reiniciado, o qual começará a receber publicações. O cliente receberá as publicações quando ele reiniciar com `CleanSession` como `false`. Por padrão, o estado de sessão do cliente não será salvo quando um daemon parar e as assinaturas não forem restauradas, mesmo se o cliente configurar `CleanSession` como `false`.

`Retained_persistence` é um mecanismo de salvamento automático. Ele não pode salvar as publicações ou assinaturas retidas mais recentes. É possível mudar a frequência com que publicações e assinaturas retidas são salvas. Configure o intervalo entre os salvamentos ou o número de mudanças entre os salvamentos, usando as opções de configuração `autosave_on_changes` e `autosave_interval`.

Configuração de exemplo para persistência configuração

```
# 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
```

Daemon do WebSphere MQ Telemetry para segurança de dispositivos

O daemon do WebSphere MQ Telemetry para dispositivos pode autenticar clientes que se conectam a ele, usar credenciais para conectar a outros brokers e controlar o acesso a tópicos. A segurança que o daemon fornece é limitada por ser construído usando o cliente C do WebSphere MQ Telemetry , que não fornece suporte SSL Consequentemente, as conexões para e do daemon não são criptografadas e não podem ser autenticadas usando certificados.

Por padrão, nenhuma segurança está ativada.

Autenticação de clientes

Os clientes MQTT podem configurar uma senha e um nome de usuário usando os métodos `MqttConnectOptions.setUserName` e `MqttConnectOptions.setPassword`.

Autenticar um cliente que se conecta ao daemon, verificando a senha e o nome do usuário fornecidos por um cliente com relação às entradas no arquivo de senha. Para ativar a autenticação, crie um arquivo de senha e configure o parâmetro `password_file` no arquivo de configuração do daemon; consulte password_file.

Configure o parâmetro `allow_anonymous` no arquivo de configuração do daemon para permitir que os clientes se conectem sem as senhas ou nomes dos usuários para se conectarem a um daemon que é a verificação de autenticação; consulte allow_anonymous. Se um cliente fornecer uma senha ou um nome do usuário, ele será sempre verificado com relação ao arquivo de senha, se o parâmetro `password_file` for configurado.

Configure o parâmetro `clientid_prefixes` no arquivo de configuração do daemon para limitar as conexões a clientes específicos. Os clientes devem ter `clientIdentifiers` iniciados com um dos prefixos listados no parâmetro `clientid_prefixes`; consulte clientid_prefixes.

Segurança de conexão de ponte

Cada daemon WebSphere MQ Telemetry para conexão de ponte de dispositivos é um cliente MQTT V3 . É possível configurar a senha e o nome do usuário para cada conexão de ponte como um parâmetro de conexão de ponte no arquivo de configuração do daemon; consulte username e password. Uma ponte pode, então, autenticar-se para um broker.

Controle de acesso dos tópicos

Se os clientes estiverem sendo autenticados, o daemon também poderá fornecer acesso de controle para tópicos para cada usuário. O daemon concede o controle de acesso com base na correspondência do tópico para o qual um cliente é a publicação ou assinatura com uma sequência de tópicos de acesso no arquivo de controle de acesso; consulte acl_file.

A lista de controle de acesso tem duas partes. A primeira parte controla o acesso para todos os clientes, incluindo clientes anônimos. A segunda parte tem uma seção para qualquer usuário no arquivo de senha. Ele lista o controle de acesso específico para cada usuário.

exemplo

Os parâmetros de segurança são mostrados no exemplo a seguir.

```
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. Arquivo de configuração do daemon

```
Fred:Fredpassword
Barney:Barneypassword
```

Figura 44. Password file, 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. Access control file, acl.txt

Administrando o multicast

Use estas informações para aprender sobre as tarefas de administração do WebSphere MQ Multicast, como reduzir o tamanho de mensagens multicast e ativar a conversão de dados.

Introdução ao Multicast

Use estas informações para iniciar com os tópicos do WebSphere MQ Multicast e objetos de informações de comunicação.

Sobre esta tarefa

WebSphere MQ O sistema de mensagens multicast usa a rede para entregar mensagens mapeando tópicos para endereços de grupo. As tarefas a seguir são uma forma rápida para testar se o endereço IP e a porta necessários estão corretamente configurados para o sistema de mensagens de multicast.

Criando um objeto COMMINFO para multicast

O objeto de informações de comunicação (COMMINFO) contém os atributos associados à transmissão de multicast. Para obter mais informações sobre os parâmetros de objeto COMMINFO, consulte DEFINE COMMINFO.

Use o exemplo da linha de comandos a seguir para definir um objeto COMMINFO para multicast:

```
DEFINE COMMINFO(MC1) GRPADDR(group address) PORT(port number)
```

em que *MC1* é o nome do seu objeto COMMINFO, *group address* é o seu endereço IP ou nome de DNS de multicast de grupo e *port number* é a porta na qual transmitir (o valor padrão é 1414).

Um novo objeto COMMINFO chamado *MC1* é criado; este nome é o nome que você deve especificar ao definir um objeto TOPIC no próximo exemplo.

Criando um objeto TOPIC para multicast

Um tópico é o assunto das informações que são publicadas em uma mensagem de publicação/assinatura e um tópico é definido criando um objeto TOPIC. Os objetos TOPIC possuem dois parâmetros que definem se eles podem ser usados com multicast ou não. Estes parâmetros são: **COMMINFO** e **MCAST**.

- **COMMINFO** Este parâmetro especifica o nome do objeto de informações de comunicação multicast. Para obter mais informações sobre os parâmetros de objeto COMMINFO, consulte DEFINE COMMINFO.
- **MCAST** Este parâmetro especifica se multicast é permitido nesta posição na árvore de tópicos.

Use o exemplo de linha de comandos a seguir para definir um objeto TOPIC para multicast:

```
DEFINE TOPIC(ALLSPORTS) TOPICSTR('Sports') COMMINFO(MC1) MCAST(ENABLED)
```

Um novo objeto TOPIC chamado *ALLSPORTS* é criado. Ele possui uma sequência de tópicos *Sports*, seu objeto de informações de comunicação relacionado é chamado *MC1* (que é o nome especificado ao definir um objeto COMMINFO no exemplo anterior) e multicast é ativado.

Testando a publicação/assinatura multicast

Depois que os objetos TOPIC e COMMINFO foram criados, eles podem ser testados usando a amostra do `amqspubc` e a amostra do `amqssubc`. Para obter informações adicionais sobre estas amostras, consulte Os programas de amostra de Publicação/Assinatura.

1. Abra duas janelas de linha de comandos; a primeira linha de comando é para a amostra de publicação do `amqspubc` e a segunda linha de comandos é para a amostra de assinatura do `amqssubc`.
2. Insira o comando a seguir na linha de comandos 1:

   ```
   amqspubc Sports QM1
   ```

   em que *Sports* é a sequência de tópicos do objeto TOPIC definido em um exemplo anterior, e *QM1* é o nome do gerenciador de filas.
3. Insira o comando a seguir na linha de comandos 2:

   ```
   amqssubc Sports QM1
   ```

   em que *Sports* e *QM1* são os mesmos usados na etapa “2” na página 154.
4. Insira `Hello world` na linha de comandos 1. Se a porta e o endereço IP especificados no objeto COMMINFO estiverem configurados corretamente; a amostra `amqssubc` , que está atendendo na porta para publicações do endereço especificado, gera `Hello world` na linha de comandos 2.

IBM WebSphere MQ Topologia de tópico multicast

Utilize este exemplo para compreender a topologia do tópico do IBM WebSphere MQ Multicast.

O suporte do IBM WebSphere MQ Multicast requer que cada subárvore possua seu próprio grupo de multicast e fluxo de dados dentro da hierarquia total.

O esquema de endereçamento IP de *rede com classes* tem espaço de endereço designado para endereço multicast. O intervalo completo de multicast de endereço IP é `224.0.0.0` a `239.255.255.255`, mas alguns desses endereços são reservados. Para obter uma lista de endereços reservados, entre em contato com o administrador do sistema ou consulte IPv4 Multicast Address Space Registry para obter mais informações. Recomenda-se o uso do endereço multicast com escopo definido localmente na faixa de `239.0.0.0` a `239.255.255.255`.

No diagrama a seguir há dois possíveis fluxos de dados de multicast:

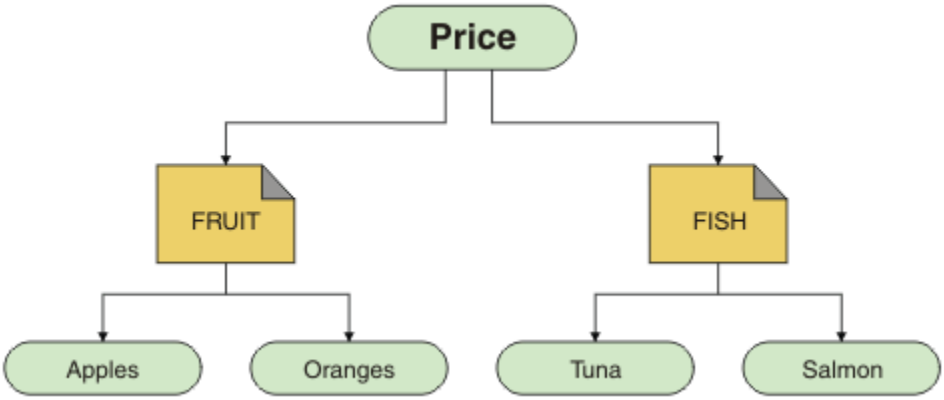
```
DEF COMMINFO(MC1) GRPADDR(239.XXX.XXX.XXX
)

DEF COMMINFO(MC2) GRPADDR(239.YYY.YYY.YYY)
```

em que *239.XXX.XXX.XXX* e *239.YYY.YYY.YYY* são endereços de multicast válidos

Essas definições de tópico são usadas para criar uma árvore de tópicos, conforme mostrado no diagrama a seguir:

```
DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)
DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



Cada objeto de informações de comunicação multicast (COMMINFO) representa um fluxo de dados diferentes porque seus endereços do grupo são diferentes. Neste exemplo, o tópico FRUIT é definido para usar o objeto COMMINFO MC1, o tópico FISH é definido para usar o objeto COMMINFO MC2e o nó Price não possui definições multicast.

WebSphere MQ Multicast possui um limite de 255 caracteres para sequências de tópicos. Essa limitação significa que os cuidados devem ser tomados com os nomes de nós e nós de folha dentro da árvore; se os nomes de nós e nós de folha forem muito longos, a sequência de tópicos poderá exceder 255 caracteres e retornar o código de razão 2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR. É recomendável tornar sequências de tópicos o mais curtas possível porque sequências de tópicos mais longas podem ter um efeito negativo no desempenho.

Controlando o tamanho das mensagens multicast

Use estas informações para aprender sobre o formato da mensagem do WebSphere MQ e reduzir o tamanho das mensagens do WebSphere MQ .

As mensagens do WebSphere MQ possuem vários atributos associados a elas que estão contidos no descritor de mensagens. Para mensagens pequenas, esses atributos podem representar a maior parte do tráfego de dados e pode ter um efeito prejudicial significativo na taxa de transmissão. WebSphere MQ O Multicast permite que o usuário configure quais, se houver, desses atributos são transmitidos juntamente com a mensagem

A presença de atributos de mensagem, diferente de sequência de tópico, depende de se o objeto COMMINFO indica que eles devem ser enviados ou não. Se um atributo não for transmitido, o aplicativo de recebimento se aplicará a um valor padrão. Os valores MQMD padrão não são necessariamente iguais ao valor MQMD_DEFAULT e são descritos em Tabela 9 na página 156.

O objeto COMMINFO contém o atributo MCPROP que controla quantos dos campos MQMD e propriedades do usuário são com a mensagem. Configurando o valor desse atributo para um nível apropriado, é possível controlar o tamanho das mensagens de Multicast do WebSphere MQ :

MCPROP
As propriedades multicast controlam quantas propriedades do MQMD e propriedades do usuário são enviadas com a mensagem.

ALL
Todas as propriedades do usuário e todos os campos do MQMD são transmitidos.

RESPOSTA
Apenas as propriedades do usuário e os campos MQMD que lidam com a resposta das mensagens são transmitidos. Essas propriedades são:

- MsgType
- MessageId
- CorrelId
- ReplyToQ
- ReplyToQmgr

USUÁRIO
Apenas as propriedades do usuário são transmitidas.

NENHUMA
Nenhuma propriedade do usuário ou campo do MQMD é transmitido.

COMPAT
Esse valor faz com que a transmissão da mensagem seja feita em um modo compatível para RMM, o que permite alguma inter-operação com os aplicativos XMS atuais e aplicativos WebSphere Message Broker RMM .

Atributos de mensagens multicast

Atributos de mensagens podem vir de vários locais, como o MQMD, os campos no MQRFH2 e as propriedades de mensagem.

A tabela a seguir mostra o que acontece quando as mensagens são enviadas sujeitas ao valor de MCPROP e o valor padrão usado quando um atributo não é enviado.

Tabela 9. Atributos de Mensagens e como eles se relacionam com multicast

Atribuir	**Ação quando utilizar multicast**	**padrão se não transmitidos**
TopicString	Sempre Incluído	Não-aplicável
MQMQ StrucId	Não transmitido	Não-aplicável
MQMD Versão	Não transmitido	Não-aplicável
Relatório	Incluído se não padrão	0
MsgType	Incluído se não padrão	MQMT_DATAGRAM
Expiração	Incluído se não padrão	0
Feedback	Incluído se não padrão	0
Encoding	Incluído se não padrão	MQENC_NORMAL(equiv)
CodedCharSetId	Incluído se não padrão	1208
Formato	Incluído se não padrão	MQRFH2
Priority	Incluído se não padrão	4
Persistence	Incluído se não padrão	MQPER_NOT_PERSISTENT
MsgId	Incluído se não padrão	Nulo
CorrelId	Incluído se não padrão	Nulo
BackoutCount	Incluído se não padrão	0
ReplyToQ	Incluído se não padrão	Espaço em Branco
ReplyToQMgr	Incluído se não padrão	Espaço em Branco

Tabela 9. Atributos de Mensagens e como eles se relacionam com multicast (continuação)		
Atribuir	**Ação quando utilizar multicast**	**padrão se não transmitidos**
UserIdentifier	Incluído se não padrão	Espaço em Branco
AccountingToken	Incluído se não padrão	Nulo
PutAppIType	Incluído se não padrão	MQAT_JAVA
PutAppIName	Incluído se não padrão	Espaço em Branco
PutDate	Incluído se não padrão	Espaço em Branco
PutTime	Incluído se não padrão	Espaço em Branco
ApplOriginData	Incluído se não padrão	Espaço em Branco
GroupID	Excluído	Não-aplicável
MsgSeqNumber	Excluído	Não-aplicável
Offset	Excluído	Não-aplicável
MsgFlags	Excluído	Não-aplicável
OriginalLength	Excluído	Não-aplicável
UserProperties	Incluído	Não-aplicável

Referências relacionadas
ALTER COMMINFO
DEFINE COMMINFO

Ativando a conversão de dados para mensagens multicast

Use estas informações para entender como a conversão de dados funciona para o sistema de mensagens WebSphere MQ Multicast.

WebSphere MQ Multicast é um protocolo compartilhado, sem conexão e, portanto, não é possível para cada cliente fazer solicitações específicas para conversão de dados Cada cliente inscrito no mesmo fluxo multicast recebe os mesmos dados binários; portanto, se a conversão de dados do WebSphere MQ for necessária, a conversão será executada localmente em cada cliente.

Em uma instalação de plataforma mista, pode ser que a maioria dos clientes requeiram os dados em um formato que não seja o formato nativo do aplicativo de transmissão. Nessa situação, os valores **CCSID** e **ENCODING** do objeto multicast COMMINFO podem ser utilizados para definir a codificação da transmissão da mensagem para eficiência.

WebSphere MQ O Multicast suporta a conversão de dados da carga útil da mensagem para os seguintes formatos integrados:

- MQADMIN
- MQEVENT
- MQPCF
- MQRFH
- MQRFH2
- MQSTR

Além desses formatos, também é possível definir seus próprios formatos e utilizar uma saída de conversão de dados parâmetro MQDXP – saída de conversão de dados.

Para obter informações sobre programação de dados conversões, consulte Conversão de dados no MQI para sistema de mensagens multicast.

Para obter mais informações sobre a conversão de dados, consulte Conversão de dados.

Para obter mais informações sobre saídas de conversão de dados e `ClientExitPath`, consulte ClientExitPath subrotina do arquivo de configuração do cliente.

Monitoramento de aplicativo multicast

Use estas informações para saber como administrar e monitorar o WebSphere MQ Multicast.

O status do atual publicadores e assinantes para tráfego multicast (por exemplo, o número de mensagens enviadas e recebidas ou o número de mensagens perdidas) é transmitido periodicamente para o servidor do cliente. Quando o status é recebido, o atributo COMMEV do objeto COMMINFO especifica se o gerenciador de filas coloca ou não uma mensagem do evento em SYSTEM.ADMIN.PUBSUB.EVENT. A mensagem do evento contém as informações de status recebidas. Essa informação é um auxílio de diagnóstico de um valor inestimável para localizar a origem de um problema.

Use o comando do MQSC **DISPLAY CONN** para exibir informações de conexão sobre os aplicativos conectados ao gerenciador de fila Para obter mais informações sobre o comando **DISPLAY CONN**, consulte DISPLAY CONN.

Use o comando MQSC **DISPLAY TPSTATUS** para exibir o status de seus publicadores e assinantes. Para obter mais informações sobre o comando **DISPLAY TPSTATUS** , consulte DISPLAY TPSTATUS..

COMMEV e o indicador de confiabilidade de mensagem multicast

O *indicador de confiabilidade*, utilizado em conjunto com o atributo **COMMEV** do objeto COMMINFO, é um elemento chave no monitoramento de publicadores e assinantes do WebSphere MQ Multicast. O indicador de confiabilidade (o campo **MSGREL** que é retornado nos comandos de status de Publicação ou Assinatura) é um indicador do WebSphere MQ que ilustra a porcentagem de transmissões que não possuem erros. Às vezes, as mensagens precisam ser retransmitidas devido a um erro de transmissão, que é refletido no valor de **MSGREL**. Potenciais causas de erros de transmissão incluem assinantes lentos, redes ocupadas e interrupções na rede. **COMMEV** controla se as mensagens de eventos são geradas para manipulações multicast que são criadas utilizando o objeto COMMINFO e configuradas para um dos três valores possíveis:

DISABLED
Mensagens do evento não são gravadas.

ATIVADO
Mensagens de eventos são sempre gravadas, com uma frequência definida no parâmetro **MONINT** de COMMINFO.

EXCEÇÃO
As mensagens de evento são escritas se a confiabilidade da mensagem está abaixo do limite de confiabilidade. Um nível de confiabilidade de mensagem de 90% ou menos indica que pode haver um problema com a configuração de rede ou que um ou mais dos aplicativos de Publicação/Assinatura está em execução muito lenta:

- Um valor de **MSGREL(100,100)** indica que não houve nenhum problema no curto prazo ou longo prazo.
- Um valor **MSGREL(80,60)** indica que 20% das mensagens estão atualmente tendo problemas, mas que é igualmente uma melhoria no valor de longo prazo de 60.

Os clientes podem continuar a transmitir e receber tráfego multicast mesmo quando a conexão unicast para o gerenciador de filas é quebrada, portanto, os dados podem estar desatualizados.

Confiabilidade de mensagem multicast

Use estas informações para aprender como configurar a assinatura do WebSphere MQ Multicast e o histórico de mensagens

Um elemento-chave para superar a falha de transmissão com multicast é o buffer de dados transmitidos do WebSphere MQ(um histórico de mensagens a ser mantido na extremidade de transmissão do link). Esse processo significa que nenhum buffer de mensagens é necessário no processo de aplicativo de

colocação porque o WebSphere MQ fornece a confiabilidade. O tamanho deste histórico é configurado por meio do objeto de informações de comunicação (COMMINFO), conforme descrito nas informações a seguir. Um buffer de transmissão maior significa que há mais histórico de transmissão a ser retransmitido, se necessário, mas devido à natureza do multicast, 100% de garantia de entrega não pode ser suportado.

O histórico de mensagens multicast do WebSphere MQ é controlado no objeto de informações de comunicação (COMMINFO) pelo atributo **MSGHIST** :

MSGHIST
Esse valor é a quantia de histórico de mensagens em kilobytes que é mantida pelo sistema para manipular retransmissões no caso de NACKs (reconhecimentos negativos).

Um valor igual a 0 fornece o menor nível de confiabilidade. O valor padrão é 100 KB.

O histórico da nova assinatura do WebSphere MQ Multicast é controlado no objeto de informações de comunicação (COMMINFO) pelo atributo **NSUBHIST** :

NSUBHIST
O novo histórico do assinante controla se um assinante que une-se a um fluxo de publicação recebe todos os dados que estiverem disponíveis atualmente ou recebe apenas as publicações feitas a partir do momento da assinatura.

NENHUMA
Um valor de NONE faz o transmissor transmitir apenas a publicação feita a partir do momento da assinatura. NONE é o valor padrão.

ALL
Um valor ALL faz com que o transmissor retransmita todos os históricos de tópicos que forem conhecidos. Em algumas circunstâncias, esta situação pode causar um comportamento semelhante nas publicações retidas.

Nota: Usar um valor igual a ALL pode ter um efeito negativo no desempenho se houver um grande histórico do tópico, pois todo o histórico do tópico é retransmitido.

Referências relacionadas
DEFINE COMMINFO
ALTER COMMINFO

Tarefas avançadas de multicast

Use estas informações para aprender sobre as tarefas avançadas de administração do WebSphere MQ Multicast, como configurar arquivos .ini e interoperabilidade com o LLM do WebSphere MQ .

Para obter considerações sobre segurança em uma instalação Multicast, consulte Segurança Multicast .

Ponte entre domínios de publicação/assinatura multicast e não multicast

Use essas informações para entender o que acontece quando um publicador não multicast publica em um tópico ativado do WebSphere MQ Multicast

Se um publicador não multicast publica em um tópico que está definido como ativado **MCAST** e **BRIDGE** ativado, o gerenciador de filas transmite a mensagem de saída através de multicast diretamente para todos os assinantes que podem estar atendendo. Um publicador não pode publicar para tópicos que não são ativados para multicast.

Tópicos existentes podem ser multicast ativados por meio da configuração dos parâmetros **MCAST** e **COMMINFO** de um objeto do tópico. Veja Conceitos de multicast iniciais para obter mais informações sobre esses parâmetros.

O atributo **BRIDGE** do objeto COMMINFO controla as publicações de aplicativos que não estão utilizando multicast. Se **BRIDGE** está configurado como ENABLED e o parâmetro **MCAST** do tópico também está configurado como ENABLED, as publicações de aplicativos que não estão utilizando multicast são vinculadas aos aplicativos que estão. Para obter mais informações sobre o parâmetro **BRIDGE**, consulte DEFINE COMMINFO.

Configurando os arquivos .ini para Multicast

Use estas informações para entender os campos de Multicast do WebSphere MQ nos arquivos `.ini`

Adicional WebSphere MQ Configuração Multicast pode ser feita em um arquivo `ini` . O arquivo `ini` específico que deve-se utilizar depende do tipo de aplicativos:

- Cliente: configure o arquivo *MQ_DATA_PATH*`/mqclient.ini` .
- Gerenciador de filas: Configure o arquivo *MQ_DATA_PATH*`/qmgrs/`*QMNAME*`/qm.ini` .

em que *MQ_DATA_PATH* é o local do diretório de dados do WebSphere MQ (`/var/mqm/mqclient.ini`) e *QMNAME* é o nome do gerenciador de filas ao qual o arquivo `.ini` se aplica

O arquivo `.ini` contém campos usados para ajustar o comportamento do 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
Neste modo, os pacotes são enviados utilizando o protocolo UDP. Elementos de rede não pode fornecer assistência na distribuição de multicast como fazem em modo de IP no entanto. O formato do pacote permanece compatível com o PGM. Esse é o valor-padrão.

IP
Nesse modo, o transmissor envia pacotes de IP brutos. Elementos da rede com o assistente de suporte PGM na distribuição de pacote multicast confiável. Esse modo é totalmente compatível com o padrão PGM.

IPVersion

IPV4
Comunique-se usando apenas o protocolo IPv4 Esse é o valor-padrão.

IPV6
Comunique-se usando apenas o protocolo IPv6

QUALQUER
Comunique-se usando IPv4, IPv6ou ambos, dependendo de qual protocolo está disponível.

AMBOS
Suporta a comunicação usando IPv4 e IPv6.

LimitTransRate

DISABLED
Não há nenhum controle de taxa de transmissão. Esse é o valor-padrão.

STATIC
Implementa de controle de taxa de transmissão estático. O transmissor não transmitiria em uma taxa que excedesse a taxa especificada pelo parâmetro TransRateLimit.

DINÂMICO
O transmissor se adapta à sua taxa de transmissão de acordo com o feedback que obtém dos receptores. Neste caso, o limite de taxa de transmissão não pode ser maior que o valor especificado pelo parâmetro TransRateLimit. O transmissor tenta alcançar uma taxa de transmissão otimizada.

TransRateLimit
O limite da taxa de transmissão em Kbps.

SocketTTL
O valor de SocketTTL determina se o tráfego multicast pode passar por um roteador ou o número de roteadores pelos quais ele pode passar.

Batch
Controla se as mensagens são colocadas em lote ou enviadas imediatamente. Existem 2 valores possíveis:

- *NO* As mensagens não são colocadas em lote, elas são enviadas imediatamente.
- As mensagens são*YES* em lote.

Loop
Configure o valor para 1 para ativar loop multicast. loop Multicast define se os dados enviados são um circuito fechado para o host ou não.

Interface
O endereço IP da interface no qual os fluxos de tráfego multicast. Para obter mais informações e resolução de problemas, consulte: Testando aplicativos multicast em uma rede não multicast e Configurando a rede apropriada para tráfego multicast

FeedbackMode

NACK
Feedback, reconhecimentos negativos. Esse é o valor-padrão.

ACK
feedback por confirmações positivas.

WAIT1
Feedback por confirmações positivas quando o transmissor aguarda somente 1 ACK a partir de qualquer um dos receptores.

HeartbeatTimeout
O tempo limite de pulsação em milissegundos. Um valor igual a 0 indica que os eventos de tempo limite de pulsação não são gerados pelo receptor ou receptores do tópico. O valor padrão é 20000.

HeartbeatInterval
O intervalo de pulsação em milissegundos. Um valor igual a 0 indica que nenhuma pulsação é enviada. O intervalo de pulsação deve ser consideravelmente menor que o valor de **HeartbeatTimeout** para evitar eventos de tempo limite de pulsação falso. O valor-padrão é 2000.

Interoperabilidade multicast com o WebSphere MQ Sistema de Mensagens de Baixa Latência

Use estas informações para entender a interoperabilidade entre o WebSphere MQ Multicast e o WebSphere MQ Low Latency Messaging (LLM).

A transferência de carga útil básica é possível para um aplicativo que usa o LLM, com outro aplicativo usando multicast para trocar mensagens em ambas as direções. Embora multicast utilize tecnologia LLM, o próprio produto LLM não é integrado. Portanto, é possível instalar o LLM e o WebSphere MQ Multicast e operar e atender os dois produtos separadamente.

Aplicativos LLM que se comunicam com multicast podem precisar enviar e receber propriedades de mensagem. As propriedades de mensagem WebSphere MQ e os campos MQMD são transmitidos como propriedades de mensagem LLM com códigos de propriedade de mensagem LLM específicos, conforme mostrado na tabela a seguir:

Tabela 10. Propriedades de mensagem do WebSphere MQ para mapeamentos de propriedade do WebSphere MQ LLM

propriedade WebSphere MQ	WebSphere MQ tipo de propriedade LLM	tipo de propriedade LLM	código de propriedade 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 obter mais informações sobre o LLM, consulte a documentação do produto LLM: WebSphere MQ Low Latency Messaging.

administrandoHP Integrity NonStop Server

Use estas informações para aprender sobre tarefas de administração para o cliente IBM WebSphere MQ para HP Integrity NonStop Server.

Duas tarefas de administração estão disponíveis para você:

1. Iniciando Manualmente o TMF/Gateway de Pathway.
2. Parando o TMF/Gateway de Pathway.

Iniciando Manualmente o TMF/Gateway de Pathway

É possível permitir que Pathway inicie automaticamente o TMF/Gateway na primeira solicitação de cadastramentoe é possível iniciar manualmente o TMF/Gateway de Pathway.

Procedimento

Para iniciar manualmente o TMF/Gateway de Pathway, digite o comando PATHCOM seguinte:

```
START SERVER <server_class_name>
```

Se um aplicativo cliente fizer uma solicitação de cadastramento antes de o TMF/Gateway concluir a recuperação de transações indeterminadas, a solicitação será retida por até 1 segundo. Se a recuperação

não for concluída dentro desse tempo, o cadastramento será rejeitado. Em seguida, o cliente recebe um erro MQRC_UOW_ENLISTMENT_ERROR do uso de um MQI transacional.

Parando o TMF/Gateway de Pathway

Esta tarefa descreve como parar o TMF/Gateway a partir de Pathway, e como reiniciar o TMF/Gateway após pará-lo.

Procedimento

1. Para evitar que novas solicitações de inscrição sejam feitas ao TMF/Gateway, digite o seguinte comando:

   ```
   FREEZE SERVER <server_class_name>
   ```

2. Para acionar o TMF/Gateway para concluir quaisquer operações em andamento e para terminar, digite o seguinte comando:

   ```
   STOP SERVER <server_class_name>
   ```

3. Para permitir que o TMF/Gateway reinicie, automaticamente na primeira inscriçãoe manualmente, seguindo as etapas 1 e 2, digite o seguinte comando:

   ```
   THAW SERVER <server_class_name>
   ```

   Aplicativos são impedidos de fazer novas solicitações de cadastramento e não é possível emitir o comando **START** START até que você emita o comando **THAW**.

Avisos

Estas informações foram desenvolvidas para produtos e serviços oferecidos nos Estados Unidos.

É possível que a IBM não ofereça os produtos, serviços ou recursos discutidos nesta publicação em outros países. Consulte seu representante local do IBM para obter informações sobre produtos e serviços disponíveis atualmente em sua área. Qualquer referência a um IBM produto, programa ou serviço não se destina a estado ou significa que apenas esse produto IBM, programas ou serviços possam ser utilizados. Qualquer produto, programa ou serviço funcionalmente equivalente, que não infrinja nenhum direito de propriedade intelectual da IBM poderá ser utilizado em substituição. Entretanto, a avaliação e verificação da operação de qualquer produto, programa ou serviço não IBM são de responsabilidade do Cliente.

A IBM pode ter patentes ou aplicativos de patentes pendentes relativas aos assuntos tratados nesta publicação. O fornecimento desta publicação não garante ao Cliente nenhum sobre tais patentes. é possível enviar pedidos de licença, por escrito, para:

Gerência de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur, 138-146
Botafogo
Rio de Janeiro, RJ
CEP 22290-240

Para pedidos de licença relacionados a informações de DBCS (Conjunto de Caracteres de Byte Duplo), entre em contato com o Departamento de Propriedade Intelectual da IBM em seu país ou envie pedidos de licença, por escrito, para:

Licenciamento de Propriedade Intelectual e Direito de Propriedade Intelectual IBM Japão, Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japão

O parágrafo a seguir não se aplica a nenhum país em que tais disposições não estejam de acordo com a legislação local: A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO "NO ESTADO EM QUE SE ENCONTRA", SEM GARANTIA DE NENHUM TIPO, SEJA EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS A ELAS NÃO SE LIMITANDO, AS GARANTIAS IMPLÍCITAS DE NÃO INFRAÇÃO, COMERCIALIZAÇÃO OU ADEQUAÇÃO A UM DETERMINADO PROPÓSITO. Alguns países não permitem a exclusão de garantias expressas ou implícitas em certas transações; portanto, essa disposição pode não se aplicar ao Cliente.

Essas informações podem conter imprecisões técnicas ou erros tipográficos. Periodicamente, são feitas nas informações aqui contidas; essas alterações serão incorporadas em futuras edições desta publicação. IBM pode aperfeiçoar e/ou alterar no produto(s) e/ou programa(s) descritos nesta publicação a qualquer momento sem aviso prévio.

Referências nestas informações a Web sites não-IBM são fornecidas apenas por conveniência e não representam de forma alguma um endosso a esses Web sites. Os materiais contidos nesses websites não fazem parte dos materiais desse produto IBM e a utilização desses websites é de inteira responsabilidade do Cliente.

A IBM pode utilizar ou distribuir as informações fornecidas da forma que julgar apropriada sem incorrer em qualquer obrigação para com o Cliente.

Licenciados deste programa que desejam obter informações sobre este assunto com objetivo de permitir: (i) a troca de informações entre programas criados independentemente e outros programas (incluindo este) e (ii) a utilização mútua das informações trocadas, devem entrar em contato com:

IBM Coordenador de Interoperabilidade de Software da Corporação, Departamento 49XA 3605 Highway 52 N Rochester, MN 55901 U.S.A.

Tais informações podem estar disponíveis, sujeitas a termos e condições apropriadas, incluindo em alguns casos o pagamento de uma taxa.

O programa licenciado descrito nesta publicação e todo o material licenciado disponível para ele são fornecidos pela IBM sob os termos do IBM Customer Agreement, IBM Contrato de Licença do Programa Internacional ou qualquer contrato equivalente entre as partes.

Todos os dados de desempenho aqui contidos foram determinados em um ambiente controlado. Portanto, os resultados obtidos em outros ambientes operacionais podem variar significativamente. Algumas medidas podem ter sido tomadas em sistemas em nível de desenvolvimento e não há garantia de que estas medidas serão iguais em sistemas geralmente disponíveis. Além disto, algumas medidas podem ter sido estimadas através de extrapolação. Os resultados reais podem variar. usuários deste documento devem verificar os dados aplicáveis para seu ambiente específico.

As informações relativas a produtos não IBM foram obtidas junto aos fornecedores dos respectivos produtos, de seus anúncios publicados ou de outras fontes disponíveis publicamente. A IBM não testou estes produtos e não pode confirmar a precisão de seu desempenho, compatibilidade nem qualquer outra reivindicação relacionada a produtos não IBM. Dúvidas sobre os recursos de produtos não IBM devem ser encaminhadas diretamente a seus fornecedores.

Todas as declarações relacionadas aos objetivos e intenções futuras da IBM estão sujeitas a alterações ou cancelamento sem aviso prévio e representam apenas metas e objetivos.

Essas informações contêm exemplos de dados e relatórios utilizados em operações diárias de negócios. Para ilustrá-los da forma mais completa possível, os exemplos incluem nomes de indivíduos, empresas, marcas e produtos. Todos estes nomes são fictícios e qualquer semelhança com os nomes e endereços utilizados por uma empresa real é mera coincidência.

LICENÇA DE COPYRIGHT :

Estas informações contêm programas de aplicativos de amostra na linguagem fonte, ilustrando as técnicas de programação em diversas plataformas operacionais. O Cliente pode copiar, modificar e distribuir estes programas de amostra sem a necessidade de pagar à IBM, com objetivos de desenvolvimento, uso, marketing ou distribuição de programas aplicativos em conformidade com a interface de programação de aplicativo para a plataforma operacional para a qual os programas de amostra são criados. Esses exemplos não foram testados completamente em todas as condições. Portanto, a IBM não pode garantir ou implicar a confiabilidade, manutenção ou função destes programas.

Se estiver visualizando estas informações em formato eletrônico, as fotografias e ilustrações coloridas poderão não aparecer.

Informações sobre a Interface de Programação

As informações da interface de programação, se fornecidas, destinam-se a ajudá-lo a criar software aplicativo para uso com este programa.

Este manual contém informações sobre interfaces de programação desejadas que permitem que o cliente grave programas para obter os serviços do IBM WebSphere MQ.

No entanto, estas informações também podem conter informações sobre diagnósticos, modificações e ajustes. As informações sobre diagnósticos, modificações e ajustes são fornecidas para ajudá-lo a depurar seu software aplicativo.

Importante: Não use essas informações de diagnóstico, modificação e ajuste como uma interface de programação, pois elas estão sujeitas a mudanças

Marcas comerciais

IBM, o logotipo IBM , ibm.com, são marcas registradas da IBM Corporation, registradas em várias jurisdições no mundo todo Uma lista atual de marcas registradas da IBM está disponível na Web em "Informações de copyright e marca registrada"www.ibm.com/legal/copytrade.shtml. Outros nomes de produtos e serviços podem ser marcas comerciais da IBM ou de outras empresas.

Microsoft e Windows são marcas comerciais da Microsoft Corporation nos Estados Unidos e/ou em outros países.

UNIX é uma marca registrada do The Open Group nos Estados Unidos e em outros países.

Linux é uma marca registrada de Linus Torvalds nos Estados Unidos e/ou em outros países.

Este produto inclui software desenvolvido pelo Projeto Eclipse (http://www.eclipse.org/).

Java e todas as marcas comerciais e logotipos baseados em Java são marcas comerciais ou marcas registradas da Oracle e/ou de suas afiliadas.

Part Number:

(1P) P/N: