7.5

Desenvolvendo Aplicativos para o IBM WebSphere MQ

Nota

Antes de usar estas informações e o produto que elas suportam, leia as informações em “Avisos” na página 1137.

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

Desenvolvendo Aplicativos

O IBM WebSphere MQ fornece várias maneiras nas quais é possível desenvolver aplicativos para enviar e receber mensagens necessárias para suportar seus processos de negócios. É possível também desenvolver os aplicativos para gerenciar os gerenciadores de fila e recursos relacionados.

Antes de desenvolver aplicativos para o IBM WebSphere MQ, assegure-se de estar familiarizado com os conceitos em IBM WebSphere MQ Visão geral técnicaIBM WebSphere MQ Visão geral técnica.

É possível desenvolver aplicativos para o IBM WebSphere MQ em várias linguagens de programação diferentes. Para obter informações sobre as linguagens de programação suportadas e seus recursos, consulte “Decidindo qual linguagem de programação usar” na página 80..

Consulte as seções a seguir para obter os tipos de aplicativos que podem ser gravados para o IBM WebSphere MQ em diferentes plataformas

Tipos de aplicativos que podem ser gravados para IBM WebSphere MQ

Essas informações são sobre os tipos de aplicativos que podem ser gravados no IBM WebSphere MQ.

Os produtos IBM WebSphere MQ são gerenciadores de filas e ativadores de aplicativos Eles suportam o Message Queue Interface (MQI) IBM por meio do qual os programas podem colocar mensagens em uma fila e obter mensagens de uma fila.

Com IBM WebSphere MQ para plataformas nãoz/OS , é possível gravar aplicativos que:

- Enviar mensagens para outros aplicativos em execução sob os mesmos sistemas operacionais. Os aplicativos podem estar no mesmo sistema ou em outro.
- Enviar mensagens para aplicativos que são executados em outras plataformas IBM WebSphere MQ.
- Use o enfileiramento de mensagem de dentro do CICS para TXSeries para AIX, TXSeries para HP-UX, TXSeries para Solaris e TXSeries para aplicativos de sistemas Windows .
- Use o enfileiramento de mensagens de dentro de Encina para AIX, HP-UX, Solaris e sistemas Windows ..
- Use o enfileiramento de mensagens de dentro do Tuxedo para os sistemas AIX, AT & T, HP-UX, Solaris e Windows
- Use o IBM WebSphere MQ como um gerenciador de transações, coordenando as atualizações feitas pelos gerenciadores de recursos externos nas unidades de trabalho do IBM WebSphere MQ. Os gerenciadores de recursos externos a seguir são suportados e compatíveis com a interface X/OPEN XA
  - DB2
  - Informix
  - Oracle
  - Sybase
- Processe várias mensagens juntas como uma única unidade de trabalho que pode ser confirmada ou restaurada.
- Execute a partir de um ambiente integral do IBM WebSphere MQ ou a partir de um ambiente do cliente MQI do IBM WebSphere MQ nas plataformas a seguir:
  - UNIX and Linux® sistemas
  - Windows

Conceitos relacionados

Segurança

Conceitos de desenvolvimento de aplicativos

É possível usar uma opção de linguagens orientadas a objeto ou processuais para gravar aplicativos IBM WebSphere MQ. Use os links neste tópico para obter informações sobre conceitos do IBM WebSphere MQ que são úteis para desenvolvedores de aplicativos.

Antes de começar a projetar e gravar seus aplicativos IBM WebSphere MQ , familiarize-se com os conceitos básicos do IBM WebSphere MQ , consulte os tópicos em Visão geral técnica. Para obter informações sobre os tipos de aplicativo que podem ser gravados para o IBM WebSphere MQ, consulte “Desenvolvendo Aplicativos” na página 7

Use os links a seguir para descobrir sobre os conceitos do IBM WebSphere MQ específicos do desenvolvimento de aplicativos:

- “Mensagens IBM WebSphere MQ” na página 9
- Sistema de mensagens ponto a ponto
- Introdução ao Sistema de Mensagens de Publicação/Assinatura do WebSphere MQ
- “Usando a interface da fila de mensagens (MQI) em um aplicativo cliente” na página 357
- “Usando serviços da Web no WebSphere MQ” na página 958
- “Programas de Saída de Canal para Canais de Mensagens” na página 402
- “Cenários de Suporte Transacional” na página 42

Antes de poder executar aplicativos que usam o MQI, deve-se criar certos objetos do IBM WebSphere MQ. Para obter informações adicionais, consulte “Programas aplicativos usando a MQI” na página 9.

Conceitos relacionados

“Projetando aplicativos IBM WebSphere MQ” na página 91
Quando você tiver decidido como seus aplicativos podem aproveitar as plataformas e ambientes disponíveis para você, será necessário decidir como usar os recursos oferecidos pelo WebSphere MQ.

“Programas de amostra do WebSphere MQ” na página 98
Use esta coleção de tópicos para aprender sobre programas de amostra do WebSphere MQ em diferentes plataformas

“Gravando um Aplicativo de Enfileiramento” na página 197
Use essas informações para aprender como escrever aplicativos de enfileiramento, conectar e desconectar de um gerenciador de filas, publicar/assinar e abrir e fechar objetos.

“Gravando aplicativos clientes” na página 356
O que você precisa saber para gravar os aplicativos clientes no WebSphere MQ

“Decidindo qual linguagem de programação usar” na página 80
Use estas informações para descobrir sobre linguagens de programação e estruturas suportadas pelo IBM WebSphere MQe algumas considerações para usá-las.

“Usando classes do WebSphere MQ para JMS” na página 724
WebSphere MQ classes para Java Message Service (WebSphere MQ classes para JMS) é o provedor JMS que é fornecido com o WebSphere MQ. Além de implementar as interfaces definidas no pacote javax.jms , as classes WebSphere MQ para JMS fornecem dois conjuntos de extensões para a API JMS.

“Usando o Component Object Model Interface (WebSphere MQ Classes de Automação para ActiveX)” na página 1045
O WebSphere MQ Automation Classes para ActiveX (MQAX) são componentes ActiveX que fornecem classes que podem ser usadas em seu aplicativo para acessar o WebSphere MQ

“Usando classes do WebSphere MQ para Java” na página 660
As classes do WebSphere MQ para Java permitem usar o WebSphere MQ em um ambiente Java. Um aplicativo Java pode usar classes WebSphere MQ para Java ou classes WebSphere MQ para JMS para acessar recursos do WebSphere MQ .

“Usando .NET” na página 567

WebSphere MQ classes para .NET permitem que um programa gravado na estrutura de programação .NET se conecte ao WebSphere MQ como um cliente MQI do WebSphere MQ ou se conecte diretamente a um servidor WebSphere MQ .

“Usando C++” na página 636
WebSphere MQ fornece classes C++ equivalentes a objetos WebSphere MQ e algumas classes adicionais equivalentes aos tipos de dados da matriz. Ele fornece inúmeros recursos não disponíveis por meio do MQI.

“Construindo um aplicativo IBM WebSphere MQ” na página 435
Use estas informações para saber como construir um aplicativo IBM WebSphere MQ em diferentes plataformas.

Programas aplicativos usando a MQI

Programas de aplicativoIBM WebSphere MQ precisam de determinados objetos antes que eles possam ser executados com sucesso.

Figura 1 na página 9 mostra um aplicativo que remove mensagens de uma fila, processa-as e, em seguida, envia alguns resultados para outra fila no mesmo gerenciador de filas.

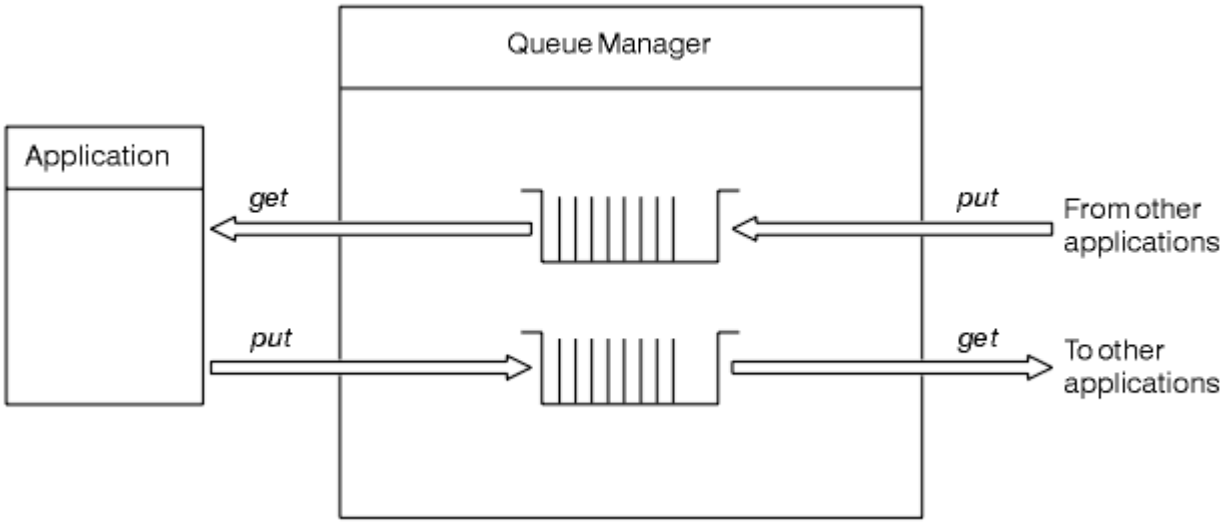


Figura 1. Filas, mensagens e aplicativos

Enquanto os aplicativos podem colocar mensagens em filas locais ou remotas (usando MQPUT), eles só podem obter mensagens diretamente de filas locais (usando MQGET).

Antes que esse aplicativo possa ser executado, as condições a seguir devem ser satisfeitas:

- O gerenciador de filas deve existir e estar em execução.
- A primeira fila do aplicativo, da qual as mensagens devem ser removidas, deve ser definida.
- A segunda fila, na qual o aplicativo coloca as mensagens, também deve ser definida.
- O aplicativo deve ser capaz de se conectar ao gerenciador de filas. Para fazer isso, ele deve ser vinculado ao IBM WebSphere MQ. Consulte o “Construindo um aplicativo IBM WebSphere MQ” na página 435.
- Os aplicativos que colocam as mensagens na primeira fila também devem se conectar a um gerenciador de filas. Se eles forem remotos, também devem ser configurados com filas de transmissão e canais. Esta parte do sistema não é mostrada em Figura 1 na página 9.

Mensagens IBM WebSphere MQ

Estas informações apresentam o conceito de mensagens do IBM WebSphere MQ, partes da mensagem e o descritor de mensagens.

As mensagens do IBM WebSphere MQ consistem em duas partes:

- Propriedades da Mensagem
- Dados do aplicativo

O Figura 2 na página 10 representa uma mensagem e mostra como ele é logicamente dividido em propriedades de mensagem e dados do aplicativo.

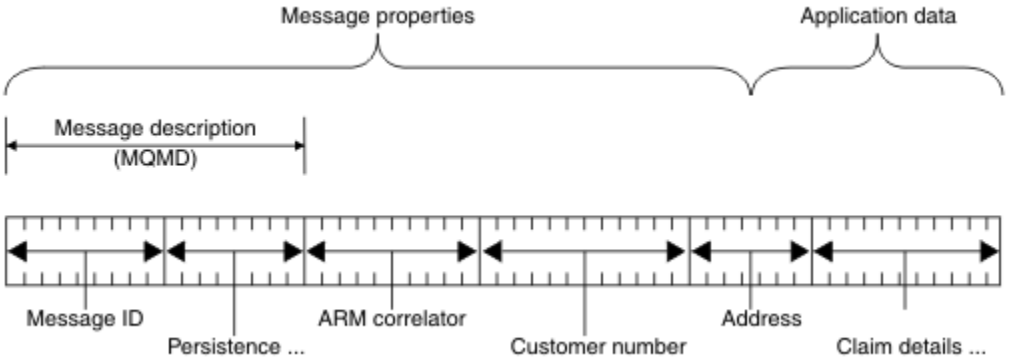


Figura 2. Representação de uma mensagem

Os dados do aplicativo transportados em uma mensagem do WebSphere MQ não são mudados por um gerenciador de filas, a menos que a conversão de dados seja executada nele Além disso, WebSphere MQ não coloca nenhuma restrição no conteúdo desses dados. O comprimento dos dados em cada mensagem não pode exceder o valor do atributo *MaxMsgLength* da fila e do gerenciador de filas.

Em WebSphere MQ para AIX, WebSphere MQ para HP-UX, WebSphere MQ para Linux, WebSphere MQ para Solaris e WebSphere MQ para Janelas, o *MaxMsgLength* é padronizado para 100 MB (104 857 600 bytes).

Faça suas mensagens um pouco menores que o valor do atributo *MaxMsgLength* em algumas circunstâncias. Consulte “Os dados em sua mensagem” na página 233 para obter mais informações..

Você cria uma mensagem quando usa as chamadas MQPUT ou MQPUT1 MQI. Como entrada para essas chamadas, você fornece as informações de controle (como a prioridade da mensagem e o nome de uma fila de resposta) e seus dados e a chamada, então, coloca a mensagem em uma fila. Consulte MQPUT e MQPUT1 para obter mais informações sobre estas chamadas.

Descritor de Mensagens

É possível acessar as informações de controle de mensagem usando a estrutura MQMD, que define o *descritor de mensagens*

Para obter uma descrição integral da estrutura MQMD, consulte MQMD – Descritor de mensagens.

Consulte “Contexto da mensagem” na página 39 para obter uma descrição de como usar os campos no MQMD que contém as informações sobre a origem da mensagem.

Existem diferentes versões do descritor de mensagens. Informações adicionais para agrupamento e segmentação de mensagens (consulte “Grupos de mensagens” na página 36) são fornecidas na Versão 2 do descritor de mensagens (ou o MQMDE). Ele é igual ao descritor de mensagens da Versão 1, mas possui campos adicionais. Eles são descritos na extensão do descritor de mensagens MQMDE.

Tipos de Mensagem

Existem quatro tipos de mensagens definidos por IBM WebSphere MQ.

Estas quatro mensagens são:

- Datagrama
- Mensagens de solicitação
- Mensagens de resposta

- Mensagens de relatório
  - Tipos de mensagem de relatório
  - Opções de mensagem de relatório

Aplicativos podem usar os três primeiros tipos de mensagens para passar as informações entre si. O quarto tipo, relatório, é para os aplicativos e os gerenciadores de fila usarem para relatar as informações sobre eventos como a ocorrência de um erro.

Cada tipo de mensagem é identificado por um valor MQMT_*. Também é possível definir seus próprios tipos de mensagens. Para o intervalo de valores que podem ser usados, consulte MsgType.

Datagramas

Use um *datagrama* quando não precisar de uma resposta do aplicativo que recebe a mensagem (ou seja, recebe a mensagem a partir da fila).

Um exemplo de um aplicativo que pode usar os datagramas é aquele que exibe as informações de voo em um salão do aeroporto. Uma mensagem pode conter os dados para uma tela inteira de informações do voo. É pouco provável que tal aplicativo solicite um reconhecimento para uma mensagem porque provavelmente não importa se uma mensagem não é entregue. O aplicativo envia uma mensagem de atualização após um curto período.

Mensagens de Pedidos

Use uma *mensagem de solicitação* quando desejar uma resposta do aplicativo que recebe a mensagem.

Um exemplo de um aplicativo que poderia usar mensagens de solicitação é um que exiba o saldo de uma conta de verificação. A mensagem de solicitação poderia conter o número da conta e a mensagem de resposta iria conter o saldo da conta.

Se desejar vincular sua mensagem de resposta a sua mensagem de solicitação, existem duas opções:

- Torne o aplicativo que trata da mensagem de solicitação responsável por assegurar que coloque as informações na mensagem de resposta relacionada à mensagem de solicitação.
- Use o campo de relatório no descritor de mensagens da sua mensagem de solicitação para especificar o conteúdo dos campos *MsgId* e *CorrelId* da mensagem de resposta:
  - É possível solicitar que o *MsgId* ou o *CorrelId* da mensagem original seja copiado para o campo *CorrelId* da mensagem de resposta (a ação padrão é copiar *MsgId*).
  - É possível solicitar que um novo *MsgId* seja gerado para a mensagem de resposta ou que o *MsgId* da mensagem original seja copiado para o campo *MsgId* da mensagem de resposta (a ação padrão é gerar um novo identificador de mensagem).

Mensagens de Resposta

Use uma *mensagem de resposta* ao responder outra mensagem.

Ao criar uma mensagem de resposta, respeite quaisquer opções que tenham sido configuradas no descritor de mensagens da mensagem à qual estiver respondendo. Opções de relatório especificam o conteúdo do identificador de mensagem (*MsgId*) e campos do identificador de correlação (*CorrelId*). Esses campos permitem que o aplicativo que recebe a resposta correlacioná-la com sua solicitação original.

Mensagens de relatório

Mensagens de relatório informam os aplicativos sobre os eventos como a ocorrência de um erro ao processar uma mensagem.

Elas podem ser geradas por:

- Um gerenciador de filas,

- Um agente do canal de mensagem (por exemplo, se não puder entregar a mensagem) ou
- Um aplicativo (por exemplo, se ele não puder usar os dados na mensagem).

Mensagens de relatório podem ser geradas a qualquer momento e podem chegar em uma fila quando o seu aplicativo não estiver esperando por elas.

Tipos de Mensagem de Relatório

Ao colocar uma mensagem em uma fila, é possível selecionar para receber:

- Uma *mensagem de relatório de exceção*. Isso é enviado em resposta a uma mensagem com o conjunto de sinalizações de exceção. Isso é gerado pelo agente do canal de mensagem (MCA) ou pelo aplicativo.
- Uma *mensagem de relatório de validação*. Isso indica que um aplicativo tentou recuperar uma mensagem que tinha atingido seu limite de expiração; a mensagem é marcada para ser descartada. Este tipo de relatório é gerado pelo gerenciador de filas.
- Uma *mensagem de relatório de confirmação de chegada (COA)*. Isso indica que a mensagem atingiu sua fila de destino. Ela é gerada pelo gerenciador de filas.
- Uma *mensagem de relatório de confirmação de entrega (COD)*. Isso indica que a mensagem foi recuperada por um aplicativo de recebimento. Ela é gerada pelo gerenciador de filas.
- Uma *mensagem de relatório de notificação de ação positiva (PAN)*. Isso indica que uma solicitação foi atendida com êxito (ou seja, a ação solicitada na mensagem foi executada com êxito). Este tipo de relatório é gerado pelo aplicativo.
- Uma *mensagem de relatório de notificação de ação negativa (NAN)*. Isso indica que uma solicitação não foi atendida com êxito (ou seja, a ação solicitada na mensagem não foi executada com êxito). Este tipo de relatório é gerado pelo aplicativo.

Nota: Cada tipo de mensagem de relatório contém um dos seguintes:

- A mensagem original inteira
- Os primeiros 100 bytes de dados na mensagem original
- Nenhum dado da mensagem original

É possível solicitar mais de um tipo de mensagem de relatório ao colocar uma mensagem em uma fila. Se forem selecionadas a mensagem de relatório de confirmação de entrega e as opções de mensagem de relatório de exceção, caso a mensagem falhe ao ser entregue, você receberá uma mensagem de relatório de exceção. No entanto, se for selecionada apenas a opção da mensagem de relatório de confirmação de entrega e a mensagem não for entregue, você não receberá uma mensagem de relatório de exceção.

As mensagens de relatório solicitadas, quando os critérios para gerar uma mensagem específica forem atendidos, serão as únicas que você receberá.

Opções da Mensagem de Relatório

É possível *descartar* uma mensagem depois que uma exceção tiver suscitado. Se for selecionada a opção descartar e tiver sido solicitada uma mensagem de relatório de exceção, a mensagem de relatório vai para o *ReplyToQ* e *ReplyToQMgr* e a mensagem original será descartada.

Nota: Um benefício disso é que é possível reduzir o número de mensagens indo para a fila de mensagens não entregues. No entanto, isso significa que seu aplicativo, a menos que envie apenas mensagens de datagrama, tem que lidar com mensagens retornadas. Quando uma mensagem de relatório de exceção é gerada, ela herda a persistência da mensagem original.

Se uma mensagem de relatório não puder ser entregue (se a fila estiver cheia, por exemplo), a mensagem de relatório será colocada na fila de mensagens não entregues.

Se desejar receber uma mensagem de relatório, especifique o nome de sua fila de resposta no campo *ReplyToQ*; caso contrário, o MQPUT ou MQPUT1 de sua mensagem original falhará com MQRC_MISSING_REPLY_TO_Q.

É possível usar outras opções de relatório no descritor de mensagens (MQMD) de uma mensagem para especificar o conteúdo dos campos *MsgId* e *CorrelId* de quaisquer mensagens de relatório que sejam criadas para a mensagem:

- É possível solicitar que o *MsgId* ou o *CorrelId* da mensagem original seja copiado ao campo *CorrelId* da mensagem de relatório. A ação padrão é copiar o identificador de mensagem. Use MQRO_COPY_MSG_ID_TO_CORRELID porque ele permite que o emissor de uma mensagem correlacione a mensagem de resposta ou relatório à mensagem original. O identificador de correlação da mensagem de resposta ou de relatório é idêntico ao identificador de mensagem da mensagem original.
- É possível solicitar que um novo *MsgId* seja gerado para a mensagem de relatório ou que o *MsgId* da mensagem original seja copiado para o campo *MsgId* da mensagem de relatório. A ação padrão é gerar um novo identificador de mensagem. Use MQRO_NEW_MSG_ID porque ele assegura que cada mensagem no sistema tenha um identificador de mensagem diferente e possa ser distinguida de maneira inequívoca de todas as outras mensagens no sistema.
- Aplicativos especializados podem precisar usar MQRO_PASS_MSG_ID ou MQRO_PASS_CORREL_ID. No entanto, é necessário designar o aplicativo que lerá as mensagens da fila para assegurar que funcionará corretamente quando, por exemplo, a fila contiver diversas mensagens com o mesmo identificador de mensagem.

  Aplicativos do servidor devem verificar as configurações dessas sinalizações na mensagem de solicitação e configurar os campos *MsgId* e *CorrelId* na mensagem de resposta ou de relatório conforme apropriado.

  Aplicativos que agem como intermediários entre um aplicativo do solicitante e um aplicativo do servidor não precisam verificar as configurações dessas sinalizações. Isso ocorre porque esses aplicativos geralmente precisam redirecionar a mensagem para o aplicativo do servidor com os campos *MsgId*, *CorrelId* e *Report* inalterados. Isto permite que o aplicativo do servidor copie o *MsgId* da mensagem original no campo *CorrelId* da mensagem de resposta.

Ao gerar um relatório sobre uma mensagem, os aplicativos do servidor devem testar para ver se alguma dessas opções foi configurada.

Para obter mais informações sobre como usar mensagens de relatório, consulte Relatório.

Para indicar a natureza do relatório, os gerenciadores de filas usam um intervalo de códigos de feedback. Eles colocam esses códigos no campo *Feedback* do descritor de mensagens de uma mensagem de relatório. Gerenciadores de filas também podem retornar códigos de razão MQI no campo *Feedback*.IBM WebSphere MQ define um intervalo de códigos de feedback para aplicativos usarem.

Para obter mais informações sobre feedback e códigos de razão, consulte Feedback.

Um exemplo de um programa que poderia usar um código de feedback é um que monitore as cargas de trabalho de outros programas que atendem a uma fila. Se houver mais de uma instância de um programa atendendo uma fila, o número de mensagens chegando na fila não mais justificará isso, tal programa pode enviar uma mensagem de relatório (com o código de feedback MQFB_QUIT) para um dos programas de atendimento para indicar que o programa deve terminar sua atividade. (Um programa de monitoramento poderia usar a chamada MQINQ para descobrir quantos programas estão atendendo uma fila.)

Relatórios e mensagens segmentadas

Não é suportado no WebSphere MQ para z/OS

Se uma mensagem for segmentada (consulte “Segmentação de mensagem” na página 266 para obter uma descrição de mensagens segmentadas) e você solicitar que relatórios sejam gerados, poderá receber mais relatórios do que receberia se a mensagem não tivesse sido segmentada.

Para relatórios gerados pelo WebSphere MQ

Se você segmentar suas mensagens ou permitir que o gerenciador de filas faça isso, há apenas um caso em que é possível esperar receber um único relatório para a mensagem inteira. Isso ocorre quando você tiver solicitado apenas relatórios COD e tiver especificado MQGMO_COMPLETE_MSG no aplicativo de get.

Em outros casos, seu aplicativo deve estar preparado para lidar com diversos relatórios; geralmente, um para cada segmento.

Nota: Se segmentar suas mensagens e precisar que somente os primeiros 100 bytes dos dados da mensagem original sejam retornados, mude a configuração das opções de relatório para solicitar relatórios sem dados para segmentos que têm um deslocamento de 100 ou mais. Se não fizer isso e deixar a configuração para que cada segmento solicite 100 bytes de dados e você recuperar as mensagens de relatório com um único MQGET especificando MQGMO_COMPLETE_MSG MQGET, os relatórios serão montados em uma grande mensagem contendo 100 bytes de dados lidos em cada deslocamento apropriado. Se isso acontecer, precisará de um buffer grande ou especificar MQGMO_ACCEPT_TRUNCATED_MSG.

Para relatórios gerados por aplicativos

Se seu aplicativo gerar relatórios, sempre copie os cabeçalhos WebSphere MQ que estão presentes no início dos dados da mensagem original para os dados da mensagem de relatório.

Em seguida, inclua nenhum, 100 bytes ou todos os dados da mensagem original (ou qualquer outra quantia que você normalmente incluiria) nos dados da mensagem de relatório.

É possível reconhecer os cabeçalhos do WebSphere MQ que devem ser copiados examinando os nomes de Formato sucessivos, começando com o MQMD e continuando através de quaisquer cabeçalhos presentes. Os seguintes nomes `Format` indicam estes cabeçalhos WebSphere MQ :

- MQMDE
- MQDLH
- MQXQH
- MQIIH
- MQH*

MQH* significa qualquer nome que começa com os caracteres MQH.

O nome `Format` ocorre em posições específicas para MQDLH e MQXQH, mas para os outros cabeçalhos do WebSphere MQ ele ocorre na mesma posição. O comprimento do cabeçalho está contido em um campo que também ocorre na mesma posição para os cabeçalhos MQMDE, MQIMS e MQH*.

Se estiver usando um MQMD Versão 1 e estiver relatando sobre um segmento, uma mensagem em um grupo ou uma mensagem para a qual a segmentação é permitida, os dados do relatório devem começar com um MQMDE. Configure o campo *`OriginalLength`* para o comprimento dos dados da mensagem original, excluindo os comprimento de quaisquer cabeçalhos do WebSphere MQ que você localizar.

Recuperando relatórios

Se você solicitar relatórios para COA ou COD, é possível solicitar que sejam remontados com MQGMO_COMPLETE_MSG.

Um MQGET com MQGMO_COMPLETE_MSG é satisfeito quando mensagens de relatório suficientes (de um único tipo, por exemplo COA e com o mesmo *`GroupId`*) estão presentes na fila para representar uma mensagem original completa. Isso é verdadeiro mesmo se as mensagens de relatório em si não contiverem dados originais completos; o campo *`OriginalLength`* em cada mensagem de relatório fornece o comprimento dos dados originais representados por essa mensagem de relatório, mesmo se os dados em si não estiverem presente.

É possível usar essa técnica mesmo se houver vários tipos de relatório diferentes presentes na fila (por exemplo, COA e COD), porque um MQGET com MQGMO_COMPLETE_MSG remonta mensagens de

relatório apenas se eles tiverem o mesmo código *Feedback* . No entanto, geralmente não é possível usar essa técnica para relatórios de exceções, pois, em geral, eles possuem códigos *Feedback* diferentes

É possível utilizar essa técnica para obter uma indicação positiva que a mensagem inteira tiver chegado. Entretanto, na maioria dos casos, é necessário atender a possibilidade de que alguns segmentos chegam enquanto outras podem gerar uma exceção (ou expiração, se você tiver permitido isso). Não é possível utilizar MQGMO_COMPLETE_MSG nesse caso, porque, em geral, você pode obter diferentes *Feedback* códigos para diferentes segmentos e, você pode obter mais de um relatório para um segmento. É possível, no entanto, utilizar MQGMO_ALL_SEGMENTS_AVAILABLE.

Para permitir isso, você pode precisar recuperar relatórios conforme eles chegam e construir uma imagem em seu aplicativo de o que aconteceu com a mensagem original. É possível usar o campo *GroupId* na mensagem de relatório para correlacionar relatórios com o*GroupId* da mensagem original e o campo *Feedback* para identificar o tipo de cada mensagem de relatório.. A maneira na qual você faz isso depende de seus requisitos de aplicativo.

Uma abordagem é conforme a seguir:

- Peça para relatórios COD e relatórios de exceção.
- Após um tempo específico, verifique se um conjunto completo de relatórios de confirmação de entrega foi recebido usando MQGMO_COMPLETE_MSG. Se sim, seu aplicativo sabe que a mensagem inteira foi processada.
- Se não, e relatórios de exceções relacionados a esta mensagem estiverem presentes, manipule o problema como para mensagens não segmentadas, mas se assegure de limpar os segmentos órfãos em algum ponto.
- Se houver segmentos para o qual não há relatos de nenhum tipo, os segmentos originais (ou os relatórios) podem estar esperando que um canal seja reconectado ou a rede pode estar sobrecarregada em algum ponto. Se nenhuma exceção em todos os relatórios for recebida (ou se você achar que aquelas que você possui podem ser temporárias apenas), você poderá optar por permitir que seu aplicativo esperar um pouco mais.

  Como antes, isso é similar às considerações que você tem quando lidar com mensagens não segmentadas, exceto que se deve também considerar a possibilidade de limpeza de segmentos órfão.

Se a mensagem original não é crítica (por exemplo, se for uma consulta ou uma mensagem que pode ser repetido mais tarde), configure um tempo de expiração para assegurar que os segmentos órfãos são removidos.

os gerenciadores de filas com versão anterior

Quando um relatório é gerado por um gerenciador de filas que suporta segmentação, mas é recebido em um gerenciador de filas que *não* suporta segmentação, a estrutura MQMDE (que identifica o *Offset* e o *OriginalLength* representado pelo relatório) é sempre incluída nos dados do relatório, além de zero, 100 bytes ou todos os dados originais na mensagem.

No entanto, se um segmento de uma mensagem passa através de um gerenciador de filas que não suporta segmentação, se um relatório for gerado, a estrutura MQMDE na mensagem original será tratada apenas como dados. Não é, portanto, incluído nos dados de relatório se zero bytes dos dados originais foram solicitados. Sem o MQMDE, a mensagem de relatório não pode ser útil.

Pedido de pelo menos 100 bytes de dados em relatórios se houver uma possibilidade de que a mensagem pode percorrer através de um gerenciador de filas de nível anterior.

Formato de informações de controle de mensagem e de dados de mensagens

O gerenciador de filas está interessado apenas no formato das informações de controle dentro de uma mensagem, enquanto que os aplicativos que tratam da mensagem estão interessados no formato das informações de controle e dos dados.

Formato de informações de controle de mensagem

As informações de controle nos campos de sequência de caracteres do descritor de mensagens devem estar no conjunto de caracteres usado pelo gerenciador de filas.

O atributo *CodedCharSetId* do objeto do gerenciador de filas define esse conjunto de caracteres. As informações de controle devem estar nesse conjunto de caracteres, pois, quando aplicativos transmitem mensagens de um gerenciador de filas para outro, os agentes do canal de mensagens que transmitem as mensagens usam o valor desse atributo para determinar qual conversão de dados executar.

Formato dos dados da mensagem

É possível especificar qualquer uma das seguintes ações:

- O formato dos dados do aplicativo
- O conjunto de caracteres dos dados de caractere
- O formato de dados numéricos

Para fazer isso, use estes campos:

Format
Isso indica para o receptor de uma mensagem o formato dos dados do aplicativo na mensagem.

Quando o gerenciador de filas cria uma mensagem, em algumas circunstâncias ele usa o campo *Format* para identificar o formato dessa mensagem. Por exemplo, quando um gerenciador de filas não pode entregar uma mensagem, ele coloca a mensagem em uma fila de devoluções (mensagem não entregue). Ele inclui um cabeçalho (contendo mais informações de controle) na mensagem, e muda o campo *Format* para mostrar isso.

O gerenciador de filas possui vários *formatos integrados* com nomes que começam com MQ, por exemplo, MQFMT_STRING. Se eles não atenderem às suas necessidades, será possível definir seus próprios formatos (*formatos definidos pelo usuário*), mas você não deverá usar nomes que comecem com MQ para eles.

Quando você criar e usar seus próprios formatos, deverá gravar uma saída de conversão de dados para suportar um programa obtendo a mensagem usando MQGMO_CONVERT.

CodedCharSetId
Isso define o conjunto de caracteres de dados de caracteres na mensagem. Se desejar configurar esse conjunto de caracteres como aquele do gerenciador de filas, poderá configurar esse campo como a constante MQCCSI_Q_MGR ou MQCCSI_INHERIT.

Quando você obtiver uma mensagem de uma fila, compare o valor do campo *CodedCharSetId* com o valor que seu aplicativo está esperando. Se os dois valores forem diferentes, talvez seja necessário converter os dados de caracteres na mensagem ou usar uma saída de mensagem de conversão de dados, se houver uma disponível.

Encoding
Isto descreve o formato dos dados de mensagens numéricas que contêm números inteiros binários, números inteiros decimais compactados e números de vírgula flutuante. Ele é, geralmente, codificado de acordo com a máquina específica na qual o gerenciador de filas está em execução.

Ao colocar uma mensagem em uma fila, você, geralmente, especifica a constante MQENC_NATIVE no campo *Encoding*. Isso significa que a codificação dos dados de mensagens é a mesma que a da máquina na qual seu aplicativo está em execução.

Quando você obtiver uma mensagem de uma fila, compare o valor do campo *Encoding* no descritor de mensagens com o valor da constante MQENC_NATIVE em sua máquina. Se os dois valores forem diferentes, talvez seja necessário converter os dados numéricos na mensagem ou usar uma saída de mensagem de conversão de dados, se houver uma disponível.

Conversão de Dados do Aplicativo

Dados do aplicativo podem precisar ser convertidos para o conjunto de caracteres e codificação requeridos por outro aplicativo em que diferentes plataformas são de interesse.

Ele pode ser convertido no gerenciador de filas de envio ou no gerenciador de filas de recebimento. Se a biblioteca de formatos integrados não atende suas necessidades, é possível definir a sua própria. O tipo de conversão depende do formato da mensagem que estiver especificado no campo formato do descritor de mensagens, MQMD.

Nota: Mensagens com MQFMT_NONE especificado não são convertidas.

Conversão no gerenciador de filas de envio

Configure o atributo do canal CONVERT para YES se for necessário que o agente do canal da mensagem enviada (MCA) converta os dados do aplicativo.

A conversão é executada no gerenciador de filas de envio para determinados formatos integrados e para formatos definidos pelo usuário se uma saída de usuário adequada for fornecida.

Formatos integrados

Isso inclui:

- As mensagens que sejam inteiramente caracteres (usando o nome de formato MQFMT_STRING)
- WebSphere MQ mensagens definidas, por exemplo, Formatos de Comando Programáveis

  WebSphere MQ usa mensagens de Formato de Comando Programável para mensagens de administração e eventos (o nome do formato usado é MQFMT_ADMIN neste caso). É possível usar o mesmo formato (usando o nome do formato MQFMT_PCF) para suas próprias mensagens e tirar proveito da conversão de dados integrados.

Os formatos integrados do gerenciador de filas todos possuem nomes que começam com MQFMT. Eles são listados e descritos em Formato.

Formatos definidos pelo aplicativo

Para formatos definidos pelo usuário, a conversão de dados do aplicativo deve ser executada por um programa de saída de conversão de dados (para obter informações adicionais, consulte “Escrevendo saídas de conversão de dados” na página 421). Em um ambiente de cliente/servidor, a saída é carregada no servidor e a conversão ocorre ali.

Conversão no gerenciador de filas de recebimento

Dados da mensagem do aplicativo podem ser convertidos pelo gerenciador de filas de recebimento para ambos formatos, integrados e definidos pelo usuário.

A conversão é desempenhada durante o processamento de uma chamada MQGET se for especificada a opção MQGMO_CONVERT. Para obter detalhes, consulte as Opções

Conjuntos de caracteres codificados

Os produtos WebSphere MQ suportam os conjuntos de caracteres codificados fornecidos pelo sistema operacional subjacente.

Ao criar um gerenciador de filas, o ID do conjunto de caracteres codificados do gerenciador de filas (CCSID) usado será baseado naquele do ambiente subjacente. Se esta for uma página de códigos combinada, o WebSphere MQ usará a parte SBCS da página de código combinada como o CCSID do gerenciador de filas..

Para conversão de dados gerais, se o sistema operacional subjacente suportar páginas de código DBCS, o WebSphere MQ poderá usá-lo.

Consulte a documentação de seu sistema operacional para obter detalhes dos conjuntos de caracteres codificados que ele suporta.

É necessário considerar a conversão de dados do aplicativo, os nomes de formato, e saídas de usuário ao gravar aplicativos que abrangem várias plataformas. Consulte “Escrevendo saídas de conversão de dados” na página 421 para obter informações sobre como chamar e gravar saídas de conversão de dados.

Prioridades de mensagens

Configure a prioridade de uma mensagem (no campo *Priority* da estrutura MQMD) quando você colocar a mensagem em uma fila. É possível configurar um valor numérico para a prioridade ou deixar a mensagem obter a prioridade padrão da fila.

O atributo *MsgDeliverySequence* da fila determina se as mensagens na fila são armazenadas na sequência FIFO (first in, first out) ou em FIFO dentro da sequência de prioridade. Se este atributo estiver configurado como MQMDS_PRIORITY, as mensagens serão enfileiradas com a prioridade especificada no campo *Priority* de seus descritores de mensagens; mas se ele estiver configurado como MQMDS_FIFO, as mensagens serão enfileiradas com a prioridade padrão da fila. As mensagens de prioridade igual são armazenadas na fila em ordem de chegada.

O atributo *DefPriority* de uma fila configura o valor da prioridade padrão para mensagens que estão sendo colocadas nessa fila. Esse valor é configurado quando a fila é criada, mas pode ser mudado posteriormente. Filas de alias e definições locais de filas remotas podem ter prioridades padrão diferentes das filas de base para a qual elas resolvem. Se houver mais de uma definição de fila no caminho de resolução (consulte “Resolução do Nome” na página 220), a prioridade padrão será obtida do valor (no momento da operação put) do atributo *DefPriority* da fila especificada no comando open.

O valor do atributo *MaxPriority* do gerenciador de filas é a prioridade máxima que pode ser designada a uma mensagem processada por esse gerenciador de filas. Não é possível mudar o valor desse atributo. No WebSphere MQ, o atributo tem o valor 9; é possível criar mensagens com prioridades entre 0 (o mais baixo) e 9 (o mais alto).

Propriedades da Mensagem

Use propriedades de mensagem para permitir que um aplicativo selecione mensagens para processar ou recuperar informações sobre uma mensagem sem acessar cabeçalhos MQMD ou MQRFH2. Elas também facilitam a comunicação entre aplicativos WebSphere MQ e JMS.

Uma propriedade de mensagem é dados associados a uma mensagem, consistindo em um nome textual e um valor de um tipo específico. Propriedades de mensagens são usadas pelos seletores de mensagens para filtrar publicações para tópicos ou para obter seletivamente as mensagens das filas. Propriedades de mensagens podem ser usadas para incluir dados de negócios ou informações de estado sem precisar armazená-las nos dados do aplicativo. Os aplicativos não precisam acessar dados no MQ Message Descriptor (MQMD) ou nos cabeçalhos MQRFH2 porque os campos nessas estruturas de dados podem ser acessados como propriedades de mensagem usando chamadas de função Message Queue Interface (MQI).

O uso de propriedades de mensagem no WebSphere MQ imita o uso de propriedades no JMS. Isso significa que é possível configurar propriedades em um aplicativo JMS e recuperá-las em um aplicativo processual do WebSphere MQ ou o contrário. Para disponibilizar uma propriedade para um aplicativo JMS, designe a ele o prefixo "usr"; ele estará disponível (sem o prefixo) como uma propriedade do usuário da mensagem JMS. Por exemplo, a propriedade do WebSphere MQ *usr.myproperty* (uma sequência de caracteres) é acessível para um aplicativo JMS usando a chamada JMS `message.getStringProperty('myproperty')`. Observe que os aplicativos JMS não podem acessar propriedades com o prefixo "usr" se contiverem dois ou mais U+002E (".") caracteres. Uma propriedade sem prefixo e nenhum caractere U+002E (".") é tratada como se tivesse o prefixo "usr". Por outro lado, uma propriedade do usuário configurada em um aplicativo JMS pode ser acessada em um aplicativo WebSphere MQ incluindo o "usr." no nome da propriedade consultada em uma chamada MQINQMP.

Propriedades de mensagens e comprimento de mensagens

Use o atributo do gerenciador de filas *MaxPropertiesComprimento* para controlar o tamanho das propriedades que podem fluir com qualquer mensagem em um gerenciador de filas do WebSphere MQ

Em geral, ao usar MQSETMP para configurar as propriedades, o tamanho de uma propriedade é o comprimento do nome da propriedade em bytes, mais o comprimento do valor da propriedade em bytes, conforme passado na chamada MQSETMP. É possível que o conjunto de caracteres do nome da propriedade e o valor da propriedade mudem durante a transmissão da mensagem para seu destino porque podem ser convertidos em Unicode; neste caso o tamanho da propriedade pode ser mudado.

Em uma chamada MQPUT ou MQPUT1, as propriedades da mensagem não contam para o comprimento da mensagem para a fila e o gerenciador de filas, mas contam para o comprimento das propriedades conforme observado pelo gerenciador de filas (se elas foram configuradas usando a propriedade de mensagem de chamadas MQI ou não).

Se o tamanho das propriedades exceder o comprimento máximo de propriedades, a mensagem será rejeitada com MQRC_PROPERTIES_TOO_BIG. Como o tamanho das propriedades depende de sua representação, é necessário configurar o comprimento máximo das propriedades em um nível bruto.

É possível para um aplicativo para colocar com sucesso uma mensagem com um buffer que seja maior que o valor de *MaxMsgLength*, se o buffer incluir propriedades. Isso ocorre porque, mesmo quando representadas como elementos MQRFH2, as propriedades da mensagem não contam para o comprimento da mensagem. Os campos de cabeçalho MQRFH2 incluem o comprimento de propriedades apenas se uma ou mais pastas estiverem contidas e cada pasta no cabeçalho contiver propriedades. Se uma ou mais pastas estiverem contidas no cabeçalho MQRFH2 e qualquer pasta não contiver propriedades, os campos de cabeçalho MQRFH2 contam para o comprimento da mensagem.

Em uma chamada MQGET, as propriedades da mensagem não contam para o comprimento da mensagem com relação à fila e ao gerenciador de filas. No entanto, como as propriedades são contadas separadamente, é possível que o buffer retornado por uma chamada MQGET seja maior que o valor do atributo *MaxMsgLength*.

Não faça seus aplicativos consultarem o valor de *MaxMsgLength* e, em seguida, alocar um buffer desse tamanho antes da chamada MQGET; em vez disso, aloque um buffer que você considere grande o suficiente. Se MQGET falhar, aloque um buffer guiado pelo tamanho do parâmetro *DataLength*.

O parâmetro *DataLength* da chamada MQGET retorna o comprimento em bytes dos dados do aplicativo e quaisquer propriedades retornadas no buffer que você forneceu, se um identificador de mensagem não for especificado na estrutura MQGMO.

O parâmetro *Buffer* da chamada MQPUT contém os dados da mensagem do aplicativo a ser enviada e quaisquer propriedades representadas nos dados da mensagem.

Ao fluir para um gerenciador de fila anterior à Versão 7.0 do produto, as propriedades da mensagem, exceto aquelas no descritor de mensagens, contam para o comprimento da mensagem. Portanto, você deve aumentar o valor do atributo *MaxMsgLength* de canais que vão para um sistema anterior à Versão 7.0 conforme necessário, para compensar o fato de que mais dados podem ser enviados para cada mensagem. Como alternativa, é possível diminuir a fila ou o gerenciador de filas *MaxMsgLength*, para que o nível geral de dados que estão sendo enviados ao redor do sistema permaneça o mesmo.

Há um limite de comprimento de 100 MB para propriedades de mensagem, excluindo o descritor de mensagens ou extensão para cada mensagem.

O tamanho de uma propriedade em sua representação interna é o comprimento do nome, mais o tamanho de seu valor, além de alguns dados de controle para a propriedade. Há também alguns dados de controle para o conjunto de propriedades após uma propriedade ser incluída na mensagem.

Nomes de propriedades

Um nome da propriedade é uma sequência de caracteres. Determinadas restrições se aplicam a seu comprimento e ao conjunto de caracteres que podem ser usados.

Um nome da propriedade é uma sequência de caracteres com distinção entre maiúsculas e minúsculas, limitado a +4095 caracteres, a menos que restrito de outra forma pelo contexto. Este limite está contido na constante MQ_MAX_PROPERTY_NAME_LENGTH.

Se você exceder esse comprimento máximo ao usar uma chamada MQI propriedade de mensagem, a chamada falhará com o código de razão MQRC_PROPERTY_NAME_LENGTH_ERR.

Como não há comprimento máximo de nome de propriedade no JMS, é possível para um aplicativo JMS configurar um nome de propriedade JMS válido que não seja um nome de propriedade do WebSphere MQ válido quando armazenado em uma estrutura MQRFH2

Neste caso, quando analisado, apenas os primeiros 4095 caracteres do nome da propriedade são usados; os caracteres a seguir são truncados. Isso poderá fazer com que um aplicativo que está usando seletores falhe em corresponder a uma sequência de seleção ou em corresponder a uma sequência quando não esperado, uma vez que mais de uma propriedade pode ser truncada para o mesmo nome. Quando um nome de propriedade é truncado, o WebSphereMQ emite uma mensagem do log de erros

Todos os nomes de propriedades devem seguir as regras definidas pela Java Language Specification for Java Identifiers, com a exceção de que o caractere Unicode U+002E (.) é permitido como parte do nome-mas não o início. As regras para Identificadores Java são iguais àquelas contidas na especificação JMS para nomes de propriedades

Os caracteres de espaço em branco e operadores de comparação são proibidos. Os nulos integrados são permitidos em um nome de propriedade, mas não é recomendado. Se você usar os nulos integrados, isso impede o uso da constante MQVS_NULL_TERMINATED quando usada com a estrutura MQCHARV para especificar sequências de comprimento variável.

Mantenha os nomes de propriedade simples, porque os aplicativos podem selecionar as mensagens com base nos nomes de propriedade e a conversão entre o conjunto de caracteres do nome e do seletor pode fazer com que a seleção falhe inesperadamente.

WebSphere MQ nomes de propriedades usam o caractere U+002E (.) para agrupamento lógico de propriedades. Isso divide o namespace para as propriedades. Propriedades com os prefixos a seguir, em qualquer combinação de minúsculas ou maiúsculas são reservadas para uso pelo produto:

- `mcd`
- `jms`
- `usr`
- `mq`
- `sib`
- `wmq`
- `Root`
- `Body`
- `Properties`

Uma boa maneira de evitar conflitos de nomes é assegurar que todos os aplicativos prefixem suas propriedades de mensagem com seu nome de domínio da Internet. Por exemplo, se você estiver desenvolvendo um aplicativo usando o nome de domínio "`ourcompany.com`", poderá nomear todas as propriedades com o prefixo "`com.ourcompany`". Essa convenção de nomenclatura também permite uma fácil seleção de propriedades; por exemplo, um aplicativo pode consultar todas as propriedades de mensagem que iniciam "`com.ourcompany.%`"..

Consulte Restrições de nomes de propriedades para obter informações adicionais sobre o uso de nomes de propriedades.

Restrições de nome da propriedade
Ao nomear uma propriedade, deve-se observar certas regras.

As restrições a seguir se aplicam aos nomes de propriedade:

1. Uma propriedade não deve começar com as seguintes sequências:
   - "JMS"-reservado para uso pelas classes WebSphere MQ para JMS.
   - "usr.JMS"-não válido.

   As únicas exceções são as seguintes propriedades que fornecem sinônimos para propriedades JMS:

Propriedade	Sinônimo para
JMSCorrelationID	Raiz .MQMD.CorrelId ou jms.Cid
JMSDeliveryMode	Raiz .MQMD.Persistence ou jms.Dlv
JMSDestination	jms.Dst
JMSExpiration	Raiz .MQMD.Expiry ou jms.Exp
JMSMessageID	Raiz .MQMD.MsgId
JMSPriority	Raiz .MQMD.Priority ou jms.Pri
JMSRedelivered	Raiz .MQMD.BackoutCount
JMSReplyTo (uma sequência codificada como um URI)	Raiz .MQMD.ReplyToQ ou Root .MQMD.ReplyToQMgr ou jms.Rto
JMSTimestamp	Raiz .MQMD.PutDate ou Raiz .MQMD.PutTime ou jms.Tms
JMSType	mcd.Type ou mcd.Set ou mcd.Fmt
JMSXAppID	Raiz .MQMD.PutApplName
JMSXDeliveryCount	Raiz .MQMD.BackoutCount
JMSXGroupID	Raiz .MQMD.GroupId ou jms.Gid
JMSXGroupSeq	Raiz .MQMD.MsgSeqNumber ou jms.Seq
JMSXUserID	Raiz .MQMD.UserIdentifier

Esses sinônimos permitem que um aplicativo MQI acesse propriedades JMS de forma semelhante às classes WebSphere MQ para o aplicativo cliente JMS. Dessas propriedades, somente JMSCorrelationID, JMSReplyTo, JMSType, JMSXGroupID e JMSXGroupSeq podem ser configurados usando o MQI.

Observe que as propriedades JMS_IBM_ * disponíveis nas classes do WebSphere MQ para JMS não estão disponíveis usando o MQI. Os campos que as propriedades JMS_IBM_* referenciam podem ser acessados de outras maneiras por aplicativos MQI.

2. Uma propriedade não deve ser chamada, em qualquer combinação de letras maiúsculas ou minúsculas, "NULL", "TRUE", "FALSE", "NOT", "AND", "OR", "BETWEEN", "LIKE", "IN", "IS" e "ESCAPE". Estes são os nomes de palavras-chave SQL usadas em sequências de seleção.
3. Um nome da propriedade que inicia com "mq " em qualquer combinação de letras minúsculas ou maiúsculas e não iniciar "mq_usr" pode conter apenas um "." caractere (U+002E). Vários "." caracteres não são permitidos em propriedades com esses prefixos.
4. Dois "." caracteres devem conter outros caracteres entre eles; não é possível ter um ponto vazio na hierarquia. Semelhantemente, nome de propriedade não pode terminar com um "." "."
5. Se um aplicativo configurar a propriedade "a.b" e, em seguida, a propriedade "a.b.c", não será claro se na hierarquia "b" contém um valor ou outro agrupamento lógico. Tal hierarquia é "conteúdo misto" e este não é suportado. Configurando uma propriedade que causa com que o conteúdo misto não seja permitido.

Essas restrições do são impostas pelo mecanismo de validação conforme a seguir:

- Os nomes de propriedades são validados ao configurar uma propriedade usando a chamada MQSETMP – Configurar propriedade de mensagem , se a validação tiver sido solicitada quando a manipulação de mensagem foi criada Se uma tentativa de validar uma propriedade for realizada e falhar devido a um erro na especificação do nome da propriedade, o código de conclusão será MQCC_FAILED com a razão:
  - MQRC_PROPERTY_NAME_ERROR para razões 1-4.
  - MQRC_MIXED_CONTENT_NOT_ALLOWED para a razão 5.

- Os nomes de propriedades especificadas diretamente como elementos MQRFH2 não são garantidas ao ser validadas pela chamada MQPUT.

Campos do descritor de mensagens como propriedades
A maioria dos campos do descritor de mensagens pode ser tratada como propriedades. O nome da propriedade é construído pela inclusão de um prefixo no nome do campo descritor de mensagens.

Se um aplicativo MQI deseja identificar uma propriedade de mensagem contida em um campo do descritor de mensagens, por exemplo, em uma sequência do seletor ou usando as APIs de propriedade da mensagem, use a seguinte sintaxe:

Nome da Propriedade	Campo do descritor de mensagens
Root.MQMD.<Field>	<Field>

Especifique `<Field>` com a mesma letra que para os campos de estrutura do MQMD na declaração de linguagem C. Por exemplo, o nome de propriedade `Root.MQMD.AccountingToken` acessa o campo `AccountingToken` do descritor de mensagens.

Os campos `StrucId` e `Version` do descritor de mensagens não estão acessíveis usando a sintaxe mostrada.

Campos do descritor de mensagens nunca são representados em um cabeçalho MQRFH2 como para outras propriedades.

Se os dados da mensagem iniciarem com um MQMDE que é aceito pelo gerenciador de filas, os campos MQMDE poderão ser acessados usando a notação `Root.MQMD.<Field>` descrita. Neste caso, os campos MQMDE são tratados como parte, logicamente, do MQMD a partir de uma perspectiva de propriedades. Consulte a seção "MQMDE especificado em chamadas MQPUT e MQPUT1 " no Visão Geral de MQMDE.

Tipos e valores de dados de propriedades
Uma propriedade pode ser um booleano, uma sequência de bytes, uma sequência de caracteres ou um número de vírgula flutuante ou inteiro. A propriedade pode armazenar qualquer valor válido no intervalo do tipo de dados, a menos que restringido de outra forma pelo contexto.

O tipo de dados de um valor de propriedade deve ser um dos valores a seguir:

- MQBOOL
- MQBYTE[ ]
- MQCHAR[ ]
- MQFLOAT32
- MQFLOAT64
- MQINT8
- MQINT16
- MQINT32
- MQINT64

Uma propriedade pode existir, mas não têm nenhum valor definido; é uma propriedade nula. Uma propriedade nula é diferente de uma propriedade de byte (MQBYTE[ ]) ou propriedade sequência de caracteres (MQCHAR[ ]), pois tem um valor definido, mas vazio, ou seja, um valor de comprimento zero.

A cadeia de bytes não é um tipo de dados de propriedade válido no JMS ou no XMS Aconselha-se não usar propriedades de sequência de bytes na pasta <usr>.

Selecionando mensagens nas filas

É possível selecionar mensagens nas filas usando os campos MsgId e CorrelId em uma chamada MQGET ou usando uma SelectionString em uma chamada MQOPEN ou MQSUB.

Seletores

Um seletor de mensagem é uma sequência de comprimento variável usada por um aplicativo para registrar seu interesse apenas naquelas mensagens que têm propriedades que satisfazem a consulta Linguagem de Consulta Estruturada (SQL) que a sequência de seleção representa.

Seleção usando as chamadas de função MQSUB e MQOPEN

É possível usar *SelectionString*, que é uma estrutura de tipo MQCHARV, para fazer seleções usando as chamadas MQSUB e MQOPEN.

A estrutura de *SelectionString* é usada para passar uma sequência de seleção de comprimento variável para o gerenciador de filas.

O CCSID associado à sequência do seletor é configurado por meio do campo VSCCSID da estrutura MQCHARV. O valor usado deve ser um CCSID que é suportado para sequências de seletor. Consulte Conversão de página de códigos para obter uma lista de páginas de códigos suportadas.

Especificar um CCSID para o qual não há conversão Unicode suportada pelo WebSphere MQ , resulta em um erro de MQRC_SOURCE_CCSID_ERROR. Esse erro é retornado no momento em que o seletor é apresentado ao gerenciador de filas, isso é, na chamada MQSUB, MQOPEN ou MQPUT1.

O valor padrão para o campo *VSCCSID* é MQCCSI_APPL, o que indica que o CCSID da sequência de seleção é igual ao CCSID do gerenciador de filas ou ao CCSID do cliente, se estiver conectado por meio de um cliente. A constante MQCCSI_APPL pode, no entanto, ser substituída por um aplicativo que a esteja redefinindo antes da compilação.

Se o seletor MQCHARV representa uma sequência NULL, nenhuma seleção ocorre para esse consumidor de mensagens e as mensagens serão entregues como se um seletor não tivesse sido usado.

O comprimento máximo de uma sequência de seleção é limitado somente por aquilo que pode ser descrito pelo campo MQCHARV *VSLength*.

SelectionString é retornado na saída de uma chamada MQSUB usando a opção de assinatura MQSO_RESUME, se você tiver fornecido um buffer e houver um comprimento de buffer positivo em VSBufSize. Se você não fornecer um buffer, somente o comprimento da sequência de seleção será retornado no campo VSLength do MQCHARV. Se o buffer fornecido for menor que o espaço necessário para retornar o campo, somente VSBufSize bytes serão retornados no buffer fornecido.

Um aplicativo não pode mudar uma sequência de seleção sem antes fechar o identificador para a fila (para MQOPEN) ou assinatura (para MQSUB). Uma nova seleção de sequência pode ser especificada em uma chamada MQOPEN ou MQSUB subsequente.

MQOPEN
: Use MQCLOSE para fechar o identificador aberto, em seguida, especifique uma sequência de seleção nova em uma chamada MQOPEN subsequente.

MQSUB
: Use MQCLOSE para fechar o identificador de assinatura retornada (hSub) e, em seguida, especifique uma sequência de seleção nova em uma chamada MQSUB subsequentes.

Figura 3 na página 24 mostra o processo de seleção usando a chamada MQSUB.

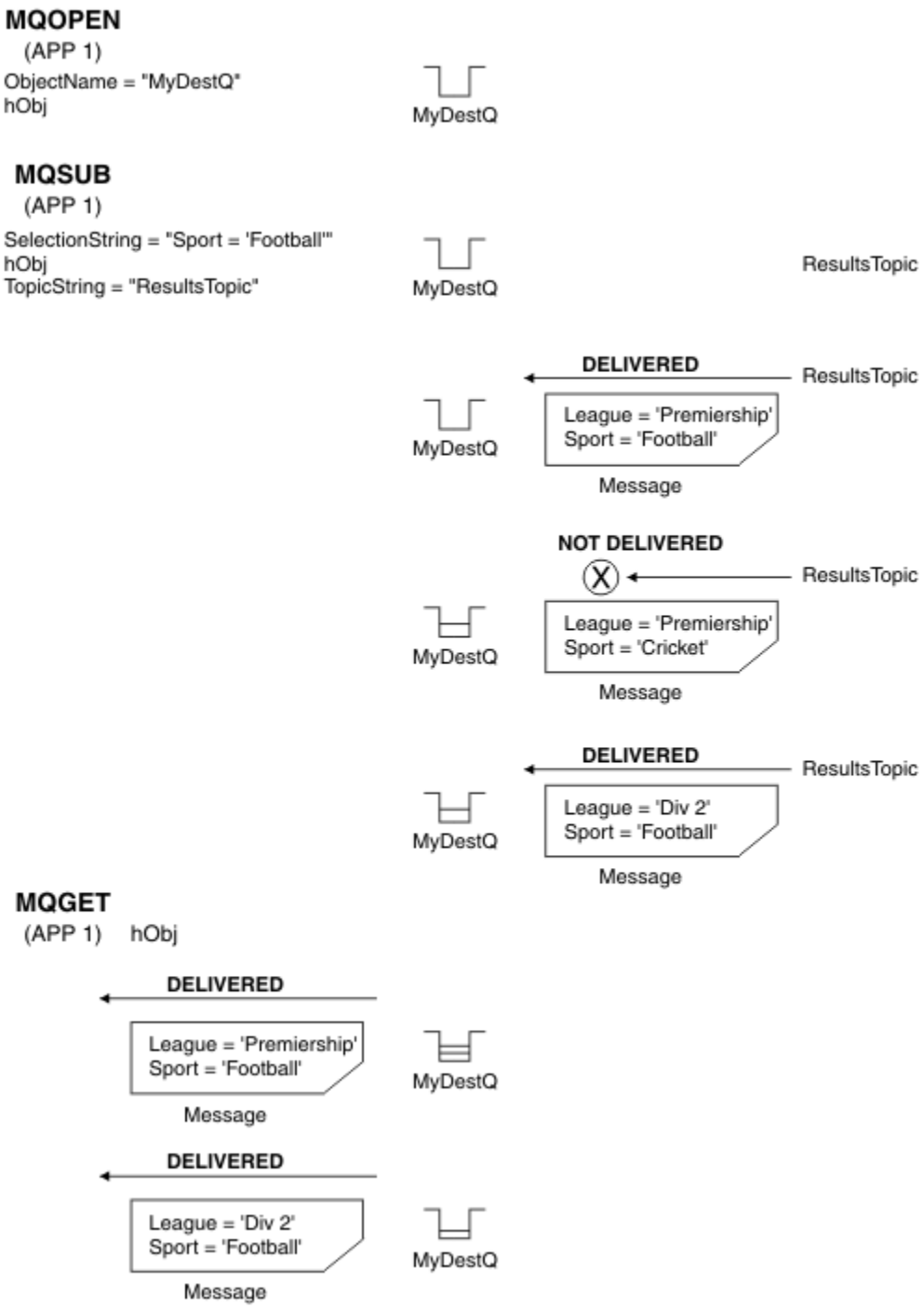


Figura 3. Seleção usando a chamada MQSUB

Um seletor pode ser passado na chamada para MQSUB usando o campo *SelectionString* na estrutura MQSD. O efeito de passar um seletor no MQSUB é que somente as mensagens publicadas para o tópico que está sendo assinado, que correspondem a uma sequência de seleção fornecida, serão disponibilizadas na fila de destino.

Figura 4 na página 25 mostra o processo de seleção usando a chamada MQOPEN.

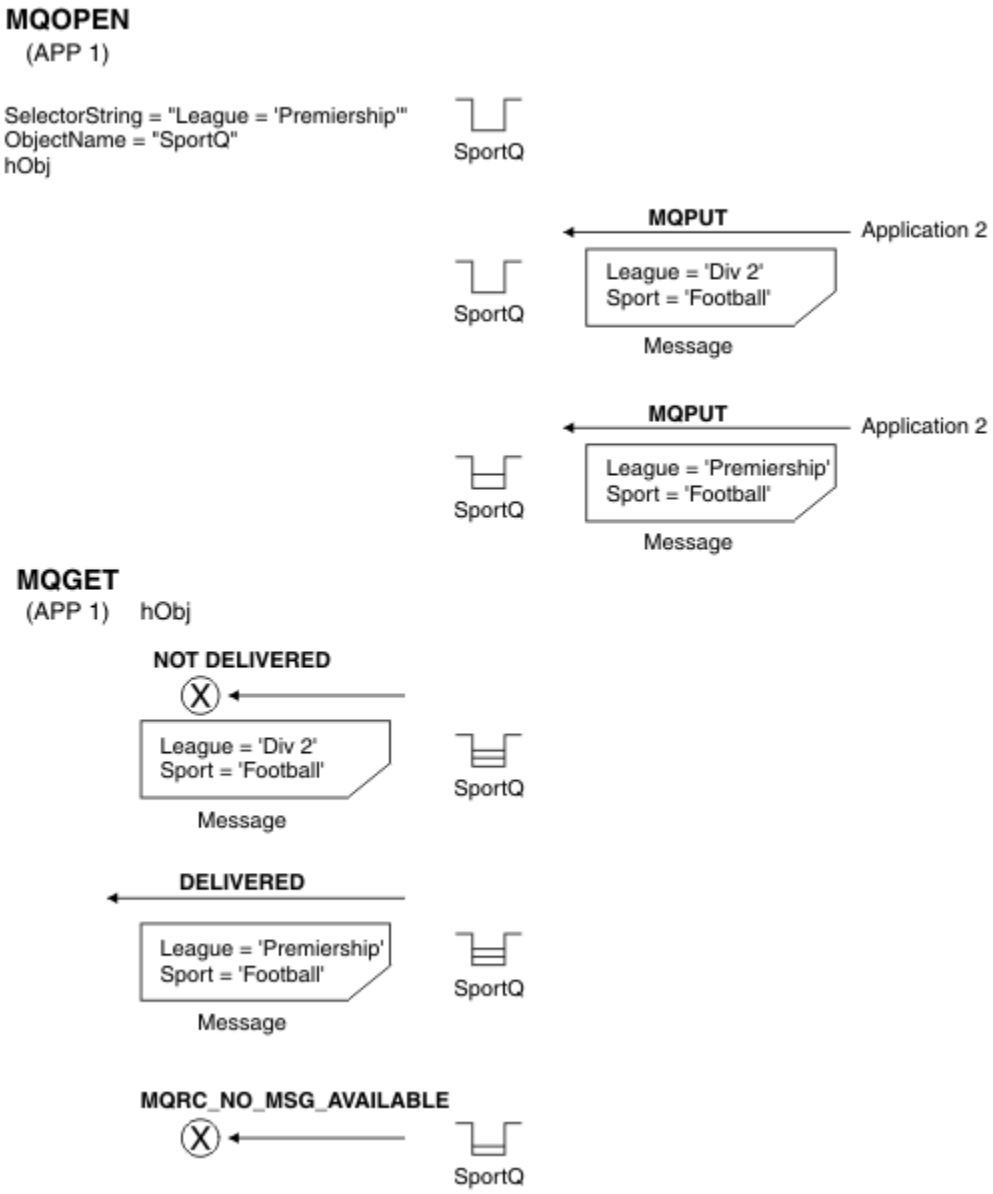


Figura 4. Seleção usando a chamada MQOPEN

Um seletor pode ser passado na chamada para MQOPEN usando o campo *SelectionString* na estrutura MQOD. O efeito de passar um seletor na chamada MQOPEN é que somente as mensagens na fila aberta, que correspondem a um seletor, serão entregues ao consumidor de mensagens.

O uso principal para o seletor na chamada MQOPEN é para o caso ponto a ponto em que um aplicativo pode optar por receber em uma fila somente as mensagens que correspondem a um seletor. O exemplo anterior mostra um cenário simples em que duas mensagens são colocadas em uma fila aberta por MQOPEN, mas somente uma é recebida pelo aplicativo que a está obtendo, uma vez que ele é o único que corresponda a um seletor.

Observe que chamadas MQGET subsequentes resultam em MQRC_NO_MSG_AVAILABLE, pois nenhuma mensagem adicional existe na fila que corresponde ao seletor fornecido.

Comportamento de seleção
Visão geral do comportamento de seleção IBM WebSphere MQ .

Os campos em uma estrutura MQMDE são considerados como propriedades de mensagem para as propriedades do descritor de mensagens correspondentes se o MQMD:

- Tiver o formato MQFMT_MD_EXTENSION
- For seguido imediatamente por uma estrutura MQMDE válida
- For uma versão ou contiver a versão padrão de dois campos somente

É possível que uma sequência de seleção seja resolvida para TRUE ou FALSE antes que qualquer correspondência com as propriedades de mensagem ocorra., Por exemplo, pode ser o caso se a sequência de seleção estiver configurada como "TRUE <>FALSE" Essa avaliação inicial é garantida que ocorre somente quando não houver referências de propriedades de mensagem na sequência de seleção.

Se uma sequência de seleção for resolvida para TRUE antes de quaisquer propriedades de mensagem serem consideradas, todas as mensagens publicadas no tópico assinado pelo consumidor serão entregues. Se uma sequência de seleção for resolvida para FALSE antes de quaisquer propriedades de mensagem serem consideradas, um código de razão MQRC_SELECTOR_ALWAYS_FALSE e o código de conclusão MQCC_FAILED serão retornados na chamada de função que apresentou o seletor.

Mesmo se uma mensagem não contiver nenhuma propriedade de mensagem (além de propriedades de cabeçalho), ela ainda pode estar elegível para seleção. Se uma sequência de seleção fizer referência a uma propriedade de mensagem que não existe, essa propriedade será assumida como tendo o valor NULL ou 'Unknown'.

Por exemplo, uma mensagem ainda pode satisfazer uma sequência de seleção como `'Color IS NULL'`, em que `'Color'` não existe como uma propriedade de mensagem na mensagem

A seleção pode ser executada somente nas propriedades associadas a uma mensagem, não à própria mensagem, a menos que um provedor de seleção de mensagem estendida esteja disponível. A seleção poderá ser executada na carga útil da mensagem apenas se um provedor de seleção de mensagem estendida estiver disponível.

Cada propriedade de mensagem tem um tipo associado a ele. Ao executar uma seleção, deve-se assegurar que os valores usados em expressões para testar as propriedades de mensagens são do tipo correto. Se ocorrer uma incompatibilidade de tipos, a expressão em questão será resolvida para FALSE.

É de sua responsabilidade assegurar que a sequência de seleção e as propriedades de mensagem usem tipos compatíveis.

Critérios de seleção continuam a ser aplicados em nome de assinantes duráveis inativos, de modo que somente as mensagens que correspondem à sequência de seleção que foi originalmente fornecida serão mantidas.

Sequências de seleção não podem ser mudadas quando uma assinatura durável é continuada com alteração (MQSO_ALTER). Se uma sequência de seleção diferente for apresentada quando um assinante durável continuar a atividade, então, MQRC_SELECTOR_NOT_ALTERABLE será retornado ao aplicativo.

Os aplicativos recebem um código de retorno de MQRC_NO_MSG_AVAILABLE se não houver mensagem em uma fila que atenda aos critérios de seleção.

Se um aplicativo tiver especificado uma sequência de seleção contendo valores de propriedades, somente aquelas mensagens que contêm propriedades correspondentes serão elegíveis para seleção. Por exemplo, um assinante especifica uma sequência de caracteres de seleção de "a = 3" e uma mensagem é publicada contendo nenhuma propriedade ou propriedades em que 'a' não existe ou não é igual a 3. O assinante não recebe essa mensagem para sua fila de destino.

Desempenho de mensagens

Selecionar mensagens de uma fila requer que o IBM WebSphere MQ inspecione sequencialmente cada mensagem na fila. As mensagens são inspecionadas até que uma mensagem seja localizada que corresponda aos critérios de seleção ou não haja mais mensagens para examinar. Portanto, o desempenho do sistema de mensagens será prejudicado se a seleção de mensagem for usada em filas profundas.

Para otimizar a seleção de mensagens em filas profundas quando a seleção é baseada em JMSCorrelationID ou JMSMessageID, use uma cadeia de seleção no formato `JMSCorrelationID = ...` ou `JMSMessageID = ...` e referencie apenas uma propriedade..

Esse método oferece uma melhoria significativa no desempenho para seleção em JMSCorrelationID e oferece uma melhoria de desempenho marginal para JMSMessageID.

Usando seletores complexos

Os seletores podem conter vários componentes, por exemplo:

a e b ou c e d ou e e f ou g e h ou i e j ... ou y e z

O uso de tais seletores complexos pode ter sérias implicações no desempenho e requisitos de recurso excessivos. Dessa forma, o IBM WebSphere MQ protegerá o sistema deixando de processar seletores muito complexos que poderiam resultar em uma falta de recursos do sistema. A proteção pode ocorrer após aproximadamente 100 testes em algumas plataformas para que os seletores que se aproximam desse número de componentes possam ver falhas. Recomenda-se que o uso de seletores com muitos componentes seja completamente tentado e testado nas plataformas apropriadas para assegurar que os limites de proteção não sejam atingidos.

O desempenho e a complexidade de seletores podem ser melhorados simplificando-os usando parênteses adicionais para combinar componentes. Por exemplo:

(a e b ou c e d) ou (e e f ou g e h) ou (i e j) ...

Conceitos relacionados

Sintaxe do seletor de mensagem
Um seletor de mensagem do WebSphere MQ é uma sequência com sintaxe baseada em um subconjunto da sintaxe de expressão condicional SQL92 .

Selecionando no conteúdo de uma mensagem
É possível assinar com base em uma seleção de conteúdo de carga útil da mensagem (também conhecido como filtragem do conteúdo), mas a decisão sobre quais mensagens devem ser entregues para tal assinatura não pode ser executada diretamente pelo WebSphere MQ; em vez disso, um provedor de seleção de mensagem estendida, por exemplo, IBM Integration Bus, é necessário para processar as mensagens.

Sintaxe do seletor de mensagem
Um seletor de mensagem do WebSphere MQ é uma sequência com sintaxe baseada em um subconjunto da sintaxe de expressão condicional SQL92 .

A ordem na qual um seletor de mensagem é avaliado é da esquerda para a direita dentro de um nível de precedência. É possível usar parênteses para mudar essa ordem. Literais do seletor e nomes de operadores predefinidos estão gravados aqui em maiúsculas; no entanto, não fazem distinção entre maiúsculas e minúsculas.

O WebSphere MQ verifica a correção sintática de um seletor de mensagem no momento em que ele é apresentado. Se a sintaxe da sequência de seleção estiver incorreta ou um nome de propriedade não for válido e um provedor de seleção de mensagem estendida não estiver disponível, `MQRC_SELECTION_NOT_AVAILABLE` será retornado ao aplicativo. Se a sintaxe da sequência de seleção estiver incorreta ou um nome de propriedade não for válido quando uma assinatura for retomada, um `MQRC_SELECTOR_SYNTAX_ERROR` será retornado ao aplicativo. Se a validação do nome da propriedade foi desativada quando a propriedade foi configurada (configurando `MQCMHO_NONE` em vez de `MQCMHO_VALIDATE`) e um aplicativo subsequentemente coloca uma mensagem com um nome de propriedade inválido, esta mensagem nunca será selecionada.

Um seletor pode conter:

- Literais:
  - Literais de sequência são colocados entre aspas simples. Duas aspas simples consecutivas representam uma aspa simples. Os exemplos são: 'literal' e 'literal''s'. Como literais de sequência Java, eles usam a codificação de caracteres Unicode Não é possível usar aspas duplas para delimitar um literal de sequência. Qualquer sequência de bytes pode ser usada entre as aspas simples.

– Uma sequência de bytes é um ou mais pares de caracteres hexadecimais colocados entre aspas duplas e com o prefixo `0x`. Os exemplos são `"0x2F1C"` ou `"0XD43A"` O comprimento de uma sequência de bytes deve ser de pelo menos um byte. Se uma sequência de bytes do seletor tiver correspondência com uma propriedade de mensagem do tipo `MQTYPE_BYTE_STRING`, nenhuma ação especial será executada no zero à esquerda ou à direita. Os bytes são tratados como outro caractere. Endianness também não é considerado. O comprimento de ambas as sequências de bytes do seletor e da propriedade deve ser igual e a sequência de bytes deve ser a mesma.

  Exemplos de seleções de sequências de bytes (suponha que *myBytes* = `0AFC23`) que têm correspondência são:

  - `"myBytes = "0x0AFC23"" = TRUE`

  As seguintes seleções de sequência não têm correspondência:

  - `"myBytes = "0xAFC23"" = MQRC_SELECTOR_SYNTAX_ERROR` (porque o número de bytes não é múltiplo de dois)
  - `"myBytes = "0x0AFC2300"" = FALSE` (porque o zero à direita é significativo na comparação)
  - `"myBytes = "0x000AFC23"" = FALSE` (porque o zero à esquerda é significativo na comparação)
  - `"myBytes = "0x23FC0A"" = FALSE` (porque a ordenação não é considerada)

– Os números hexa começam com um zero, seguido por um `x` maiúsculo ou minúsculo. O restante do literal contém um ou mais caracteres hexa válidos. Os exemplos são `0xA`, `0xAF`, `0X2020`.

– Um zero à esquerda seguido por um ou mais dígitos no intervalo `0-7` é sempre interpretado como sendo o início de um número octal. Não é possível representar um número decimal com prefixo zero dessa forma, por exemplo, `09` retorna um erro de sintaxe, pois `9` não é um dígito octal válido. Exemplos de números octais são `0177`, `0713`.

– Um literal numérico exato é um valor numérico sem um ponto decimal, como `57`, `-957e +62`. Um literal numérico exato pode ter um `L` maiúsculo ou minúsculo à direita; isso não afeta como o número é armazenado ou interpretado. WebSphere MQ suporta números exatos no intervalo de `-9,223,372,036,854,775,808` a `9,223,372,036,854,775,807`.

– Um literal numérico aproximado é um valor numérico em notação científica, como `7E3` ou `-57.9E2`, ou um valor numérico com um decimal, como `7.`, `-95.7`ou `+6.2`. WebSphere MQ suporta números no intervalo de `-1.797693134862315E+308` a `1.797693134862315E+308`.

  O significand deve seguir um caractere de sinal opcional (`+` ou `-`) O significando deve ser um número inteiro ou uma fração. Uma parte fracionária do significando não precisa ter um dígito inicial.

  Um `E` maiúsculo ou minúsculo indica o início de um expoente opcional. O expoente possui um radix decimal e a parte do número do expoente pode ser prefixada por um caractere de sinal opcional.

  Literais numéricos aproximados podem ser finalizados por um caractere `F` ou `D` (não fazem distinção entre maiúsculas e minúsculas). Essa sintaxe existe para suportar o método de linguagem cruzada de identificação de números de precisão simples ou duplos. Esses caracteres são opcionais e não afetam como um literal numérico aproximado é armazenado ou processado. Esses números são sempre armazenados e processados usando precisão dupla.

– Os literais booleanos `TRUE` e `FALSE`.

Nota: As representações IEEE-754 não finitas, como `NaN`, `+Infinity`, `-Infinity`, não são suportadas nas sequências de seleção. Portanto, não é possível usar esses valores como operandos em uma expressão. Zero negativo é tratado da mesma maneira que zero positivo para operações matemáticas.

- Identificadores:

  Um identificador é uma sequência de caracteres de comprimento variável que deve começar com um caractere inicial identificador válido, seguido por zero ou mais caracteres de partes do identificador válidos. As regras para nomes de identificador são iguais àquelas para nomes de propriedades de mensagens, consulte “Nomes de propriedades” na página 19 e “Restrições de nome da propriedade” na página 20 para obter informações adicionais.

Nota: A seleção poderá ser executada na carga útil da mensagem apenas se um provedor de seleção de mensagem estendida estiver disponível.

Identificadores são referências de campo de cabeçalho ou referências de propriedade. O tipo de um valor de propriedade em um seletor de mensagens deve corresponder ao tipo usado para configurar a propriedade, embora a promoção numérica seja executada quando possível. Se ocorrer uma incompatibilidade de tipos, então, o resultado da expressão será FALSE. Se uma propriedade que não existe em uma mensagem for referenciada, seu valor será `NULL`.

Conversões de tipo que se aplicam aos métodos get para propriedades não se aplicam quando uma propriedade é usada em uma expressão do seletor de mensagem. Por exemplo, se você configurar uma propriedade como um valor de sequência e, em seguida, usar um seletor para consultá-lo como um valor numérico, a expressão retornará FALSE.

O campo JMS e os nomes de propriedades que são mapeados para nomes de propriedades ou nomes de campos do `MQMD` também são identificadores válidos em uma cadeia de seleção O WebSphere MQ mapeia os nomes de campo e de propriedade JMS reconhecidos para os valores da propriedade de mensagens Consulte a “Seletores de mensagens no JMS..” na página 819 para obter mais informações. Como exemplo, a sequência de seleção `"JMSPriority >="` seleciona na propriedade `prim` localizada na pasta `jms` da mensagem atual.

- Estouro/estouro negativo:

  Para ambos os números decimais e aproximados, as opções a seguir são indefinidas:

  – Especificar um número que está fora do intervalo definido
  – Especificar uma expressão aritmética que pode causar estouro ou estouro negativo

  Nenhuma verificação é executada para essas condições.

- Espaço em branco:

  Definido como um espaço, alimentação de formulário, nova linha, retorno de linha, tabulação horizontal ou vertical. Os seguintes caracteres Unicode são reconhecidos como espaço em branco:

  – `\u0009 to \u000D`
  – `\u0020`
  – `\u001C`
  – `\u001D`
  – `\u001E`
  – `\u001F`
  – `\u1680`
  – `\u180E`
  – `\u2000` a `\u200A`
  – `\u2028`
  – `\u2029`
  – `\u202F`
  – `\u205F`
  – `\u3000`

- Expressões:

  – Um seletor é uma expressão condicional. Um seletor avaliado como true tem correspondência; um seletor avaliado como false ou unknown não tem correspondência.
  – As expressões aritméticas são compostas por si mesmas, por operações aritméticas, por identificadores (o valor do identificador é tratado como um literal numérico) e por literais numéricos.
  – Expressões condicionais são compostas por si mesmas, por operações de comparação e por operações lógicas.

- O uso padrão de parênteses () para configurar a ordem na qual as expressões são avaliadas é suportado.
- Os operadores lógicos em ordem de precedência: `NOT`, `AND`, `OR`.
- Operadores de comparação: =, >, >=, <, <=, <> (não igual).
  - Sequências de dois bytes são iguais apenas se as sequências tiverem o mesmo comprimento e a sequência de bytes for igual.
  - Somente valores do mesmo tipo podem ser comparados. Uma exceção é que é válido comparar valores numéricos exatos e valores numéricos aproximados (a conversão de tipo necessária é definida pelas regras de promoção numérica Java). Se houver uma tentativa de comparar tipos diferentes, o seletor será sempre false.
  - A comparação de sequência e booleano está restrita a = e <>. Duas sequências serão iguais apenas se contiverem a mesma sequência de caracteres.
- Operadores aritméticos em ordem de precedência:
  - +, - unário.
  - * multiplicação e / divisão.
  - + adição e - subtração.
  - Operações aritméticas em um valor NULL não são suportadas. Se forem tentadas, o seletor completo será sempre false.
  - Operações aritméticas devem usar promoção numérica Java.
- Operador de comparação arithmetic-expr1 [`NOT`] `BETWEEN` arithmetic-expr2 e arithmetic-expr3 :
  - `Age BETWEEN 15 and 19` é equivalente a `age >= 15 AND age <= 19`..
  - `Age NOT BETWEEN 15 and 19` é equivalente a `age < 15 OR age > 19`..
  - Se qualquer uma das expressões de uma operação `BETWEEN` for `NULL`, o valor da operação será false. Se qualquer uma das expressões de uma operação `NOT BETWEEN` for `NULL`, o valor da operação será true.
- identificador `[NOT] IN (string-literal1, string-literal2,...)` operador de comparação em que o identificador tem um valor de Sequência ou `NULL` .
  - `Country IN ('UK', 'US', 'France')` é true para `'UK'` e false para `'Peru'`. Ele é equivalente à expressão `(Country = 'UK') OR (Country = 'US') OR (Country = 'France')`
  - `Country NOT IN ('UK', 'US', 'France')` é false para `'UK'` e true para `'Peru'`. Ele é equivalente à expressão `NOT ((Country = 'UK') OR (Country = 'US') OR (Country = 'France'))`
  - Se o identificador de uma operação `IN` ou `NOT IN` for `NULL`, o valor da operação será desconhecido.
- Operador de comparação `identifier [NOT] LIKE` *`pattern-value`* `[ESCAPE` *`escape-character`*`]` , em que `identifier` tem um valor de cadeia. *pattern-value* é um literal de sequência, em que _ representa qualquer caractere único e % representa qualquer sequência de caracteres (incluindo a sequência vazia). Todos os outros caracteres representam eles mesmos. O *escape-character* opcional é um literal de sequência de caracteres único que é usado para evitar o significado especial de _ e % em *pattern-value*. O operador `LIKE` deve ser usado apenas para comparar dois valores de sequência.
  - `phone LIKE '12%3'` é true para 123 e 12993 e false para 1234.
  - `word LIKE 'l_se'` é true para lose e false para loose.
  - `underscored LIKE '\_%' ESCAPE '\'` é true para `_foo` e false para `bar`.
  - `phone NOT LIKE '12%3'` é false para 123 e 12993 e true para 1234.
  - Se o identificador de uma operação `LIKE` ou `NOT LIKE` for `NULL`, o valor da operação será desconhecido.

  Nota: O operador `LIKE` deve ser usado para comparar dois valores de sequência. O valor de `Root.MQMD.CorrelId` é uma matriz de bytes de 24 bytes, não uma sequência de caracteres.

A sequência do seletor `Root.MQMD.CorrelId LIKE 'ABC%'` é aceita pelo analisador como sintaticamente válida, mas ela é avaliada como false Quando você está comparando uma matriz de bytes com uma sequência de caracteres, `LIKE`, então, não pode ser usado.

- O operador de comparação `identifier IS NULL` testa para um valor de campo de cabeçalho `NULL` ou um valor de propriedade ausente.
- O operador de comparação `identifier IS NOT NULL` testa a existência de um valor de campo de cabeçalho ou um valor de propriedade não nulo.
- Valores nulos

  A avaliação de expressões do seletor que contêm valores `NULL` é definida pela semântica SQL 92 `NULL`, em resumo:

  – SQL trata um valor `NULL` como desconhecido.
  – Comparação ou aritmética com um valor desconhecido sempre resulta em um valor desconhecido.
  – Os operadores `IS NULL` e `IS NOT NULL` convertem um valor desconhecido nos valores `TRUE` e `FALSE`.

  Os operadores booleanos usam lógica de três valores (`T=TRUE, F=FALSE, U=UNKNOWN`)

Tabela 1. Resultado do operador booleano quando a lógica é A AND B

Operador A	**Operador B**	**Resultado (A AND B)**
T	F	F
T	U	U
T	T	T
F	T	F
F	U	F
F	F	F
U	T	U
U	U	U
U	F	F

Tabela 2. Resultado do operador booleano quando a lógica é A OR B

Operador A	**Operador B**	**Resultado (A OR B)**
T	F	T
T	U	T
T	T	T
F	T	T
F	U	U
F	F	F
U	T	T
U	U	U
U	F	U

Tabela 3. Resultado do operador booleano quando a lógica é NOT A	
Operador A	**Resultado (NOT A)**
T	F
F	T
U	U

O seletor de mensagem a seguir seleciona mensagens com um tipo de mensagem de carro, cor azul e peso maior que 2500 lbs:

```
"JMSType = 'car' AND color = 'blue' AND weight > 2500"
```

Embora SQL suporte comparação e aritmética de decimal fixo, os seletores de mensagens não suportam. Por isso literais numéricos exatos são restritos àqueles sem um decimal. Por isso também há valores numéricos com um decimal como uma representação alternativa para um valor numérico aproximado.

Comentários SQL não são suportados.

Conceitos relacionados

Comportamento de seleção
Visão geral do comportamento de seleção IBM WebSphere MQ .

Selecionando no conteúdo de uma mensagem
É possível assinar com base em uma seleção de conteúdo de carga útil da mensagem (também conhecido como filtragem do conteúdo), mas a decisão sobre quais mensagens devem ser entregues para tal assinatura não pode ser executada diretamente pelo WebSphere MQ; em vez disso, um provedor de seleção de mensagem estendida, por exemplo, IBM Integration Bus, é necessário para processar as mensagens.

“Propriedades da Mensagem” na página 18
Use propriedades de mensagem para permitir que um aplicativo selecione mensagens para processar ou recuperar informações sobre uma mensagem sem acessar cabeçalhos MQMD ou MQRFH2. Elas também facilitam a comunicação entre aplicativos WebSphere MQ e JMS.

Referências relacionadas

MsgHandle
MQBUFMH - Converter buffer em identificador de mensagens

Regras e restrições de sequência de seleção
Familiarize-se com essas regras sobre como as sequências de seleção são interpretadas e as restrições de caracteres para evitar problemas em potencial ao usar seletores.

- A equivalência é testada usando um único caractere igual; por exemplo, a `=` b está correto, enquanto a `==` b está incorreto.
- Um operador usado por muitas linguagens de programação para representar 'diferente de' é `!=`. Essa representação não é um sinônimo válido para <>; por exemplo, a `<>` b é válido, enquanto a `!=` b não é válido.
- Aspas simples são reconhecidas somente se o caractere ' (U+0027) for usado. De forma semelhante, aspas duplas, válidas somente quanto usadas para englobar sequências de bytes, devem usar o caractere " (U+0022).
- Os símbolos &, &&, | e || não são sinônimos para conjunção / disjunção lógica; por exemplo, a `&&` b deve ser especificado como a `AND` b.
- Os caracteres curinga `*` e ? não são sinônimos para `%` e `_`.
- Seletores contendo expressões compostas como `20 < b < 30` não são válidos. O analisador avalia os operadores que têm a mesma precedência da esquerda para a direita. O exemplo se tornaria, portanto, `(20 < b) < 30`, o que não faz sentido. Em vez disso, a expressão deve ser gravada como `(b > 20) AND (b < 30)`

- Sequências de bytes devem ser colocadas entre aspas duplas; se as aspas simples forem usadas, a sequência de bytes é considerada uma sequência literal. O número de caracteres (não o número que os caracteres representam) seguindo `0x` deve ser um múltiplo de dois.
- A palavra-chave `IS` não é um sinônimo do caractere de igual. Portanto, as sequências de seleção `a IS 3` e `b IS 'red'` não são válidas.. A palavra-chave `IS` existe somente para suportar casos `IS NULL` e `IS NOT NULL`.

Conceitos relacionados

Considerações sobre UTF-8 e Unicode ao usar seletores de mensagens

Considerações sobre UTF-8 e Unicode ao usar seletores de mensagens

Caracteres, não entre aspas simples, que compõem o palavras-chave reservadas de uma sequência de seleção devem ser inserido em Basic Latin Unicode (variando de caractere U+0000 para U+0007F). Não é válido usar outras representações de ponto de código de caracteres alfanuméricos. Por exemplo, o número 1 deve ser expressado como U+0031 em Unicode, não é válido usar o equivalente de dígito de largura total U+FF11 nem o equivalente árabe U+0661.

Os nomes de propriedades de mensagem podem ser especificados usando qualquer sequência válida de caracteres Unicode. Nomes de propriedades de mensagem contidos dentro de sequências de seleção que são codificados em UTF-8 serão validados mesmo se eles contiverem caracteres de multibyte. A validação de UTF-8 multibyte é rígida e deve-se assegurar que as sequências de UTF-8 válidas sejam usadas para nomes de propriedades de mensagem.

Nenhum processamento extra é executado em nomes de propriedades ou valores ao comparar igualdade. Isso significa, por exemplo, que não pré/decomposição ocorre e ligações não têm nenhum significado especial concedido. Por exemplo, o caractere til pré-composto U+00FC não é considerado como equivalente a U+0075 + U+0308 e a sequência de caracteres ff não é considerada equivalente ao Unicode U+FB00 (LATIN SMALL LIGATURE FF)

Dados de propriedades colocados entre aspas simples podem ser representados por qualquer sequência de bytes e não são validados.

Conceitos relacionados

Regras e restrições de sequência de seleção

Familiarize-se com essas regras sobre como as sequências de seleção são interpretadas e as restrições de caracteres para evitar problemas em potencial ao usar seletores.

Selecionando no conteúdo de uma mensagem

É possível assinar com base em uma seleção de conteúdo de carga útil da mensagem (também conhecido como filtragem do conteúdo), mas a decisão sobre quais mensagens devem ser entregues para tal assinatura não pode ser executada diretamente pelo WebSphere MQ; em vez disso, um provedor de seleção de mensagem estendida, por exemplo, IBM Integration Bus, é necessário para processar as mensagens.

Quando um aplicativo publica em uma cadeia de tópicos, em que um ou mais assinantes têm uma cadeia de seleção selecionando no conteúdo da mensagem, o WebSphere MQ solicitará que o provedor de seleção de mensagens estendidas analise a publicação e informe o WebSphere MQ se a publicação corresponde aos critérios de seleção especificados por cada assinante com um filtro de conteúdo...

Se o provedor de seleção de mensagem estendida determinar que a publicação corresponde à sequência de seleção do assinante, a mensagem continuará sendo entregue ao assinante.

Se o provedor de seleção de mensagem estendida determinar que a publicação não corresponde, a mensagem não será entregue ao assinante. Isso pode fazer com que a chamada MQPUT ou MQPUT1 falhe com o código de razão MQRC_PUBLICATION_FAILURE. Se o provedor de seleção de mensagem estendida não conseguir analisar a publicação, o código de razão MQRC_CONTENT_ERROR será retornado e a chamada MQPUT ou MQPUT1 falhará.

Se o provedor de seleção de mensagem estendida estiver indisponível ou não conseguir determinar se o assinante deve receber a publicação, o código de razão MQRC_SELECTION_NOT_AVAILABLE será retornado e a chamada MQPUT ou MQPUT1 falhará.

Quando uma assinatura estiver sendo criada com um filtro de conteúdo e o provedor de seleção de mensagem estendida não estiver disponível, a chamada MQSUB falhará com o código de razão MQRC_SELECTION_NOT_AVAILABLE. Se uma assinatura com um filtro de conteúdo estiver sendo retomada e o provedor de seleção de mensagem estendida não estiver disponível, a chamada MQSUB retornará um aviso de MQRC_SELECTION_NOT_AVAILABLE, mas a assinatura terá permissão para ser continuada.

Conceitos relacionados

Comportamento de seleção
Visão geral do comportamento de seleção IBM WebSphere MQ .

Sintaxe do seletor de mensagem
Um seletor de mensagem do WebSphere MQ é uma sequência com sintaxe baseada em um subconjunto da sintaxe de expressão condicional SQL92 .

Consumo assíncrono de mensagens do IBM WebSphere MQ

O consumo assíncrono usa um conjunto de extensões da Message Queue Interface (MQI), as chamadas MQI MQCB e MQCTL, que permitem que um aplicativo MQI seja escrito para consumir mensagens de um conjunto de filas. As mensagens são entregues ao aplicativo chamando uma 'unidade de código' identificada pelo aplicativo passando a mensagem ou um token que representa a mensagem.

Nos ambientes de aplicativos mais diretos, a 'unidade de código' é definida por um ponteiro de função, no entanto, em outros ambientes, a 'unidade de código' pode ser definida por um nome de programa ou módulo.

No consumo assíncrono de mensagens, os termos a seguir são usados:

Consumidor de mensagens
Uma construção de programação que permite definir um programa, ou função, a ser chamado com uma mensagem quando uma que corresponda ao requisito dos aplicativos se torna disponível.

Manipulador de eventos
Uma construção de programação que permite definir um programa ou função para chamar quando um evento assíncrono, como quiesce do gerenciador de filas, ocorre.

Retorno de chamada
Um termo genérico usado para fazer referência a uma rotina do Consumidor de mensagens ou do Manipulador de eventos.

Consumo assíncrono pode simplificar o design e a implementação de novos aplicativos, principalmente aquelas que processam diversas filas de entrada ou assinaturas. No entanto, se você estiver usando mais de uma fila de entrada e estiver processando as mensagens na sequência de prioridade, a sequência de prioridade será observada independentemente em cada fila. Pode ser que você obtenha mensagens de prioridade baixa de uma fila antes de mensagens de prioridade alta de outra. A ordem das mensagens entre várias filas não é garantida. Observe também que se você usar saídas de API, poderá precisar mudá-las para incluir as chamadas de MQCB e MQCTL.

As ilustrações a seguir fornecem um exemplo de como é possível usar essa função.

Figura 5 na página 35 mostra um aplicativo multiencadeado consumindo mensagens a partir de duas filas. O exemplo mostra todas as mensagens que estão sendo entregues a uma única função.

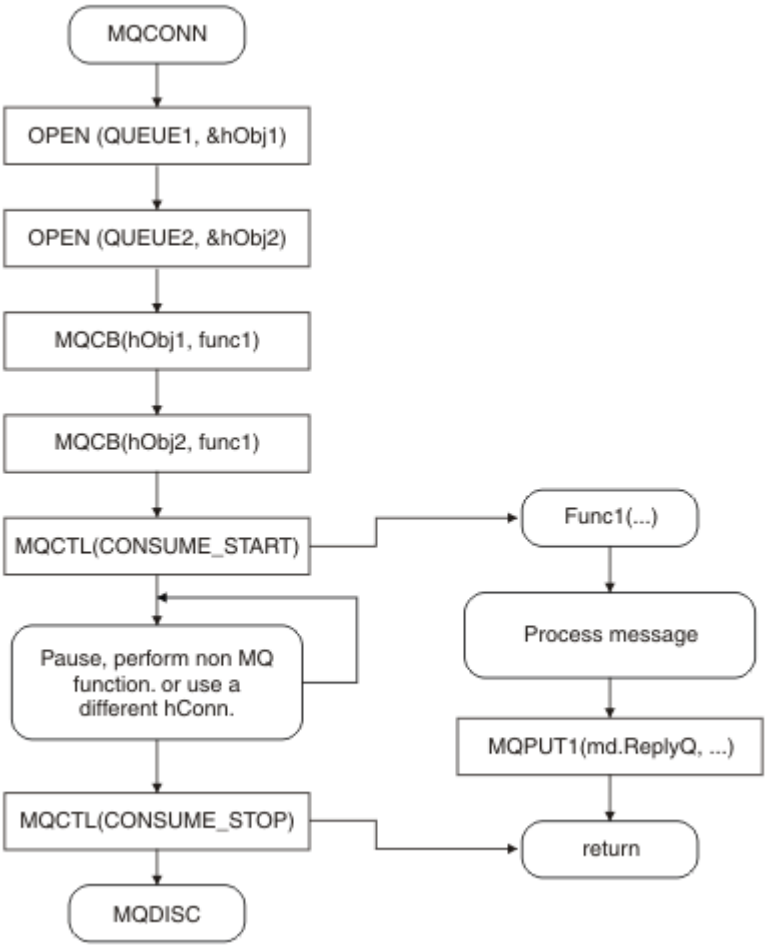


Figura 5. Aplicativo padrão acionado por mensagens consumindo de duas filas

Figura 6 na página 36Esse fluxo de amostra mostra um aplicativo de encadeamento único consumindo mensagens de duas filas. O exemplo mostra todas as mensagens que estão sendo entregues a uma única função.

A diferença do caso assíncrono é que o controle não retornará para o emissor de MQCTL até que todos os consumidores tiverem se desativado; ou seja, um consumidor emitiu uma solicitação MQCTL STOP ou o gerenciador de filas efetua quiesce.

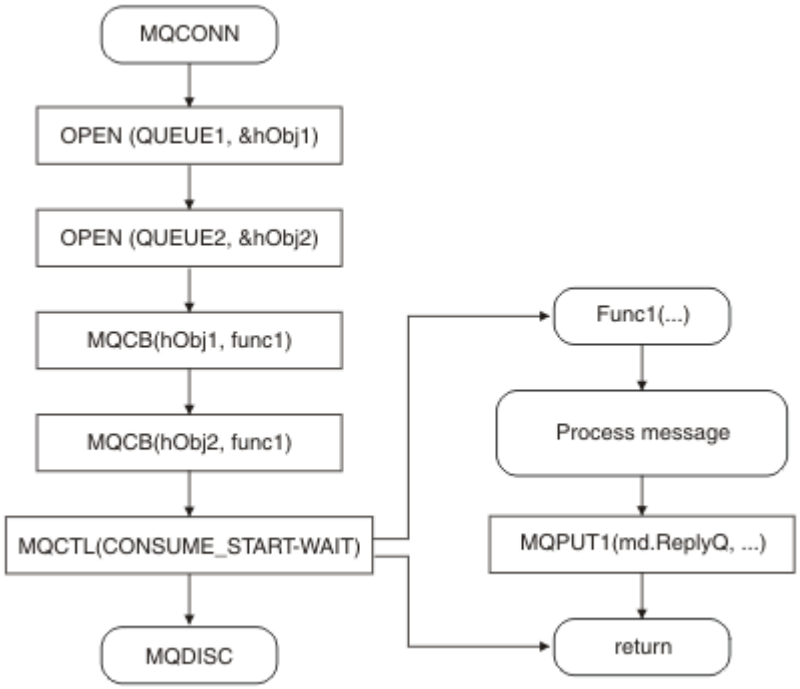


Figura 6. Aplicativo acionado por mensagens de encadeamento único consumindo de duas filas

Grupos de mensagens

As mensagens podem ocorrer dentro de grupos para permitir a ordenação de mensagens.

Os grupos de mensagens permitem que diversas mensagens sejam marcadas como relacionadas entre si e que uma ordem lógica seja aplicada ao grupo (consulte “Ordenação lógica e física” na página 249). Em plataformas diferentes de z/OS, um conceito relacionado, o “Segmentação de mensagem” na página 266 permite que mensagens grandes sejam divididas em segmentos menores. Não é possível usar mensagens agrupadas ou segmentadas ao colocar em um tópico.

A hierarquia dentro de um grupo é a seguinte:

Grupo
Este é o nível mais alto na hierarquia e é identificado por um *GroupId*. Ele consiste em uma ou mais mensagens que contêm o mesmo *GroupId*. Essas mensagens podem ser armazenadas em qualquer lugar na fila.

Nota: O termo *mensagem* é usado aqui para denotar um item em uma fila, como seria retornado por uma única MQGET que não especifique MQGMO_COMPLETE_MSG.

Figura 7 na página 36 mostra um grupo de mensagens lógicas:

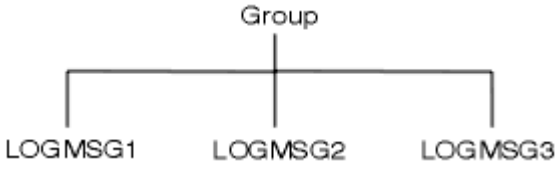


Figura 7. Grupo de mensagens lógicas

Ao abrir uma fila e especificar MQOO_BIND_ON_GROUP, você força todas as mensagens em um grupo que são enviadas para esta fila a serem enviadas à mesma instância da fila. Para obter mais informações sobre a opção BIND_ON_GROUP, consulte Manipulando afinidades de mensagens.

Mensagem lógica

As mensagens lógicas dentro de um grupo são identificadas pelos campos *GroupId* e *MsgSeqNumber*. O *MsgSeqNumber* começa em 1 para a primeira mensagem em um grupo e se uma mensagem não estiver em um grupo, o valor do campo é 1.

Use as mensagens lógicas dentro de um grupo para:

- Certificar-se da ordenação (se isto não for garantido sob as circunstâncias nas quais a mensagem é transmitida).
- Permitir que os aplicativos agrupem mensagens semelhantes (por exemplo, todas aquelas que devem ser processadas pela mesma instância do servidor).

Cada mensagem dentro de um grupo consiste em uma mensagem física, a menos que seja dividida em segmentos. Cada mensagem é logicamente uma mensagem separada, e apenas os campos *GroupId* e *MsgSeqNumber* no MQMD precisam ter qualquer relacionamento com outras mensagens no grupo Outros campos no MQMD são independentes; alguns podem ser idênticos para todas as mensagens no grupo, enquanto outros podem ser diferentes. Por exemplo, as mensagens em um grupo podem ter nomes de formato diferente, CCSIDs e codificações.

Segmentar

Os segmentos são usados para lidar com mensagens muito grandes para o aplicativo put ou get ou para o gerenciador de filas (incluindo gerenciadores de filas intervenientes pelos quais a mensagem passa). Para obter informações adicionais, consulte "Segmentação de mensagem" na página 266.

Uma mensagem individual é dividida em mensagens menores chamadas *segmentos*. Um segmento de uma mensagem é identificado pelos campos *GroupId*, *MsgSeqNumber*e *Offset* .. O campo *Offset* inicia em zero para o primeiro segmento em uma mensagem.

Cada segmento consiste em uma mensagem física que pode pertencer a um grupo (Figura 8 na página 37 mostra um exemplo de mensagens dentro de um grupo). Um segmento é logicamente parte de uma única mensagem, portanto, apenas os campos *MsgId*, *Offset*e *SegmentFlag* no MQMD devem ser diferentes entre segmentos separados da mesma mensagem. Se um segmento não chegar, o código de razão MQRC_INCOMPLETE_GROUP ou MQRC_INCOMPLETE_MSG é retornado conforme apropriado.

Figura 8 na página 37 mostra um grupo de mensagens lógicas, algumas das quais são segmentadas:

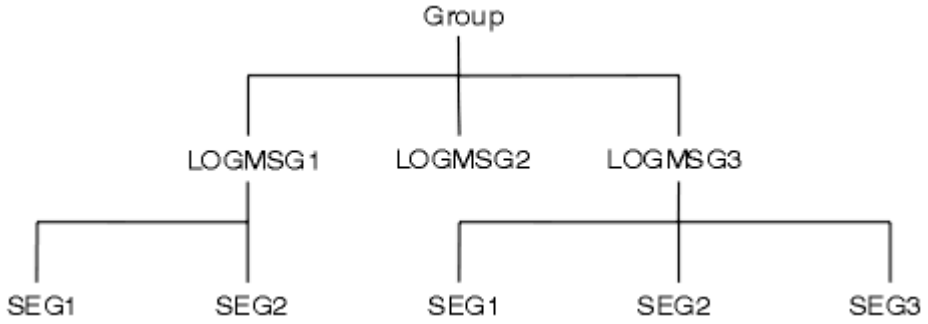


Figura 8. Mensagens segmentadas

Você não pode usar segmentos ou mensagens agrupadas com Publicação/Assinatura.

Para uma descrição de mensagens lógicas e físicas, consulte "Ordenação lógica e física" na página 249. Para obter informações adicionais sobre a segmentação de mensagens, consulte "Segmentação de mensagem" na página 266

Persistência de mensagem

As mensagens persistentes são gravadas em logs e arquivos de dados de fila.

Se um gerenciador de filas for reiniciado após uma falha, ele recuperará essas mensagens persistentes conforme necessário a partir dos dados registrados. As mensagens que não são persistentes serão

descartadas se um gerenciador de filas parar, independentemente de a parada ser resultante de um comando do operador ou devido à falha de alguma parte do seu sistema.

Ao criar uma mensagem, se você inicializar o descritor de mensagens (MQMD) usando os padrões, a persistência de mensagem é obtida do atributo *DefPersistence* da fila especificada no comando MQOPEN. Como alternativa, é possível configurar a persistência da mensagem usando o campo *Persistence* da estrutura MQMD para definir a mensagem como persistente ou não persistente..

O desempenho do seu aplicativo é afetado ao usar as mensagens persistentes. A extensão do efeito depende das características de desempenho do subsistema de E/S do computador e de como usar as opções do ponto de sincronização em cada plataforma:

- Uma mensagem persistente, fora da unidade de trabalho atual, é gravada no disco em cada operação put e get. Consulte o "Confirmando e fazendo backup de unidades de trabalho" na página 327.
- Em IBM WebSphere MQ em sistemas UNIX , IBM WebSphere MQ em Linux sistemas, e IBM WebSphere MQ para Windows, uma mensagem persistente dentro da unidade de trabalho atual é registrada somente quando a unidade de trabalho é confirmada (e a unidade de trabalho poderia conter muitas operações de filas)...

As mensagens não persistentes podem ser usadas para o sistema de mensagens rápidas. Consulte Segurança de mensagens para obter informações adicionais sobre as mensagens rápidas.

Nota: Uma combinação de composição de mensagens persistentes em uma unidade de trabalho e composição de mensagens persistentes fora de uma unidade ou de trabalho, pode, potencialmente, causar problemas severos de desempenho em seus aplicativos. Isso se aplica, em especial, quando a mesma fila de destino é usada para ambas as operações.

Mensagens que falham na entrega

Quando um gerenciador de filas não consegue colocar uma mensagem em uma fila, você tem várias opções.

É possívelrá:

- Tentar colocar a mensagem na fila novamente.
- Solicitar que a mensagem seja retornada ao emissor.
- Colocar a mensagem na fila de mensagens não entregues.

Consulte "Manipulando erros do programa" na página 555 para obter mais informações.

Mensagens que são restauradas

Ao processar mensagens de uma fila sob o controle de uma unidade de trabalho, a unidade de trabalho pode consistir em uma ou mais mensagens. Se uma restauração ocorrer, as mensagens que tiverem sido recuperadas da fila serão recolocadas na fila e elas poderão ser processadas novamente em outra unidade de trabalho. Se o processamento de uma mensagem específica estiver causando o problema, a unidade de trabalho será restaurada novamente. Isso pode causar um loop de processamento. As mensagens que foram colocadas em uma fila são removidas da fila.

Um aplicativo pode detectar mensagens que são capturadas em um loop assim testando o campo *BackoutCount* do MQMD. O aplicativo pode corrigir a situação ou emitir um aviso para um operador.

Em WebSphere MQ para WebSphere MQ para Windows, WebSphere MQ em sistemas UNIX , WebSphere MQ em Linux sistemas a contagem de restaurações sempre sobrevive às reinicializações do gerenciador de filas Qualquer mudança ao atributo *HardenGetBackout* será ignorada.

Para obter mais informações sobre a consolidação e restauração de mensagens, consulte "Confirmando e fazendo backup de unidades de trabalho" na página 327.

Fila de resposta e gerenciador de filas

Existem ocasiões em que você pode receber mensagens em resposta a uma mensagem que você enviar:

- Uma mensagem de resposta em resposta a uma mensagem de pedido
- Uma mensagem de relatório sobre um evento inesperado ou de expiração
- Uma mensagem de relatório sobre um COA (Confirmação da Chegada) ou evento de um COD (Confirmação da Entrega)
- Uma mensagem de relatório sobre um PAN (Positive Action Notification) ou evento de um NAN (Negative Action Notification)

Usando a estrutura MQMD, especifique o nome da fila para a qual você deseja enviar as mensagens de resposta e de relatório no campo *ReplyToQ*. Especifique o nome do gerenciador de filas que possui a fila de resposta no campo *ReplyToQMgr*.

Se você deixar o campo *ReplyToQMgr* em branco, o gerenciador de filas configurará o conteúdo dos seguintes campos no descritor de mensagens na fila:

ReplyToQ
Se *ReplyToQ* for uma definição local de uma fila remota, o campo *ReplyToQ* será configurado para o nome da fila remota; caso contrário, esse campo não será mudado.

ReplyToQMgr
Se *ReplyToQ* for uma definição local de uma fila remota, o campo *ReplyToQMgr* será configurado para o nome do gerenciador de filas que possui a fila remota; caso contrário, o campo *ReplyToQMgr* será configurado para o nome do gerenciador de filas ao qual seu aplicativo está conectado.

Nota: É possível solicitar que um gerenciador de filas faça mais de uma tentativa de entregar uma mensagem e que a mensagem seja descartada se falhar. Se a mensagem, após falhar em ser entregue, não dever ser descartada, o gerenciador de filas remotas a colocará na fila de mensagens não entregues (veja "Usando a fila de mensagens não entregues" na página 558).

Contexto da mensagem

As informações de *Contexto da Mensagem* permitem que o aplicativo recupere a mensagem para descobrir sobre o originador da mensagem.

O aplicativo de recuperação pode desejar:

- Verificar se o aplicativo de envio tem o nível correto de autoridade
- Executar algumas funções de contabilidade para que ele possa carregar o aplicativo de envio para qualquer trabalho que ele precise executar
- Manter uma trilha de auditoria de todas as mensagens com as quais ele trabalhou

Quando você usa a chamada MQPUT ou MQPUT1 para colocar uma mensagem em uma fila, é possível especificar se o gerenciador de filas deve incluir algumas informações de contexto padrão para o descritor de mensagens. Aplicativos que possuem o nível apropriado de autoridade podem incluir informações de contexto adicionais. Para obter mais informações sobre como especificar informações de contexto, consulte "Controlando informações de contexto" na página 235.

O contexto do usuário é usado pelo gerenciador de filas ao gerar os seguintes tipos de mensagem de relatório:

- Confirmar na entrega
- Expiração

Quando essas mensagens de relatório são geradas, o contexto do usuário é verificado para autoridade +put e +passid no destino do relatório. Quando o contexto do usuário possui autoridade insuficiente, a mensagem de relatório é colocada na fila de mensagens não entregues se uma fila tiver sido definida. Onde não houver uma fila de mensagens não entregues, a mensagem de relatório é descartada.

Todas as informações de contexto são armazenadas nos campos de contexto do descritor de mensagens. O tipo de informação é classificado em identidade, origem e informações de contexto do usuário.

Contexto de Identidade

A informação de *Contexto de identidade* identifica o usuário do aplicativo que coloca primeiro a mensagem em uma fila. Os aplicativos devidamente autorizados podem configurar os seguintes campos:

- O gerenciador de filas preenche o campo *UserIdentifier* com um nome que identifica o usuário. O modo que o gerenciador de filas pode fazer isso depende do ambiente no qual o aplicativo está sendo executado.
- O gerenciador de filas preenche o campo *AccountingToken* com um token ou número que é determinado a partir do aplicativo que coloca a mensagem.
- Os aplicativos podem usar o campo *ApplIdentityData* para qualquer informação adicional que desejam incluir sobre o usuário (por exemplo, uma senha criptografada).

Um identificador de segurança do sistema (SID) Windows é armazenado no campo *AccountingToken* quando uma mensagem é criada em WebSphere MQ para Windows. O SID pode ser usado para suplementar o campo *UserIdentifier* e para estabelecer as credenciais de um usuário.

Para obter informações sobre como o gerenciador de filas preenche os campos *UserIdentifier* e *AccountingToken*, consulte as descrições desses campos em UserIdentifier e AccountingToken.

Os aplicativos que passam mensagens de um gerenciador de filas para outro devem também passar informações de contexto de identidade para que outros aplicativos saibam a identidade do originador da mensagem.

Contexto de origem

As informações de *Contexto de origem* descrevem o aplicativo que coloca a mensagem na fila na qual a mensagem está *atualmente* armazenada. O descritor de mensagens contém os seguintes campos para informações de contexto de origem:

PutApplType	O tipo de aplicativo que colocou a mensagem (por exemplo, uma transação CICS).
PutApplName	O nome do aplicativo que coloca a mensagem (por exemplo, o nome de um trabalho ou transação).
PutDate	A data em que a mensagem foi colocada na fila.
PutTime	A hora em que a mensagem foi colocada na fila.
ApplOriginData	Todas as informações adicionais que um aplicativo deseja incluir sobre a origem da mensagem. Por exemplo, ele pode ser configurado por aplicativos devidamente autorizados para indicar se os dados de identidade são confiáveis.

As informações de contexto de origem são geralmente fornecidas pelo gerenciador de filas. GMT (Horário de Greenwich) é usado para os campos *PutDate* e *PutTime*. Consulte as descrições desses campos em PutDate e PutTime.

Um aplicativo com autoridade suficiente pode fornecer seu próprio contexto. Isso permite que as informações de contabilidade sejam preservadas quando um único usuário tiver uma ID de usuário diferente em cada um dos sistemas que processam uma mensagem que eles originaram.

Objetos WebSphere MQ

Essas informações fornecem detalhes sobre os objetos do WebSphere MQ que incluem: gerenciadores de fila, grupos de filas compartilhadas, filas, objetos de tópico administrativo, listas de nomes, definições de processos, objetos de informações sobre autenticação, canais, classes de armazenamento, listeners e serviços

Os gerenciadores de filas definem as propriedades (conhecidas como atributos) desses objetos. Os valores desses atributos afetam a maneira na qual o WebSphere MQ processa esses objetos Em seus aplicativos, use o Message Queue Interface (MQI) para controlar esses objetos. Os objetos são identificados por um *descritor de objeto* (MQOD) quando direcionados de um programa.

Ao usar comandos do WebSphere MQ para definir, alterar ou excluir objetos, por exemplo, o gerenciador de filas verifica se você tem o nível necessário de autoridade para executar essas operações. Da mesma forma, quando um aplicativo usa a chamada MQOPEN para abrir um objeto, o gerenciador de filas verifica se o aplicativo possui o nível necessário de autoridade antes que conceda acesso a esse objeto. As verificações são feitas no nome do objeto sendo aberto.

Conceitos relacionados

"Controlando informações de contexto" na página 235
Quando você usa a chamada MQPUT ou MQPUT1 para colocar uma mensagem em uma fila, é possível especificar se o gerenciador de filas deve incluir algumas informações de contexto padrão para o descritor de mensagens. Aplicativos que possuem o nível apropriado de autoridade podem incluir informações de contexto adicionais. É possível usar o campo de opções na estrutura MQPMO para controlar informações de contexto.

Referências relacionadas

"Opções de MQOPEN relacionadas ao contexto da mensagem" na página 225
Se deseja conseguir associar as informações a uma mensagem ao colocá-la em uma fila, deve-se usar uma das opções de contexto da mensagem ao abrir a fila.

Preparando e executando aplicativos Microsoft Transaction Server

Para preparar um aplicativo MTS para executar como um aplicativo cliente MQI do WebSphere MQ , siga estas instruções conforme apropriado para seu ambiente.

Para obter informações gerais sobre como desenvolver aplicativos do Microsoft Transaction Server (MTS) que acessam os recursos do WebSphere MQ , consulte a seção no MTS na Central de Ajuda do WebSphere MQ

Para preparar um aplicativo MTS para ser executado como um aplicativo cliente MQI do WebSphere MQ , execute um dos seguintes procedimentos para cada componente do aplicativo:

- Se o componente usar as ligações de linguagem C para o MQI, siga as instruções no "Preparando programas C no Windows" na página 466, mas vincule o componente à biblioteca mqicxa.lib em vez da mqic.lib.
- Se o componente usar as classes C++ do WebSphere MQ , siga as instruções em "Construindo programas C++ no Windows" na página 660 , mas vincule o componente à biblioteca imqx23vn.lib em vez de imqc23vn.lib.
- Se o componente usar as ligações de linguagem Visual Basic para o MQI, siga as instruções no "Preparando programas do Visual Basic no Windows" na página 469, mas quando definir o projeto Visual Basic, digite `MqType=3` no campo **Argumentos de compilação condicional**.
- Se o componente usar o WebSphere MQ Classes de Automação para ActiveX (MQAX), defina uma variável de ambiente, GMQ_MQ_LIB, com o valor `mqic32xa.dll` .

  É possível definir a variável de ambiente a partir de seu aplicativo ou ela pode ser definida de modo que seu escopo seja para todo o sistema. Entretanto, defini-la como para todo o sistema pode causar que qualquer aplicativo MQAX existente que não defina a variável de ambiente a partir do aplicativo se comporte incorretamente.

Usando o IBM WebSphere MQ com o WebSphere Application Server

Use este tópico para entender o uso de IBM WebSphere MQ com WebSphere Application Server

Os aplicativos gravados em Java que estão em execução no WebSphere Application Server podem usar a especificação Java Messaging Service (JMS) para executar o sistema de mensagens. Sistema de mensagens ponto a ponto neste ambiente pode ser fornecido por um gerenciador de filas do IBM WebSphere MQ

Um benefício de usar um gerenciador de fila do IBM WebSphere MQ para fornecer o sistema de mensagens ponto a ponto é que a conexão de aplicativos JMS pode participar totalmente da funcionalidade de uma rede IBM WebSphere MQ , que permite que os aplicativos troquem mensagens com gerenciadores de filas em execução em uma variedade de plataformas.

Os aplicativos podem usar o *transporte de cliente* ou *transporte de ligações* para o objeto connection factory da fila. Para *transporte de ligações* , o gerenciador de filas deve existir localmente para o aplicativo que requer uma conexão. Se o gerenciador de filas não for local para o aplicativo, então a *Conexão do Cliente* deverá ser instalada para permitir que o aplicativo se conecte a um gerenciador de filas em execução em outra máquina ou imagem

Por padrão, as mensagens JMS retidas em filas do IBM WebSphere MQ usam um cabeçalho MQRFH2 para conter algumas informações do cabeçalho da mensagem JMS. Muitos aplicativos IBM WebSphere MQ legados não podem processar mensagens com esses cabeçalhos e requerem seus próprios cabeçalhos de características, por exemplo, o MQCIH para CICS Bridge ou MQWIH para aplicativos de Fluxo de trabalho do IBM WebSphere MQ Para obter mais detalhes sobre essas considerações especiais, consulte “Mapeando mensagens JMS para mensagens do WebSphere MQ” na página 822.

Cenários de Suporte Transacional

Usando o suporte transacional é possível ativar seus aplicativos para funcionarem confiavelmente com bancos de dados.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Esta seção apresenta suporte transacional. O trabalho requerido para ativar que seus aplicativos usem o IBM WebSphere MQ com um produto de banco de dados abrange as áreas de administração do sistema e programação de aplicativo. Use as informações aqui junto com o “Confirmando e fazendo backup de unidades de trabalho” na página 327

Iniciamos apresentando as unidades de trabalho que formam as transações e, em seguida, descrevemos as maneiras pelas quais é possível ativar o IBM WebSphere MQ para coordenar transações com bancos de dados.

Conceitos relacionados

“Introduzindo Unidades de Trabalho” na página 42
Este tópico introduz os conceitos gerais de unidade de trabalho, confirmação, restauração e ponto de sincronização. Ele também contém dois cenários que ilustram unidades globais de trabalho.

IBM WebSphere MQ e HP NonStop TMF

Introduzindo Unidades de Trabalho

Este tópico introduz os conceitos gerais de unidade de trabalho, confirmação, restauração e ponto de sincronização. Ele também contém dois cenários que ilustram unidades globais de trabalho.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Quando um programa colocar mensagens nas filas em uma unidade de trabalho, essas mensagens se tornarão visíveis para outros programas somente quando o programa *confirmar* a unidade de trabalho. Para confirmar uma unidade de trabalho, todas as atualizações devem ser bem sucedidas em preservar a integridade dos dados.

Se o programa detectar um erro e decidir não tornar a operação de inserção permanente, ela poderá *restaurar* a unidade de trabalho. Quando um programa executa uma restauração, WebSphere MQ restaura as filas removendo as mensagens que foram colocadas nas filas por essa unidade de trabalho.

Da mesma forma, quando um programa obtiver mensagens de uma ou mais filas em uma unidade de trabalho, essas mensagens permanecerão nas filas até que o programa confirme a unidade de trabalho,

mas as mensagens não estão disponíveis para serem recuperadas por outros programas. As mensagens serão permanentemente excluídas das filas quando o programa confirmar a unidade de trabalho. Se o programa restaurar a unidade de trabalho, o WebSphere MQ restaurará as filas tornando as mensagens disponíveis para serem recuperadas por outros programas.

A decisão de confirmar ou restaurar as mudanças é tomada, no caso mais simples, no final de uma tarefa. No entanto, pode ser mais útil para um aplicativo sincronizar alterações de dados em outros pontos lógicos dentro de uma tarefa. Esses pontos lógicos são chamados de pontos de sincronização e o período de processamento de um conjunto de atualizações entre dois pontos de sincronização é chamado de *unidade de trabalho*. Várias chamadas MQGET e MQPUT podem fazer parte de uma única unidade de trabalho.

Com o WebSphere MQ, é necessário distinguir entre unidades de trabalho *locais* e *globais* :

Unidades de trabalho locais
São aquelas nas quais as únicas ações são inseridas e obtêm das filas do WebSphere MQ e a coordenação de cada unidade de trabalho é fornecida dentro do gerenciador de filas usando um processo de *confirmação de fase única* .

Use as unidades de trabalho locais quando os únicos recursos a serem atualizados forem as filas que são gerenciadas por um único gerenciador de fila do WebSphere MQ As atualizações são consolidadas usando o verbo MQCMIT ou restauradas usando MQBACK.

Não há tarefas de administração do sistema, além de gerenciamento de log, que está envolvido no uso de unidades de trabalho locais. Em seus aplicativos, em que as chamadas MQPUT e MQGET são usadas com MQCMIT e MQBACK, tente usar as opções MQPMO_SYNCPOINT e MQGMO_SYNCPOINT. (Para obter informações sobre o gerenciamento de log, consulte Gerenciando arquivos de log .)

Unidades de trabalho globais
São aquelas em que outros recursos, como tabelas em um banco de dados relacional, também são atualizados. Quando mais de um *gerenciador de recursos* estiver envolvido, haverá a necessidade do software *gerenciador de transações* que usa um processo de *two-phase commit* coordenar a unidade global de trabalho.

Use unidades globais de trabalho quando você também precisar incluir atualizações no software do gerenciador de banco de dados relacional, como Db2, Oracle, Sybasee Informix.

Existem vários cenários possíveis para usar unidades globais de trabalho. Dois cenários estão documentados aqui:

1. No primeiro, o gerenciador de filas em si age como o gerenciador de transações. Neste cenário, os verbos MQI controlam as unidades globais de trabalho; eles são iniciados em aplicativos que usam o verbo MQBEGIN e, em seguida, são confirmados usando o MQCMIT ou restaurados usando MQBACK.
2. No segundo, a função de gerenciador de transações é executada por outro software, como TXSeries, Encinaou Tuxedo.. Nesse cenário, uma API fornecida pelo software do gerenciador de transações é usada para controlar a unidade de trabalho (por exemplo, EXEC CICS SYNCPOINT para TXSeries).

As seções a seguir descrevem todas as etapas necessárias para usar unidades globais de trabalho, organizadas por dois cenários:

- “Cenário 1: Gerenciador de Filas Executa a Coordenação” na página 43
- “Cenário 2: Outro Software Fornece a Coordenação” na página 71

Cenário 1: Gerenciador de Filas Executa a Coordenação

No cenário 1, o gerenciador de filas age como o gerenciador de transações. Neste cenário, os verbos MQI controlam as unidades globais de trabalho; eles são iniciados em aplicativos que usam o verbo MQBEGIN e, em seguida, são confirmados usando o MQCMIT ou restaurados usando MQBACK.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Nível de isolamento

No IBM WebSphere MQ, uma mensagem em uma fila pode estar visível antes de uma atualização de banco de dados, dependendo do design isolamento de transação implementado no banco de dados.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Quando um gerenciador de filas do IBM WebSphere MQ estiver trabalhando como um gerenciador de transações XA, para coordenar atualizações em gerenciadores de recursos XA, o seguinte protocolo de confirmação é seguido:

1. Prepare todos os gerenciadores de recursos XA.
2. Confirme o gerenciador de recurso do gerenciador de filas do IBM WebSphere MQ
3. Confirme outros gerenciadores de recursos.

Entre as etapas 2 e 3, um aplicativo pode ver uma mensagem que é confirmada para a fila, mas a linha correspondente no banco de dados não refletirá essa mensagem.

Isso não será um problema se o banco de dados estiver configurado de modo que as chamadas API do banco de dados do aplicativo esperem que as atualizações pendentes sejam concluídas.

É possível resolver isso configurando o banco de dados de forma diferente. O tipo de configuração necessária é referido como o "nível de isolamento". Para obter mais informações sobre os níveis de isolamento, consulte a documentação do banco de dados. Você pode, alternativamente, configurar o gerenciador de filas para confirmar os gerenciadores de recursos na ordem reversa a seguir:

1. Prepare todos os gerenciadores de recursos XA.
2. Confirme outros gerenciadores de recursos.
3. Confirme o gerenciador de recurso do gerenciador de filas do IBM WebSphere MQ

Quando você altera o protocolo, o gerenciador de filas do IBM WebSphere MQ é confirmado por último de forma que os aplicativos que leem as mensagens das filas verão uma mensagem apenas após a atualização de banco de dados correspondente tiver sido concluído.

Para configurar o gerenciador de filas para usar esse protocolo mudado, configure a variável de ambiente **AMQ_REVERSE_COMMIT_ORDER**.

Defina essa variável de ambiente no ambiente a partir do qual o **strmqm** é executado para iniciar o gerenciador de filas. Por exemplo, execute no shell a seguir antes de iniciar o gerenciador de filas:

```
export AMQ_REVERSE_COMMIT_ORDER=1
```

Nota: A configuração dessa variável de ambiente pode causar uma entrada de log adicional por transação, portanto, isso terá um pequeno impacto no desempenho de cada transação.

Coordenação do Banco de Dados

Quando o gerenciador de filas coordenar unidades de trabalho globais, será possível integrar as atualizações do banco de dados dentro das unidades de trabalho. Ou seja, um aplicativo MQI e SQL combinado pode ser gravado e os verbos MQCMIT e MQBACK podem ser usados para consolidar ou recuperar as alterações nas filas e bancos de dados em conjunto.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

O gerenciador de filas alcança isto usando o protocolo two-phase commit descrito em *Processamento de Transação Distribuída do X/Open: A Especificação XA*. Quando uma unidade de trabalho estiver para ser

confirmada, o gerenciador de filas primeiro solicita a cada gerenciador de banco de dados participante se ele está disposto a confirmar suas atualizações. Somente se todos os participantes, incluindo o gerenciador de filas em si, estiverem preparados para confirmar, todas as atualizações de filas e banco de dados serão confirmadas. Se algum participante não puder preparar suas atualizações, a unidade de trabalho será restaurada.

Em geral, uma unidade global de trabalho é implementada em um aplicativo pelo método a seguir (em pseudocódigo):

```
MQBEGIN
MQGET (inclui o sinalizador MQGMO_SYNCPOINT nas opções de mensagem)
MQPUT (inclui o sinalizador MQPMO_SYNCPOINT nas opções de mensagem)
SQL INSERT
MQCMIT
```

O propósito do MQBEGIN é denotar o início de uma unidade global de trabalho. O propósito do MQCMIT é denotar o final da unidade global de trabalho e concluí-la, com todos os gerenciadores de recursos participantes, usando o protocolo two-phase commit.

Quando a unidade de trabalho (também conhecida como uma *transação*) é concluída com êxito usando MQCMIT, todas as ações executadas nessa unidade de trabalho se tornam permanentes ou irreversíveis. Se, por algum motivo, a unidade de trabalho falhar, todas as ações serão restauradas. Não é possível que uma ação em uma unidade de trabalho seja tornada permanente enquanto outra estiver sendo restaurada. Este é o princípio de uma unidade de trabalho: ou todas as ações na unidade de trabalho são tornadas permanentes ou nenhuma delas é.

Nota:

1. O programador de aplicativo pode forçar que uma unidade de trabalho seja restaurada chamando o MQBACK. A unidade de trabalho também será restaurada pelo gerenciador de filas se o aplicativo ou banco de dados *falhar* antes do MQCMIT ser chamado.
2. Se um aplicativo chamar a chamar o MQDISC sem o MQCMIT, o gerenciador de filas se comporta como se o MQCMIT tivesse sido chamado e confirma a unidade de trabalho.

Entre MQBEGIN e MQCMIT, o gerenciador de filas não faz nenhuma chamada para o banco de dados para atualizar seus recursos. Ou seja, a única maneira de alterar tabelas de um banco de dados é por meio do seu código (por exemplo, SQL INSERT no pseudocódigo).

Suporte de recuperação completo será fornecido se o gerenciador de filas perder contato com qualquer um dos gerenciadores de banco de dados durante o protocolo de confirmação. Se um gerenciador de banco de dados se tornar indisponível enquanto ele estiver em dúvida ou seja, ele tiver sido preparado para confirmação com sucesso, mas ainda precisar receber uma decisão de confirmação ou de restauração, o gerenciador de filas manterá o resultado da unidade de trabalho até que esse resultado seja entregue com sucesso ao banco de dados. Da mesma forma, se o gerenciador de filas for finalizado com operações de confirmação incompletas pendentes, elas serão lembradas na reinicialização do gerenciador de filas. Se um aplicativo for finalizado inesperadamente, a integridade da unidade de trabalho não será comprometida, mas o resultado depende de onde no processo o aplicativo terminado, conforme descrito em Tabela 5 na página 46.

O que acontece quando o programa de aplicativo ou banco de dados falhar está resumido nas tabelas a seguir:

Tabela 4. O que Acontece Quando um Servidor de Banco de Dados Falhar

Ocorrência de falha	**Resultado**
Antes da chamada do aplicativo para MQCMIT.	A unidade de trabalho é restaurada.
Durante a chamada do aplicativo para MQCMIT, **antes** de todos os bancos de dados indicarem que foram preparados com sucesso.	A unidade de trabalho é restaurada com um código de razão de MQRC_BACKED_OUT.

Tabela 4. O que Acontece Quando um Servidor de Banco de Dados Falhar (continuação)	
Ocorrência de falha	**Resultado**
Durante a chamada do aplicativo para MQCMIT, **após** todos os bancos de dados terem indicado que foram preparados com sucesso, mas antes de todos terem indicado que foram confirmados com sucesso.	A unidade de trabalho é mantida no estado recuperável pelo gerenciador de filas, com um código de razão de MQRC_OUTCOME_PENDING.
Durante a chamada do aplicativo para MQCMIT, **após** todos os bancos de dados terem indicado que foram confirmados com sucesso.	A unidade de trabalho é confirmada com um código de razão de MQRC_NONE.
Após a chamada do aplicativo para MQCMIT.	A unidade de trabalho é confirmada com um código de razão de MQRC_NONE.

Tabela 5. O que Acontece Quando um Programa de Aplicativo Falha	
Ocorrência de falha	**Resultado**
Antes da chamada do aplicativo para MQCMIT.	A unidade de trabalho é restaurada.
Durante a chamada do aplicativo para MQCMIT, **antes** do gerenciador de filas ter recebido a solicitação de MQCMIT do aplicativo.	A unidade de trabalho é restaurada.
Durante a chamada do aplicativo para MQCMIT, **após** o gerenciador de filas ter recebido a solicitação de MQCMIT do aplicativo.	O gerenciador de filas tenta confirmar usando o two-phase commit (sujeito à execução com sucesso de produtos de banco de dados à confirmação de suas partes da unidade de trabalho).

Caso o código de razão no retorno do MQCMIT seja MQRC_OUTCOME_PENDING, a unidade de trabalho será lembrada pelo gerenciador de filas até que seja capaz de reestabelecer o contato com o servidor de banco de dados, e dizer a ele para confirmar sua parte da unidade de trabalho. Consulte “Considerações quando o contato estiver perdido com o gerenciador de recursos XA” na página 63 para obter informações sobre como e quando a recuperação é concluída.

O gerenciador de filas se comunica com os gerenciadores de banco de dados usando a interface XA conforme descrito em *Processamento de Transação Distribuída do X/Open: A Especificação XA*. Exemplos dessas chamadas de função são xa_open, xa_start, xa_end, xa_prepare e xa_commit. Usamos os termos *gerenciador de transações* e *gerenciador de recursos* com o mesmo sentido que eles são usados na especificação XA.

Restrições
Há restrições ao suporte de coordenação de banco de dados.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

As seguintes restrições se aplicam:

- A capacidade de coordenar atualizações de banco de dados dentro das unidades de trabalho do WebSphere MQ não é **suportada** em um aplicativo cliente MQI O uso de MQBEGIN em um aplicativo cliente falhará. Um programa que as chamadas MQBEGIN devem ser executadas como um aplicativo *servidor* na mesma máquina que o gerenciador de filas.

  Nota: Um aplicativo *servidor* é um programa que foi vinculado às bibliotecas do servidor WebSphere MQ necessárias; um aplicativo *cliente* é um programa que foi vinculado às biblioteca do cliente WebSphere MQ necessárias. Consulte “Construindo aplicativos para clientes MQI do WebSphere MQ” na página 362

e “Construindo um aplicativo IBM WebSphere MQ” na página 435 para obter detalhes sobre compilar e vincular seus programas.

- O servidor de banco de dados pode residir em uma máquina diferente do servidor do gerenciador de filas, contanto que o cliente de banco de dados esteja instalado na mesma máquina que o gerenciador de filas e ela suporte essa função. Consulte a documentação do produto do banco de dados para determinar se seu software cliente pode ser usado para sistemas two-phase commit.
- Embora o gerenciador de filas se comporte como um gerenciador de recursos (para fins de envolvimento nas unidades globais de trabalho do Cenário 2), não será possível fazer um gerenciador de filas coordenar outro gerenciador de filas dentro das unidades globais de trabalho do Cenário 1.

Arquivos de carregamento do comutador

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

O arquivo de carregamento do comutador é uma biblioteca compartilhada (uma DLL em sistemas Windows) que é carregada pelo código em seu aplicativo IBM WebSphere MQ e o gerenciador de filas. Seu objetivo é simplificar o carregamento da biblioteca compartilhada do cliente do banco de dados e para retornar os ponteiros para as funções XA.

Os detalhes do arquivo de carregamento do comutador devem ser especificados antes do gerenciador de filas ser iniciado. Os detalhes são colocados no arquivo qm.ini em sistemas Windows, UNIX and Linux

- Em sistemas Windows e Linux (plataformasx86 e x86-64), use IBM WebSphere MQ Explorer para atualizar o arquivo qm.ini .
- Em todos os outros sistemas edite o arquivo, qm.ini, diretamente.

A origem C para o arquivo de carregamento do comutador é fornecido com a instalação do IBM WebSphere MQ se suportar as unidades globais de Cenário 1 de trabalho. A origem contém uma função chamada MQStart. Quando o arquivo de carregamento do comutador for carregado, o gerenciador de filas chamará essa função, que retornará o endereço de uma estrutura denominada como um *comutador XA*.

A estrutura do comutador XA existe na biblioteca compartilhada do cliente de banco de dados e contém um número de ponteiros de funções, conforme descrito em Tabela 6 na página 47:

Tabela 6. Ponteiros de Funções do Comutador XA

Nome do ponteiro de função	**Funções XA**	**Finalidade**
xa_open_entry	xa_open	Conecta-se ao banco de dados
xa_close_entry	xa_close	Desconecta-se do banco de dados
xa_start_entry	xa_start	Inicia uma ramificação de uma unidade global de trabalho
xa_end_entry	xa_end	Suspende uma ramificação de uma unidade global de trabalho
xa_rollback_entry	xa_rollback	Retrocede uma ramificação de uma unidade global de trabalho
xa_prepare_entry	xa_prepare	Prepara para confirmar uma ramificação de uma unidade global de trabalho
xa_commit_entry	xa_commit	Confirma uma ramificação de uma unidade global de trabalho

Tabela 6. Ponteiros de Funções do Comutador XA (continuação)		
Nome do ponteiro de função	**Funções XA**	**Finalidade**
xa_recover_entry	xa_recover	Descobre do banco de dados se possua uma unidade de trabalho indeterminado
xa_forget_entry	xa_forget	Permite que um banco de dados esqueça uma ramificação de uma unidade global de trabalho
xa_complete_entry	xa_complete	Conclui uma ramificação de uma unidade global de trabalho

Durante a primeira chamada MQBEGIN em seu aplicativo, o código do IBM WebSphere MQ que é executado como parte do MQBEGIN carregará o arquivo de carregamento do comutador e chamará a função xa_open na biblioteca compartilhada do banco de dados. Da mesma forma, durante a inicialização do gerenciador de filas e em outras ocasiões subsequentes, alguns processos de gerenciador de filas carregarão o arquivo de carregamento do comutador e o chamarão de xa_open.

Você pode reduzir o número de chamadas xa_* usando o *registro dinâmico*. Para obter uma descrição completa dessa técnica de otimização, consulte “Registro dinâmico de XA” na página 68.

Configurando seu sistema para a coordenação do banco de dados

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Há várias tarefas que você deve executar antes de um gerenciador de banco de dados poder participar em unidades globais de trabalhos coordenados pelo gerenciador de filas. Serão descritas aqui conforme a seguir:

- “Instalando e configurando o produto do banco de dados” na página 48
- “Criando arquivos de carregamento do comutador” na página 49
- “Incluindo informações de configuração no gerenciador de filas” na página 50
- “Gravando e modificando seus aplicativos” na página 51
- “Testando o sistema” na página 52

Instalando e configurando o produto do banco de dados
Para instalar e configurar seu produto de banco de dados, consulte a documentação própria do produto. Estes tópicos nesta seção descrevem problemas gerais de configuração e como eles estão relacionados à interoperação entre o WebSphere MQ e o banco de dados.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Conexões com o Banco de Dados

Um aplicativo que estabelece uma conexão padrão com o gerenciador de filas estará associado a um encadeamento em um processo do agente do gerenciador de filas locais separadas. (Uma conexão que não é uma conexão *atalho* é uma conexão *padrão* nesse contexto. Para obter mais informações, consulte “Conectando-se a um gerenciador de filas usando a chamada MQCONNX” na página 212.

Quando o aplicativo emite MQBEGIN, ele e o processo do agente chamam a função xa_open na biblioteca do cliente de banco de dados. Em resposta a isso, o código da biblioteca do cliente do banco de dados *se conecta* com o banco de dados que deverá ser envolvido na unidade de trabalho do *a partir de dos aplicativos e processos do gerenciador de filas*. Essas conexões com o banco de dados serão mantidas enquanto o aplicativo permanecer conectado ao gerenciador de filas.

Essa é uma consideração importante se o banco de dados suportar apenas um número limitado de usuários ou conexões, porque duas conexões estão sendo feitas ao banco de dados para suportar um programa de aplicativo.

Configuração de Cliente/Servidor

A biblioteca do cliente de banco de dados carregada no WebSphere MQ gerenciador de filas e processos de aplicativos **deve** ser capaz de enviar e receber de seu servidor. Assegure-se de que:

- Os arquivos de configuração cliente/servidor do banco de dados possuam os detalhes corretos
- As variáveis de ambiente relevante estão configuradas no ambiente do gerenciador de filas **e** no processos de aplicativos

Criando arquivos de carregamento do comutador
WebSphere MQ vem com um makefile de amostra, usado para construir arquivos de carregamento do comutador para os gerenciadores de banco de dados suportados.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

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

O makefile de amostra, juntamente com todos os arquivos necessários de origem C associados para construir os arquivos de carregamento do comutador, será instalado nos diretórios a seguir:

- Para WebSphere MQ para Windows, no diretório *MQ_INSTALLATION_PATH*\tools\c\samples\xatm\
- Para sistemas WebSphere MQ para UNIX and Linux , no diretório *MQ_INSTALLATION_PATH*/samp/xatm/

Os módulos de origem de amostra usados para construir os arquivos de carregamento do comutador são:

- Para DB2, db2swit.c
- Para o Oracle, oraswit.c
- Para Informix, infswit.c
- Para o Sybase, sybswit.c

Ao gerar os arquivos de carregamento do comutador, instale os arquivos de carregamento do comutador de 32 bits no /var/mqm/exits e instale os arquivos de carregamento do comutador de 64 bits no /var/mqm/exits64.

Se você tiver os gerenciadores de filas de 32 bits, então o arquivo make de amostra, xaswit.mak, instalará um arquivo de carregamento do comutador de 32 bits no/var/mqm/exits.

Se você tiver os gerenciadores de filas de 64 bits, então o arquivo make de amostra, xaswit.mak, instalará um arquivo de carregamento do comutador de 32 bits no/var/mqm/exits e um arquivo de carregamento do comutador de 64 bits no /var/mqm/exits64.

Segurança do Arquivo

É possível que seu sistema operacional possa falhar o carregamento do arquivo de carregamento do comutador pelo WebSphere MQ, por razões fora do controle do WebSphere MQ.. Se isso ocorrer e mensagens de erro forem gravadas nos logs de erro do WebSphere MQ e potencialmente a chamada MQBEGIN pode falhar. Para ajudar a garantir que seu sistema operacional não falhe o carregamento do arquivo de carregamento do comutador, você deverá atender aos requisitos a seguir:

1. O arquivo de carregamento do comutador deverá estar disponível no local que é fornecido no arquivo qm.ini.
2. O arquivo de carregamento do comutador deve ser acessível a todos os processos que precisarem carregá-lo, incluindo os processos do gerenciador de filas e os processos do aplicativos.

3. Todas as bibliotecas no qual o arquivo de carregamento do comutador depende, incluindo as bibliotecas que são fornecidas pelo produto de banco de dados, deverão estar presentes e acessíveis.

Incluindo informações de configuração no gerenciador de filas
Quando você tiver criado um arquivo de carregamento do comutador para o seu gerenciador de banco de dados e o colocou em um local seguro, será necessário especificar aquele local para seu gerenciador de filas.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Para especificar o local, execute as etapas a seguir:

- Nos sistemas Windows e Linux (plataformasx86 e x86-64), use o WebSphere MQ Explorer. Especifique os detalhes do arquivo de carregamento do comutador no painel de propriedades do gerenciador de filas, sob o gerenciador de recursos XA.
- Em todos os outros sistemas, especifique os detalhes do arquivo de carregamento do comutador na sub-rotina XAResourceManager no arquivo qm.ini do gerenciador de filas.

Inclua uma sub-rotina XAResourceManager no banco de dados que seu gerenciador de filas irá coordenar. O caso mais comum é para que haja apenas um banco de dados, e portanto, apenas uma sub-rotina XAResourceManager. Para obter detalhes sobre configurações mais complicadas que envolvam bancos de dados múltiplos, consulte “Diversas configurações do banco de dados” na página 62. Os atributos da sub-rotina XAResourceManager são os seguintes:

Name=name
A sequência escolhida pelo usuário que identifica o gerenciador de recursos. Com efeito, fornecerá um nome para a sub-rotina XAResourceManager. O nome é obrigatório e pode ter até 31 caracteres de comprimento.

O nome escolhido deve ser exclusivo; deve haver apenas uma sub-rotina XAResourceManager com esse nome nesse arquivo qm.ini. O nome também deve ser significativo, porque o gerenciador de filas o usará para se referir a esse gerenciador de recursos em ambas as mensagens de log de erros do gerenciador de filas e na saída quando o comando `dspmqtrn` for usado. (Consulte “Exibindo Unidades de Trabalho Pendentes com o Comando dspmqtrn” na página 64 para obter mais informações.)

Após ter escolhido um nome e iniciado o gerenciador de filas, não altere o atributo Nome. Para obter mais detalhes sobre como alterar as informações de configuração, consulte “Alterando Informações de Configuração” na página 67.

SwitchFile=name
Esse é o nome do arquivo de carregamento do comutador XA que foi construído anteriormente. Esse é um atributo obrigatório. O código no gerenciador de filas e os processos do aplicativo WebSphere MQ tentam carregar o arquivo de carregamento do comutador em duas ocasiões:

1. Na inicialização do gerenciador de filas
2. Quando você faz a primeira chamada para MQBEGIN em seu processo de aplicativo WebSphere MQ

Os atributos de permissões e segurança do arquivo de carregamento do comutador deverão permitir que esses processos executem essa ação.

XAOpenString=string
Esta é uma sequência de dados que o código do WebSphere MQ transmite em suas chamadas para a função xa_open do gerenciador do banco de dados. Esse é um atributo opcional; se for omitido uma sequência de comprimento zero será assumida.

O código no gerenciador de filas e os processos do aplicativo WebSphere MQ chamam a função xa_open em duas ocasiões:

1. Na inicialização do gerenciador de filas

2. Quando você faz a primeira chamada para MQBEGIN em seu processo de aplicativo WebSphere MQ

O formato para essa cadeia é específica para cada produto de banco de dados e será descrita na documentação para esse produto. Em geral, a sequência xa_open contém informações de autenticação (nome e senha do usuário) para permitir uma conexão com o banco de dados em ambos o gerenciador de filas e os processos de aplicativos.

XACloseString=string

Essa é uma sequência de dados que o código do WebSphere MQ transmite em suas chamadas para a função xa_close do gerenciador do banco de dados. Esse é um atributo opcional; se for omitido uma sequência de comprimento zero será assumida.

O código no gerenciador de filas e os processos do aplicativo WebSphere MQ chamam a função xa_close em duas ocasiões:

1. Na inicialização do gerenciador de filas
2. Quando você faz uma chamada para MQDISC em seu processo de aplicativo do WebSphere MQ , tendo feito anteriormente uma chamada para MQBEGIN

O formato para essa cadeia é específica para cada produto de banco de dados e será descrita na documentação para esse produto. Em geral, a cadeia está vazia e é comum omitir o atributo XACloseString da sub-rotina XAResourceManager.

ThreadOfControl=THREAD |PROCESS

O valor ThreadOfControl poderá ser THREAD ou PROCESS. O gerenciador de filas usa-o para propósitos de serialização. Esse é um atributo opcional; se for omitido, o valor PROCESS será assumido.

Se o código do cliente do banco de dados permitir que os encadeamentos chamem as funções XA sem serialização, o valor para ThreadOfControl poderá ser THREAD. O gerenciador de filas assume que ele possa chamar as funções XA na biblioteca compartilhada do cliente de banco de dados a partir de vários encadeamentos ao mesmo tempo, se necessário.

Se o código do cliente do banco de dados não permitir que os encadeamentos chamem suas funções XA dessa maneira, o valor para ThreadOfControl deverá ser PROCESS. Nesse caso, o gerenciador de filas serializa todas as chamadas para a biblioteca compartilhada do cliente de banco de dados para que somente uma chamada de cada vez seja feita de dentro de um determinado processo. Você provavelmente também precise assegurar-se de que seu aplicativo desempenhe a serialização semelhante, se for executado com múltiplos encadeamentos.

Observe que esse problema, da capacidade do produto de banco de dados de lidar com os processos multiencadeados dessa forma, será um problema para o fornecedor desse produto. Consulte a documentação do produto de banco de dados para obter detalhes se você pode configurar o atributo ThreadOfControl para THREAD ou PROCESS. Recomendamos que, se puder, configure ThreadOfControl como THREAD. Se estiver em dúvida, a opção mais *segurança* será configurar PROCESS, embora você perca os benefícios potenciais de desempenho usando o THREAD.

Gravando e modificando seus aplicativos

Como implementar uma unidade global de trabalho.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Os programas de aplicativos de amostra para unidades globais de trabalho do Cenário 1 que são fornecidas com uma instalação do WebSphere MQ são descritos no “Introduzindo Unidades de Trabalho” na página 42.

Em geral, uma unidade global de trabalho é implementada em um aplicativo pelo método a seguir (em pseudocódigo):

```
MQBEGIN
MQGET
```

MQPUT
SQL INSERT
MQCMIT

O propósito do MQBEGIN é denotar o início de uma unidade global de trabalho. O propósito do MQCMIT é denotar o final da unidade global de trabalho e concluí-la, com todos os gerenciadores de recursos participantes, usando o protocolo two-phase commit.

Entre MQBEGIN e MQCMIT, o gerenciador de filas não faz nenhuma chamada para o banco de dados para atualizar seus recursos. Ou seja, a única maneira de alterar tabelas de um banco de dados é por meio do seu código (por exemplo, SQL INSERT no pseudocódigo).

A função do gerenciador de filas, no que se refere ao banco de dados, é indicar isso quando uma unidade global de trabalho foi iniciada, quando foi concluída e se ela deve ser confirmada ou retrocedida.

Até onde a sua aplicação esteja aos cuidados, o gerenciador de filas executará duas funções: um gerenciador de recursos (no qual os recursos serão as mensagens em filas) e o gerenciador de transações para a unidade global de trabalho.

Inicie com os programas de amostra fornecidos e trabalhe com as várias chamadas de API do WebSphere MQ e do banco de dados que estão sendo feitas nesses programas. As chamadas API em questão são totalmente documentadas em “Programas de amostra do WebSphere MQ” na página 98, Tipos de dados usados no MQIe (no caso da própria API do banco de dados) na própria documentação do banco de dados.

Testando o sistema
Você saberá se seu aplicativo e sistema estão configurados corretamente apenas executando-os durante o teste. Você poderá testar a configuração do sistema (a comunicação bem-sucedida entre o gerenciador de filas e o banco de dados), construindo e executando um dos programas de amostra fornecidos.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Configurando Db2
Informações de suporte e configuração do DB2 .

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Os níveis suportados do Db2 são definidos na página IBM WebSphere MQ requisitos detalhados do sistema .

Nota: Instâncias de 32 bits do Db2 não são suportadas em plataformas em que o gerenciador de filas é de 64 bits.

Execute as ações a seguir:

1. Verifique as configurações de variável de ambiente.
2. Crie o arquivo de carregamento do comutador Db2 .
3. Inclua informações de configuração do gerenciador de recursos.
4. Altere os parâmetros de configuração do Db2 se necessário.

Leia essas informações em conjunção com as informações gerais fornecidas em “Configurando seu sistema para a coordenação do banco de dados” na página 48.

Aviso: Se você executar `db2profile` nas plataformas UNIX and Linux, a variável de ambiente LIBPATH e LD_LIBRARY_PATH serão configuradas. É aconselhável `unset` essas variáveis de ambiente. Consulte o Guia de *Iniciação Rápida* apropriado.

Verificando as configurações da variável de ambiente Db2

Assegure-se de que suas variáveis de ambiente do Db2 estejam configuradas para processos do gerenciador de filas ***, bem como em*** seus processos de aplicativo. Em particular, você deve sempre configurar a variável de ambiente DB2INSTANCE ***antes*** de iniciar o gerenciador de filas. A variável de ambiente DB2INSTANCE identifica a instância Db2 contendo os bancos de dados Db2 que estão sendo atualizados. Por exemplo:

- Em sistemas UNIX and Linux , use:

```
export DB2INSTANCE=db2inst1
```

- Em sistemas Windows , use:

```
set DB2INSTANCE=DB2
```

No Windows com um banco de dados Db2 , deve-se incluir o usuário MUSR_MQADMIN no grupo DB2USERS para permitir que o gerenciador de filas seja iniciado.

Criando o arquivo de carregamento do comutador Db2

A maneira mais fácil de criar o arquivo de carregamento do comutador Db2 é usar o arquivo de amostra xaswit.mak, que o WebSphere MQ fornece para construir os arquivos de carregamento do comutador para uma variedade de produtos de banco de dados.

Em sistemas Windows , é possível localizar xaswit.mak no diretório *MQ_INSTALLATION_PATH*`\tools\c\samples\xatm`. *MQ_INSTALLATION_PATH* Representa o diretório de alto nível no qual o WebSphere MQ está instalado Para criar o arquivo de carregamento do comutador Db2 com o Microsoft Visual C + +, use:

```
nmake /f xaswit.mak db2swit.dll
```

O arquivo de comutador gerado é colocado em `c:\Program Files\IBM\WebSphere MQ\exits`..

É possível localizar xaswit.mak no diretório *MQ_INSTALLATION_PATH*`/samp/xatm`. *MQ_INSTALLATION_PATH* Representa o diretório de alto nível no qual o WebSphere MQ está instalado

Edite xaswit.mak para *remover o comentário* das linhas apropriadas para a versão do Db2 que você está usando: Em seguida, execute o makefile usando o comando:

```
make -f xaswit.mak db2swit
```

O arquivo de carregamento do comutador gerado de 32 bits será colocado em `/var/mqm/exits`.

O arquivo de carregamento do comutador gerado de 64 bits será colocado em `/var/mqm/exits64`.

Incluindo informações de configuração do gerenciador de recursos para Db2

Deve-se modificar as informações de configuração para o gerenciador de filas para declarar Db2 como um participante em unidades globais de trabalho. A modificação de informações de configuração dessa forma está descrito com mais detalhes em “Incluindo informações de configuração no gerenciador de filas” na página 50.

- Nos sistemas Windows e Linux (plataformasx86 e x86-64), use o WebSphere MQ Explorer. Especifique os detalhes do arquivo de carregamento do comutador no painel de propriedades do gerenciador de filas, sob o gerenciador de recursos XA.
- Em todos os outros sistemas, especifique os detalhes do arquivo de carregamento do comutador na sub-rotina XAResourceManager no arquivo qm.ini do gerenciador de filas.

Figura 9 na página 54 é uma amostra UNIX , mostrando uma entrada XAResourceManager na qual o banco de dados a ser coordenado é chamado mydbname, sendo esse nome especificado no XAOpenString:

```
XAResourceManager:
  Name=mydb2
  SwitchFile=db2swit
  XAOpenString=mydbname,myuser,mypasswd,toc=t
  ThreadOfControl=THREAD
```

Figura 9. Entrada XAResourceManager de amostra para Db2 em plataformas UNIX

Nota:

1. ThreadOfControl=THREAD não pode ser usado com Db2 versões anteriores à versão 8. Configure ThreadOfControl e o parâmetro toc do XAOpenString para uma das combinações a seguir:
   - ThreadOfControl=THREAD e toc=t
   - ThreadOfControl=PROCESS e toc=p

   Se estiver usando o arquivo de carregamento do comutador jdbcdb2 XA para ativar a coordenação JDBC/JTA, será necessário usar ThreadOfControl=PROCESS e toc=p.

Mudando os parâmetros de configuração do Db2

Para cada banco de dados Db2 que o gerenciador de filas está coordenando, deve-se configurar privilégios do banco de dados, mudar o parâmetro tp_mon_name e reconfigurar o parâmetro maxappls. Para fazer isso, execute as etapas a seguir:

Configure os privilégios do banco de dados

Os processos do gerenciador de filas executados com o mqm de usuário e grupo efetivos em sistemas UNIX and Linux. Em sistemas Windows , eles são executados como o usuário que iniciou o gerenciador de filas Esse pode ser um de:

1. O usuário que emitiu o comando strmqm ou
2. O usuário sob o qual o servidor COM do IBM MQSeries Service é executado

Por padrão, esse usuário é chamado de MUSR_MQADMIN.

Se você não tiver especificado um nome de usuário e senha na sequência xa_open, **o usuário sob o qual o gerenciador de filas está em execução** será usado pelo Db2 para autenticar a chamada xa_open. Se esse usuário (por exemplo, o usuário mqm em sistemas UNIX and Linux) não possuir os privilégios mínimos no banco de dados, o banco de dados se recusará a autenticar a chamada xa_open.

As mesmas considerações se aplicarão a seu processo de aplicativo. Se você não tiver especificado um nome de usuário e senha na sequência xa_open, o usuário sob o qual seu aplicativo está em execução será usado pelo Db2 para autenticar a chamada xa_open que é feita durante o primeiro MQBEGIN. Novamente, esse usuário deverá possuir privilégios mínimos no banco de dados para que isso funcione.

Por exemplo, forneça a autoridade de conexão do usuário mqm no banco de dados mydbname emitindo os comandos Db2 a seguir:

```
db2 connect to mydbname
db2 grant connect on database to user mqm
```

Consulte “Considerações sobre Segurança” na página 63 para obter mais informações sobre a segurança.

Windows **Mude o parâmetro TP_MON_NAME**

Para Db2 apenas para sistemas Windows , mude o parâmetro de configuração TP_MON_NAME para nomear a DLL que o Db2 usa para chamar o gerenciador de filas para registro dinâmico..

Use o comando `db2 update dbm cfg using TP_MON_NAME mqmax` para nomear MQMAX.DLL como a biblioteca que o Db2 usa para chamar o gerenciador de filas. Isso deve estar presente em um diretório no PATH.

Reconfigure o parâmetro maxappls

Talvez seja necessário rever a configuração para o parâmetro *maxappls*, o que limita o número máximo de aplicativos que poderá ser conectado a um banco de dados. Consulte “Instalando e configurando o produto do banco de dados” na página 48.

Configurando o Oracle

Informações de configuração e suporte do Oracle.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Conclua as etapas a seguir:

1. Verifique as configurações de variável de ambiente.
2. Crie o arquivo de carregamento do comutador Oracle.
3. Inclua informações de configuração do gerenciador de recursos.
4. Altere os parâmetros de configuração do Oracle, se necessário.

Uma lista atual de níveis de Oracle suportado pelo IBM WebSphere MQ será fornecido na página IBM WebSphere MQ requisitos detalhados do sistema.

Verificando as Configurações de Variável de Ambiente do Oracle

Assegure-se de que as variáveis de ambiente do Oracle estão configuradas com os processos do gerenciador de filas, bem como nos processos de aplicativos. Em particular, sempre configure as variáveis de ambiente a seguir, antes de iniciar o gerenciador de filas:

ORACLE_HOME

O diretório inicial do Oracle. Por exemplo, em sistemas UNIX and Linux , use:

```
export ORACLE_HOME=/opt/oracle/product/8.1.6
```

Em sistemas Windows , use:

```
set ORACLE_HOME=c:\oracle\ora81
```

ORACLE_SID

O Oracle SID está sendo usado. Se você estiver usando o Net8 para a conectividade do cliente/servidor, talvez não seja necessário configurar essa variável de ambiente. Consulte a documentação do Oracle.

O exemplo subsequente é um exemplo de como configurar essa variável de ambiente, em sistemas UNIX and Linux:

```
export ORACLE_SID=sid1
```

O equivalente em sistemas Windows é:

```
set ORACLE_SID=sid1
```

Nota: A variável de ambiente PATH deve ser configurada para incluir o diretório de binários (por exemplo, `ORACLE_INSTALL_DIR/VERSION/32BIT_NAME/bin` ou `ORACLE_INSTALL_DIR/VERSION/64BIT_NAME/bin`), caso contrário, poderá ser exibida uma mensagem informando que as bibliotecas oraclient estão ausentes da máquina.

Se você executar gerenciadores de filas em sistemas Windows de 64 bits, ambos os clientes Oracle de 64 bits e 32 bits deverão ser instalados Deve-se instalar ambos os clientes porque o gerenciador de filas é executado como processos de 32 bits que usam um arquivo de carregamento do comutador de 32 bits, que por sua vez deve iniciar um dll do cliente Oracle de 32 bits.

O arquivo de carregamento do comutador, carregado por gerenciadores de filas de 64 bits, deverá acessar as bibliotecas do cliente Oracle de 64 bits. Os gerenciadores de filas de 32 bits devem acessar o cliente Oracle de 32 bits quando o IBM WebSphere MQ estiver em execução em um sistema de 64 bits do Windows

Criando o Arquivo de Carregamento do Comutador Oracle

Para criar o arquivo de carregamento do comutador Oracle, use o arquivo de amostra `xaswit.mak`, que o IBM WebSphere MQ fornece para construir os arquivos de carregamento de comutador para vários produtos de banco de dados. Em sistemas Windows , é possível localizar `xaswit.mak` no diretório `C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm`.. Para criar o arquivo de carregamento do comutador Oracle com Microsoft Visual C + +, use: `nmake /f xaswit.mak oraswit.dll`

O arquivo de comutador gerado é colocado em *MQ_INSTALLATION_PATH*`\exits`.*MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual IBM WebSphere MQ está instalado.

Você pode localizar `xaswit.mak` no diretório *MQ_INSTALLATION_PATH*`/samp/xatm`. *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual IBM WebSphere MQ está instalado.

Edite `xaswit.mak` para remover o comentário das linhas apropriadas para a versão do Oracle que você estiver usando. Em seguida, execute o makefile usando o comando:

```
make -f xaswit.mak oraswit
```

O arquivo de carregamento do comutador de 32 bits gerado será colocado em `/var/mqm/exits`.

O arquivo de carregamento do comutador de 64 bits gerado será colocado em `/var/mqm/exits64`.

Incluindo Informações de Configuração do Gerenciador de Recursos para o Oracle

Você deve modificar as informações de configuração para o gerenciador de filas para declarar o Oracle como um participante em unidades globais de trabalho. Modificando as informações de configuração no gerenciador de filas, dessa maneira será descrito com mais detalhes em “Incluindo informações de configuração no gerenciador de filas” na página 50.

- Nos sistemas Windows e Linux (x86 e x86-64 plataformas), use IBM WebSphere MQ Explorer. Especifique os detalhes do arquivo de carregamento do comutador no painel de propriedades do gerenciador de filas, sob o gerenciador de recursos XA.
- Em todos os outros sistemas, especifique os detalhes do arquivo de carregamento do comutador na sub-rotina XAResourceManager no arquivo `qm.ini` do gerenciador de filas.

O Figura 10 na página 57 é uma amostra de sistemas do UNIX and Linux mostrando uma entrada XAResourceManager. Você deve incluir um `LogDir` na sequência aberta do XA de forma que todos os erros e as informações de rastreio sejam registrados no mesmo local.

```
XAResourceManager:
  Name=myoracle
  SwitchFile=oraswit
  XAOpenString=Oracle_XA+Acc=P/myuser/mypasswd+SesTm=35+LogDir=/tmp+threads=true
  ThreadOfControl=THREAD
```

Figura 10. Entrada do XAResourceManager de Amostra para o Oracle em Plataformas UNIX and Linux

Nota:

1. No Figura 10 na página 57, a sequência xa_open foi usada com quatro parâmetros. Os parâmetros adicionais podem ser incluídos, conforme descrito na documentação do Oracle.
2. Ao usar o parâmetro IBM WebSphere MQ `ThreadOfControl=THREAD`, você deverá usar o parâmetro do Oracle `+threads=true` na sub-rotina XAResourceManager.

Consulte o *Oracle8 Server Application Developer's Guide* para obter mais informações sobre a sequência xa_open.

Alterando os Parâmetros de Configuração do Oracle

Para cada banco de dados Oracle que o gerenciador de filas esteja coordenando, você deverá revisar o máximo de sessões e configurar os privilégios do banco de dados. Para fazer isso, conclua essas etapas:

Revise o máximo de sessões
Talvez seja necessário revisar suas configurações LICENSE_MAX_SESSIONS e PROCESSOS para levar em conta as conexões adicionais necessárias pelos processos pertencentes ao gerenciador de filas. Consulte “Instalando e configurando o produto do banco de dados” na página 48 para obter mais detalhes.

Configure os privilégios do banco de dados
O nome de usuário do Oracle especificado na sequência xa_open deverá possuir os privilégios para acessar a visualização DBA_PENDING_TRANSACTIONS, conforme descrito na documentação do Oracle.

O privilégio necessário poderá ser fornecido usando o comando de exemplo a seguir:

```
grant select on DBA_PENDING_TRANSACTIONS to myuser;
```

Configurando o Informix

Informações de suporte e de configuração do Informix

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Conclua as etapas a seguir:

1. Assegure-se de ter instalado o SDK do cliente Informix apropriado:
   - Gerenciadores de filas e aplicativos de 32 bits requerem um SDK do cliente Informix de 32 bits.
   - Gerenciadores de filas e aplicativos de 64 bits requerem um SDK do cliente Informix de 64 bits.
2. Assegure-se de que os bancos de dados Informix sejam criados corretamente.
3. Verifique as configurações de variável de ambiente.
4. Construa o arquivo de carregamento do comutador do Informix .
5. Inclua informações de configuração do gerenciador de recursos.

Uma lista atual de níveis de Informix suportados pelo WebSphere MQ é fornecida na página doIBM WebSphere MQ requisitos detalhados do sistema

Assegurando que os bancos de dados Informix sejam criados corretamente

Cada banco de dados Informix que deve ser coordenado por um gerenciador de fila WebSphere MQ deve ser criado especificando o parâmetro `log` . Por exemplo:

```
create database mydbname with log;
```

Os gerenciadores de fila do WebSphere MQ não podem coordenar os bancos de dados Informix que não possuem o parâmetro `log` especificado na criação Se um gerenciador de fila tentar coordenar um banco de dados Informix que não tenha o parâmetro `log` especificado na criação, a chamada xa_open para o Informix falhará e vários erros FFST serão gerados

Verificando as configurações de variáveis de ambiente do Informix

Assegure-se de que suas variáveis de ambiente do Informix estejam configuradas para os processos do gerenciador de fila ***, bem como em*** seus processos de aplicativos Em particular, sempre configure as variáveis de ambiente a seguir **antes** de iniciar o gerenciador de filas:

INFORMIXDIR
O diretório da instalação do produto Informix ..

- Para aplicativos UNIX and Linux de 32 bits, use o comando a seguir:

  ```
  export INFORMIXDIR=/opt/informix/32-bit
  ```

- Para aplicativos UNIX and Linux de 64 bits, use o comando a seguir:

  ```
  export INFORMIXDIR=/opt/informix/64-bit
  ```

- Para aplicativos Windows , use o comando a seguir:

  ```
  set INFORMIXDIR=c:\informix
  ```

Para sistemas que possuem gerenciadores de fila de 64 bits que devem suportar aplicativos de 32 bits e 64 bits, é necessário que os SDKs do cliente de 32 bits e 64 bits do Informix estejam instalados. O makefile de amostra `xaswit.mak`, usado na criação de um arquivo de carregamento do comutador também configura os diretórios de instalação do produto.

INFORMIXSERVER
O nome do servidor Informix .. Por exemplo, em sistemas UNIX and Linux , use:

```
export INFORMIXSERVER=hostname_1
```

Em sistemas Windows , use:

```
set INFORMIXSERVER=hostname_1
```

ONCONFIG
O nome do arquivo de configuração do Informix . Por exemplo, em sistemas UNIX and Linux , use:

```
export ONCONFIG=onconfig.hostname_1
```

Em sistemas Windows , use:

```
set ONCONFIG=onconfig.hostname_1
```

Criando o arquivo de carregamento do comutador Informix

Para criar o arquivo de carregamento do comutador Informix , use o arquivo de amostra xaswit.mak, que o WebSphere MQ fornece para construir os arquivos de carregamento do comutador para vários produtos de banco de dados Em sistemas Windows , é possível localizar xaswit.mak no diretório *MQ_INSTALLATION_PATH*\tools\c\samples\xatm. *MQ_INSTALLATION_PATH* Representa o diretório de alto nível no qual o WebSphere MQ está instalado Para criar o arquivo de carregamento do comutador Informix com Microsoft Visual C + +, use:

```
nmake /f xaswit.mak infswit.dll
```

O arquivo de comutador gerado é colocado em c:\Program Files\IBM\WebSphere MQ\exits..

É possível localizar xaswit.mak no diretório *MQ_INSTALLATION_PATH*/samp/xatm. *MQ_INSTALLATION_PATH* Representa o diretório de alto nível no qual o WebSphere MQ está instalado

Edite xaswit.mak para *remover o comentário* das linhas apropriadas para a versão do Informix que você está usando Em seguida, execute o makefile usando o comando:

```
make -f xaswit.mak infswit
```

O arquivo de carregamento do comutador de 32 bits gerado será colocado em /var/mqm/exits.

O arquivo de carregamento do comutador de 64 bits gerado será colocado em /var/mqm/exits64.

Incluindo informações de configuração do gerenciador de recursos para Informix

Você deve modificar as informações de configuração do gerenciador de filas para declarar Informix como um participante em unidades globais de trabalho. Modificando as informações de configuração no gerenciador de filas, dessa maneira será descrito com mais detalhes em "Incluindo informações de configuração no gerenciador de filas" na página 50.

- Nos sistemas Windows e Linux (plataformasx86 e x86-64), use o WebSphere MQ Explorer. Especifique os detalhes do arquivo de carregamento do comutador no painel de propriedades do gerenciador de filas, sob o gerenciador de recursos XA.
- Em todos os outros sistemas, especifique os detalhes do arquivo de carregamento do comutador na sub-rotina XAResourceManager no arquivo qm.ini do gerenciador de filas.

Figura 11 na página 59 é uma amostra UNIX , mostrando uma entrada qm.ini XAResourceManager em que o banco de dados a ser coordenado é chamado mydbname, esse nome sendo especificado no XAOpenString:

```
XAResourceManager:
  Name=myinformix
  SwitchFile=infswit
  XAOpenString=DB=mydbname@myinformixserver\;USER=myuser\;PASSWD=mypasswd
  ThreadOfControl=THREAD
```

Figura 11. Entrada XAResourceManager de amostra para Informix em plataformas UNIX

Nota: Por padrão, a amostra xaswit.mak em plataformas UNIX cria um arquivo de carregamento do comutador que usa bibliotecas Informix encadeadas.. Você deve assegurar que o Controle ThreadOfseja configurado como THREAD ao usar essas bibliotecas do Informix No Figura 11 na página 59, o ThreadOfControl de atributo da sub-rotina XAResourceManager do arquivo qm.ini será configurado como THREAD. Quando THREAD é especificado, os aplicativos devem ser construídos usando as bibliotecas encadeadas do Informix e as bibliotecas da API encadeada do WebSphere MQ ..

O atributo XAOpenString deve conter o nome do banco de dados, seguido pelo símbolo @ e, em seguida, seguido pelo nome do servidor Informix

Para usar as bibliotecas não encadeadas do Informix , deve-se assegurar que o qm.ini arquivo XAResourceManager atributo de sub-rotina ThreadOfControl esteja configurado como PROCESS Você também deve fazer as alterações a seguir para o xaswit.mak de amostra:

1. Remova o comentário da geração de um arquivo de carregamento de comutador não encadeado.
2. Comente a linha de geração do arquivo de carregamento do comutador encadeado.

Configuração do Sybase

Informações de configuração e suporte do Sybase.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Conclua as etapas a seguir:

1. Assegure-se de que tenha instalado as bibliotecas XA do Sybase, por exemplo, instalando a opção DTM do XA.
2. Verifique as configurações de variável de ambiente.
3. Ative o suporte XA do Sybase.
4. Crie o arquivo de carregamento do comutador do Sybase.
5. Inclua informações de configuração do gerenciador de recursos.

Uma lista atual de níveis de Sybase suportados pelo WebSphere MQ é fornecida na página IBM WebSphere MQ requisitos detalhados do sistema

Verificando as Configurações de Variável de Ambiente do Sybase

Assegure-se de que as variáveis de ambiente do Sybase estão configurados nos processos do gerenciador de filas ***, bem como nos*** processos de aplicativos. Em particular, sempre configure as variáveis de ambiente a seguir **antes** de iniciar o gerenciador de filas:

SYBASE
O local da instalação do produto do Sybase. Por exemplo, em sistemas UNIX and Linux , use:

```
export SYBASE=/sybase
```

Em sistemas Windows , use:

```
set SYBASE=c:\sybase
```

SYBASE_OCS
O diretório sob o SYBASE no qual você instalou os arquivos do cliente Sybase. Por exemplo, em sistemas UNIX and Linux , use:

```
export SYBASE_OCS=OCS-12_0
```

Em sistemas Windows , use:

```
set SYBASE_OCS=OCS-12_0
```

Ativando o Suporte XA do Sybase

Dentro do arquivo de configuração Sybase XA $SYBASE/$SYBASE_OCS/`xa_config`, defina um Logical Resource Manager (LRM) para cada conexão com o servidor Sybase que está sendo atualizado. Um exemplo do conteúdo de $SYBASE/$SYBASE_OCS/`xa_config` será mostrado em Figura 12 na página 61.

```
# The first line must always be a comment

[xa]

  LRM=lrmname
  server=servername
```

Figura 12. Conteúdos de Exemplos de $SYBASE/$SYBASE_OCS/xa_config

Criando o Arquivo de Carregamento do Comutador do Sybase

Para criar o arquivo de carregamento do comutador Sybase , use os arquivos de amostra fornecidos com WebSphere MQ.Em sistemas Windows , é possível localizar xaswit.mak no diretório `C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm`.. Para criar o arquivo de carregamento do comutador Sybase com o Microsoft Visual C + +, use:

```
nmake /f xaswit.mak sybswit.dll
```

O arquivo de comutador gerado é colocado em `c:\Program Files\IBM\WebSphere MQ\exits`..

É possível localizar xaswit.mak no diretório *MQ_INSTALLATION_PATH*`/samp/xatm`. *MQ_INSTALLATION_PATH* Representa o diretório de alto nível no qual o WebSphere MQ está instalado

Edite xaswit.mak para *remover o comentário* das linhas apropriadas na versão do Sybase que você estiver usando. Em seguida, execute o makefile usando o comando:

```
make -f xaswit.mak sybswit
```

O arquivo de carregamento do comutador gerado de 32 bits será colocado em `/var/mqm/exits`.

O arquivo de carregamento do comutador gerado de 64 bits será colocado em `/var/mqm/exits64`.

Incluindo Informações de Configuração do Gerenciador de Recursos para o Sybase

Você deve modificar as informações de configuração para o gerenciador de filas para declarar o Sybase como um participante em unidades globais de trabalho. A modificação das informações de configuração será descrito com mais detalhes em "Incluindo informações de configuração no gerenciador de filas" na página 50.

- Nos sistemas Windows e Linux (plataformasx86 e x86-64), use o WebSphere MQ Explorer. Especifique os detalhes do arquivo de carregamento do comutador no painel de propriedades do gerenciador de filas, sob o gerenciador de recursos XA.
- Em todos os outros sistemas, especifique os detalhes do arquivo de carregamento do comutador na sub-rotina XAResourceManager no arquivo qm.ini do gerenciador de filas.

Figura 13 na página 62 mostra uma amostra do UNIX and Linux , que usa o banco de dados associado à definição de LRM *lrmname* no arquivo de configuração XA Sybase , $SYBASE/$SYBASE_OCS/`xa_config`. Inclua um nome do arquivo de log se você desejar que as chamadas de função XA sejam registrados:

```
XAResourceManager:
  Name=mysybase
  SwitchFile=sybswit
  XAOpenString=-Uuser -Ppassword -Nlrmname -L/tmp/sybase.log -Txa
  ThreadOfControl=THREAD
```

Figura 13. Entrada XAResourceManager de Amostra para o Sybase em Plataformas do UNIX and Linux

Usando Programas Multiencadeados com o Sybase

Se você estiver usando programas multiencadeados com unidades globais de trabalho do WebSphere MQ incorporando atualizações para Sybase, ***deverá*** usar o valor THREAD para o parâmetro de controle ThreadOf. Além disso, assegure-se de que seu programa seja vinculado (e o arquivo de carregamento do comutador) com as bibliotecas do Sybase de thread-safe (as versões _r). O uso do valor THREAD para o parâmetro ThreadOfControl será mostrado em Figura 13 na página 62.

Diversas configurações do banco de dados

Se você deseja configurar o gerenciador de filas para que as atualizações nos diversos bancos de dados possam ser incluídas nas unidades globais de trabalho, inclua uma sub-rotina XAResourceManager em cada banco de dados.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Se os bancos de dados são todos gerenciados pelo mesmo gerenciador de banco de dados, cada sub-rotina definirá um banco de dados separado. Cada sub-rotina especifica o mesmo *SwitchFile*, mas o conteúdo do *XAOpenString* será diferente porque ele especifica o nome do banco de dados que está sendo atualizado. Por exemplo, as sub-rotinas mostradas em Figura 14 na página 62 configuram o gerenciador de filas com os bancos de dados Db2 *MQBankDB* e *MQFeeDB* nos sistemas UNIX and Linux .

Importante: Não é possível ter múltiplas sub-rotinas apontando para o mesmo banco de dados. Essa configuração não funciona em nenhuma circunstância e, se for tentada, ela falhará.

Você receberá erros do formulário `when the MQ code makes its second xa_open call in any process in this environment, the database software fails the second xa_open with a -5 error, XAER_INVAL`.

```
XAResourceManager:
  Name=DB2 MQBankDB
  SwitchFile=db2swit
  XAOpenString=MQBankDB

XAResourceManager:
  Name=DB2 MQFeeDB
  SwitchFile=db2swit
  XAOpenString=MQFeeDB
```

Figura 14. Entradas XAResourceManager de amostra para diversos bancos de dados Db2

Se os bancos de dados a serem atualizados forem gerenciados por gerenciadores de banco de dados diferentes, inclua uma sub-rotina XAResourceManager para cada um. Nesse caso, cada sub-rotina especificará um *SwitchFile* diferente. Por exemplo, se o *MQFeeDB* for gerenciado pelo Oracle em vez de DB2, use as seguintes sub-rotinas nos sistemas UNIX and Linux :

```
XAResourceManager:
  Name=DB2 MQBankDB
  SwitchFile=db2swit
  XAOpenString=MQBankDB

XAResourceManager:
  Name=Oracle MQFeeDB
  SwitchFile=oraswit
  XAOpenString=Oracle_XA+Acc=P/myuser/mypassword+SesTm=35+LogDir=/tmp/ora.log+DB=MQFeeDB
```

Figura 15. Entradas XAResourceManager de amostra para um banco de dados DB2 e Oracle

A princípio, não há nenhum limite para o número de instâncias do banco de dados que poderá ser configurado com um único gerenciador de filas.

Nota: Para obter informações sobre o suporte para incluir bancos de dados Informix em diversas atualizações do banco de dados em unidades globais de trabalho, verifique o arquivo leia-me do produto

Considerações sobre Segurança

Considerações para executar seu banco de dados sob o modelo XA.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

As informações a seguir são fornecidas apenas para orientação. Em todos os casos, consulte a documentação fornecida com o gerenciador do banco de dados para determinar as implicações de segurança da execução do seu banco de dados sob o modelo XA.

Um processo de aplicativo denota o início de uma unidade global de trabalho usando o verbo `MQBEGIN`. A primeira chamada `MQBEGIN` que um aplicativo emite, conecta-se a todos os bancos de dados participantes, chamando o código da biblioteca do cliente no ponto de entrada xa_open. Todos os gerenciadores de banco de dados oferecem um mecanismo para o fornecimento de um ID e uma senha do usuário em seu XAOpenString. Esse é o único momento em que as informações de autenticação fluem.

Observe que, em plataformas UNIX and Linux, os aplicativos de atalho deverão ser executados com um ID de usuário efetivo de mqm ao fazer as chamadas de MQI.

Considerações quando o contato estiver perdido com o gerenciador de recursos XA

O gerenciador de filas tolera os gerenciadores de banco de dados que não estão disponíveis. Isso significa que você pode iniciar e parar o gerenciador de filas independentemente do servidor de banco de dados. Quando o contato é restaurado, o gerenciador de filas e gerenciador de banco de dados são ressincronizados. Você também pode utilizar o comando rsvmqtrn para resolver manualmente as unidades de trabalho em dúvida.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Em operações normais, apenas uma quantia mínima de administração será necessária após ter concluído as etapas de configuração. A tarefa de administração é facilitada porque o gerenciador de filas tolera os gerenciadores de banco de dados que não estão disponíveis. Em particular, isso significa que:

- O gerenciador de filas pode iniciar se a qualquer momento sem primeiro iniciar cada um dos gerenciadores do banco de dados.
- O gerenciador de filas não precisará parar e reiniciar, se um dos gerenciadores de banco de dados se tornar indisponível.

Isso permite que você inicie e pare o gerenciador de filas independentemente do servidor de banco de dados.

Sempre que o contato estiver perdido entre o gerenciador de filas e um banco de dados, eles precisarão ressincronizar quando se tornarem disponíveis novamente. A ressincronização é o processo pelo qual quaisquer unidades de trabalho indeterminado que envolvem esse banco de dados serão concluídas. Em geral, isso ocorre automaticamente sem a necessidade de intervenção do usuário. O gerenciador de filas solicitará o banco de dados para uma lista de unidades de trabalho que esteja pendente. Ele então instrui o banco de dados a confirmar ou retroceder cada uma dessas unidades de trabalho indeterminado.

Quando um gerenciador de filas for iniciado, ele ressincronizará com cada banco de dados. Quando um banco de dados individual se torna indisponível, apenas esse banco de dados precisa ser ressincronizado na próxima vez em que o gerenciador de filas observar que ele está disponível novamente.

O gerenciador de filas obtém novamente o contato com um banco de dados anteriormente indisponível automaticamente quando as novas unidades globais de trabalho forem iniciadas com MQBEGIN. Ele faz isso chamando a função xa_open na biblioteca do cliente de banco de dados. Se essa chamada xa_open falhar, MQBEGIN retornará com um código de conclusão de MQCC_WARNING e um código de razão de MQRC_PARTICIPANT_NOT_AVAILABLE. Você pode tentar novamente a chamada MQBEGIN posteriormente.

Não continue tentando uma unidade global de trabalho que envolva as atualizações a um banco de dados que tenha indicado uma falha durante MQBEGIN. Não haverá uma conexão com esse banco de dados através do qual as atualizações poderão ser feitas. As únicas opções são finalizar o programa ou tentar novamente oMQBEGIN periodicamente na esperança de que o banco de dados possa se tornar disponível novamente.

Como alternativa, é possível usar o comando `rsvmqtrn` para resolver explicitamente todas as unidades de trabalho indeterminado.

Unidades de Trabalho Indeterminadas

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Um banco de dados pode ser deixado com unidades de trabalho indeterminadas se o contato com o gerenciador de filas for perdido depois que o gerenciador de banco de dados tiver sido instruído a se preparar Até que o servidor de banco de dados receba o resultado do gerenciador de filas (confirmação ou retrocesso), ele precisará reter os bloqueios do banco de dados associados com as atualizações.

Como esses bloqueios evitam que outros aplicativos de atualizar ou ler registros do banco de dados, a ressincronização precisará acontecer o mais breve possível.

Se, por alguma razão, você não puder esperar que o gerenciador de filas ressincronize com o banco de dados automaticamente, será possível usar as instalações fornecidos pelo gerenciador de banco de dados para confirmar ou retroceder as atualizações do banco de dados manualmente. No *X/Open Distributed Transaction Processing: The XA Specification*, isso é chamado de tomar uma decisão *heurística*. Use-o somente como um último recurso, devido à possibilidade de comprometer a integridade de dados; você poderá, por exemplo, erradamente retroceder as atualizações do banco de dados quando todos os outros participantes tiverem confirmado suas atualizações.

É muito melhor reiniciar o gerenciador de filas ou usar o comando `rsvmqtrn` quando o banco de dados tiver sido reiniciado, inicie a ressincronização automática.

Exibindo Unidades de Trabalho Pendentes com o Comando dspmqtrn

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Enquanto um gerenciador do banco de dados estiver indisponível, é possível usar o comando **dspmqtrn** para verificar o estado das unidades globais de trabalho pendente que envolvam esse banco de dados.

O comando **dspmqtrn** exibe somente essas unidades de trabalho nas quais um ou mais participantes estão indeterminados. Os participantes estão esperando a decisão do gerenciador de filas para confirmar ou retroceder as atualizações preparadas.

Para cada uma dessas unidades globais de trabalho, o estado de cada participante será exibido na saída do **dspmqtrn**. Se a unidade de trabalho não atualizou os recursos de um gerenciador de recursos específico, ele não será exibido.

Com relação a uma unidade de trabalho indeterminado, um gerenciador de recursos será requisitado a fazer uma das coisas a seguir:

Preparado
O gerenciador de recursos é preparado para confirmar suas atualizações.

Confirmado
O gerenciador de recursos confirmou suas atualizações.

Retrocedido
O gerenciador de recursos retrocedeu suas atualizações.

Participado
O gerenciador de recursos é um participante, mas não preparou, confirmou ou retrocedeu suas atualizações.

Quando o gerenciador de filas for reiniciado, ele solicitará que cada banco possua uma sub-rotina XAResourceManager para uma lista de unidades globais de trabalho indeterminado. Se o banco de dados não tiver sido reiniciado ou estiver de outra maneira indisponível, o gerenciador de filas ainda não poderá entregar ao banco de dados o resultado final para essas unidades de trabalho. O resultado das unidades de trabalho indeterminado é entregue para o banco de dados na primeira oportunidade quando o banco de dados estiver disponível novamente.

Nesse caso, o gerenciador de banco de dados é relatado como estando no estado *preparado* até que a ressincronização tenha ocorrido.

Sempre que o comando `dspmqtrn` exibir uma unidade de trabalho indeterminado, ele primeiro listará todos os gerenciadores de recursos possíveis que possam ser participantes. Esses são alocados a um identificador exclusivo, *RMId*, que será usado em vez do *Nome* dos gerenciadores de recursos ao relatar seu estado com relação a uma unidade de trabalho indeterminado.

Saída dspmqtrn de Amostra mostra o resultado emitindo o comando a seguir:

```
dspmqtrn -m MY_QMGR
```

```
AMQ7107: Resource manager 0 is MQSeries.
AMQ7107: Resource manager 1 is DB2 MQBankDB.
AMQ7107: Resource manager 2 is DB2 MQFeeDB.

AMQ7056: Transaction number 0,1.
    XID: formatID 5067085, gtrid_length 12, bqual_length 4
         gtrid [3291A50600002013746573 74]
         bqual [00000001]
AMQ7105: Resource manager 0 has committed.
AMQ7104: Resource manager 1 has prepared.
AMQ7104: Resource manager 2 has prepared.
```

em que *Transaction number* é o ID da transação que poderá ser usado com o comando rsvmqtrn. Consulte AMQ7000-7999: WebSphere MQ product para obter informações adicionais sobre a mensagem AMQ7056 . As variáveis *XID* fazem parte da *Especificação X/Open XA*; para obter informações mais atualizadas sobre essa especificação, consulte: https://publications.opengroup.org/c193.

Figura 16. Saída dspmqtrn de Amostra

A saída em Saída dspmqtrn de amostra mostra que existem três gerenciadores de recursos associados ao gerenciador de filas. O primeiro é o gerenciador de recursos `0`, que é o próprio gerenciador de filas. As outras duas instâncias do gerenciador de recursos são os bancos de dados `MQBankDB` e `MQFeeDB` Db2 .

O exemplo mostra apenas uma única unidade de trabalho indeterminado. Uma mensagem é emitida para todos os três gerenciadores de recurso, o que significa que as atualizações foram feitas no gerenciador de filas e nos bancos de dados Db2 dentro da unidade de trabalho.

As atualizações feitas no gerenciador de filas, o gerenciador de recursos **0**, foi *confirmado*. As atualizações nos bancos de dados Db2 estão no estado *prepared* , o que significa que o Db2 deve ter se tornado indisponível antes de ser chamado para confirmar as atualizações para os bancos de dados *MQBankDB* e *MQFeeDB* .

A unidade de trabalho indeterminado tem um identificador externo chamado XID (*ID de transação*). Esta é uma parte dos dados fornecidos ao Db2 pelo gerenciador de filas para identificar sua parte da unidade global de trabalho.

Resolvendo Unidades de Trabalho Pendentes com o Comando rsvmqtrn
Unidades de trabalho pendentes são concluídas quando o gerenciador de fila e o DB2 são ressincronizados.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

A saída mostrada em Figura 16 na página 65 mostra uma única unidade de trabalho em dúvida na qual a decisão de confirmação ainda não foi entregue para ambos os bancos de dados DB2 ...

Para concluir essa unidade de trabalho, o gerenciador de filas e o DB2 precisam ressincronizar quando o DB2 se tornar disponível em seguida. O gerenciador de filas usa o início de novas unidades de trabalho para recuperar o contato com o DB2. Como alternativa, você pode instruir o gerenciador de filas para ressincronizar explicitamente usando o comando **rsvmqtrn**.

Faça isso logo após o DB2 ter sido reiniciado, para que quaisquer bloqueios do banco de dados associados à unidade de trabalho em dúvida sejam liberados o mais rápido possível. Use a opção -a, que informa ao gerenciador de filas para resolver todas as unidades de trabalho indeterminado. No exemplo a seguir, o DB2 foi reiniciado, portanto, o gerenciador de filas pode resolver a unidade de trabalho indeterminada:

```
> rsvmqtrn -m MY_QMGR -a
Any in-doubt transactions have been resolved.
```

Erros e Resultados Combinados
Embora o gerenciador de filas use um protocolo two-phase commit, isso não removerá completamente a possibilidade de algumas unidades de trabalho serem concluídas com resultados mistos. Esse é o local onde alguns participantes confirmam as suas atualizações, e outros, as retornam.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

As unidades de trabalho que concluírem com um resultado misto terá implicações graves porque os recursos compartilhados que deveriam ter sido atualizados como uma única unidade de trabalho não estarão em um estado consistente.

Os resultados mistos são causados principalmente quando as decisões heurísticas são feitas sobre as unidades de trabalho em vez de permitir que o gerenciador de filas resolva as unidades de trabalho indeterminado por si só. Tais decisões estão fora do controle do gerenciador de filas.

Sempre que o gerenciador de filas detectar um resultado combinado, ele produzirá informações de FFST e documentará a falha em seus logs de erros, com uma de duas mensagens:

- Se um gerenciador do banco de dados retrocede em vez de confirmar:

```
AMQ7606 A transaction has been committed but one or more resource
        managers have rolled back.
```

- Se um gerenciador de banco de dados confirma em vez de retroceder:

```
AMQ7607 A transaction has been rolled back but one or more resource
        managers have committed.
```

Mensagens adicionais identificam os bancos de dados que são danificados heuristicamente. Então, é sua responsabilidade restaurar a consistência localmente nos bancos de dados afetados. Esse é um procedimento complexo no qual você precisa primeiro isolar a atualização que foi indevidamente confirmada ou retrocedida, em seguida, desfazer ou refazer a mudança do banco de dados manualmente.

Alterando Informações de Configuração
Após o gerenciador de filas ter sido iniciado com êxito para coordenar as unidades globais de trabalho, não altera nenhuma das informações de configuração do gerenciador de recursos.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Se você precisar alterar as informações de configuração, será possível fazer isso a qualquer momento, mas as alterações não terão efeito até que o gerenciador de filas seja reiniciado.

Se você remover as informações de configuração do gerenciador de recursos de um banco de dados, você estará removendo efetivamente a capacidade do gerenciador de filas de entrar em contato com esse gerenciador de banco de dados.

Nunca altere o atributo *Nome* em qualquer uma das informações de configuração do gerenciador de recursos. Esse atributo identifica exclusivamente essa instância do gerenciador de banco de dados para o gerenciador de filas. Se você alterar esse identificador exclusivo, o gerenciador de filas assumirá que o banco de dados foi removido e uma instância totalmente nova foi incluída. O gerenciador de filas ainda associa as unidades de trabalho pendente com o antigo *Nome*, possivelmente deixando o banco de dados em um estado indeterminado.

Removendo Instâncias do Gerenciador de Banco de Dados
Se você precisar remover um banco de dados de sua configuração permanentemente, assegure-se de que o banco de dados não esteja indeterminado antes de reiniciar o gerenciador de filas.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Os produtos de banco de dados fornecem comandos para listar as transações indeterminadas. Se houver quaisquer transações indeterminadas, primeiro permita que o gerenciador de filas ressincronize com o banco de dados. Faça isso iniciando o gerenciador de filas. É possível verificar se a ressincronização ocorreu, usando o comando **rsvmqtrn** do próprio comando do banco de dados para visualizar as unidades de trabalho indeterminado. Quando estiver satisfeito que a ressincronização ocorreu, encerre o gerenciador de filas e remova as informações de configuração do banco de dados.

Se você não observar esse procedimento, o gerenciador de filas ainda lembrará todas as unidades de trabalho indeterminado envolvendo esse banco de dados. Uma mensagem de aviso, AMQ7623, é emitida a cada vez que o gerenciador de filas for reiniciado. Se você nunca for configurar esse banco de dados com o gerenciador de filas novamente, use a opção -r do comando **rsvmqtrn** para instruir o gerenciador de filas a esquecer da participação do banco de dados em suas transações indeterminadas. O gerenciador de filas esquece tais transações somente quando as transações indeterminadas forem concluídas com todos os participantes.

Há momentos em que talvez você precise remover algumas informações de configuração do gerenciador de recursos temporariamente. Em sistemas UNIX and Linux, isso será mais alcançado comentando a linha na sub-rotina para que possa ser facilmente restabelecida posteriormente. Você pode optar por fazer isso, se houver erros toda vez que o gerenciador de filas entrar em contato com um determinado banco de dados ou gerenciador de banco de dados. Removendo temporariamente as informações de configuração do gerenciador de recursos em questão permite que o gerenciador de filas inicie as

unidades globais de trabalho envolvendo todos os outros participantes. Aqui está um exemplo de uma sub-rotina XAResourceManager de linha comentada a seguir:

```
# This database has been temporarily removed
#XAResourceManager:
#  Name=mydb2
#  SwitchFile=db2swit
#  XAOpenString=mydbname,myuser,mypassword,toc=t
#  ThreadOfControl=THREAD
```

Figura 17. Sub-rotina XAResourceManager de Linha Comentada em Sistemas UNIX and Linux

Em sistemas Windows , use o WebSphere MQ Explorer para excluir as informações sobre a instância do gerenciador de banco de dados Tome muito cuidado ao inserir o nome correto no campo *Nome* ao restabelecê-lo. Se você inseriu o nome errado, você poderá enfrentar problemas indeterminados, conforme descrito em “Alterando Informações de Configuração” na página 67.

Registro dinâmico de XA

A especificação XA fornece uma maneira de reduzir o número de chamadas xa_* que um gerenciador de transações fará a um gerenciador de recursos. Essa otimização é conhecida como *registro dinâmico*.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

O registro dinâmico é suportado pelo DB2. Outros bancos de dados podem suportá-lo ; consulte a documentação de seu produto de banco de dados para obter detalhes.

Por que a otimização de registro dinâmico será útil? Em seu aplicativo, algumas unidades globais de trabalho podem conter atualizações para as tabelas de banco de dados; outras podem não conter essas atualizações. Quando nenhuma atualização persistente for feita para as tabelas de um banco de dados, não haverá nenhuma necessidade de incluir esse banco de dados no protocolo de confirmação que ocorre durante o MQCMIT.

Independentemente de seu banco de dados suportar ou não o registro dinâmico, seu aplicativo chama xa_open durante a primeira chamada MQBEGIN em uma conexão WebSphere MQ . Também chamará xa_close na chamada MQDISC subsequente. O padrão de chamadas XA subsequentes depende se o banco de dados suportará o registro dinâmico:

Se o banco de dados não suportar o registro dinâmico ...
Cada unidade de trabalho global envolve várias chamadas de funções XA feitas pelo código do WebSphere MQ na biblioteca do cliente de banco de dados, independentemente se você fez uma atualização persistente para as tabelas desse banco de dados dentro de sua unidade de trabalho. Isso inclui:

- xa_start e xa_end do processo do aplicativo. Esses são usados para declarar o início e o fim de uma unidade global de trabalho.
- xa_prepare, xa_commit e xa_rollback do processo do agente do gerenciador de filas, amqzlaa0. Esses são usados para entregar o resultado da unidade global de trabalho: a decisão de confirmar ou retroceder.

Além disso, o processo do agente do gerenciador de filas também chamará o xa_open durante a primeira MQBEGIN.

Se seu banco de dados suportar o registro dinâmico ...
O código do WebSphere MQ faz apenas as chamadas de função XA necessárias. Para uma unidade global de trabalho que **não** envolveu as atualizações persistentes aos recursos do banco de dados, **não** haverá as chamadas XA ao banco de dados. Para uma unidade global de trabalho que **tenha** envolvido tais atualizações persistentes, as chamadas serão para:

- xa_end do processo de aplicativo para declarar o fim da unidade global de trabalho.

- `xa_prepare`, `xa_commit` e `xa_rollback` do processo do agente do gerenciador de filas, amqzlaa0. Esses são usados para entregar o resultado da unidade global de trabalho: a decisão de confirmar ou retroceder.

Para que o registro dinâmico funcione, é vital que o banco de dados tenha uma maneira de informar ao WebSphere MQ quando ele tiver executado uma atualização persistente que deseja incluir na unidade global de trabalho atual. WebSphere MQ fornece a função `ax_reg` para este propósito.

O código do cliente do banco de dados que é executado em seu processo de aplicativo localiza a função `ax_reg` e a chama para *registrar dinamicamente* pelo fato de ter feito o trabalho persistente dentro da unidade global de trabalho atual. Em resposta a essa chamada `ax_reg` , o WebSphere MQ registra que o banco de dados participou. Se esta for a primeira chamada `ax_reg` nessa conexão do WebSphere MQ , o processo do agente do gerenciador de filas chamará `xa_open`.

O código do cliente de banco de dados fará essa chamada `ax_reg` quando estiver em execução no processo, por exemplo, durante uma chamada SQL UPDATE ou qualquer chamada na API do cliente do banco de dados ao qual for responsável

Condições de erro
No registro dinâmico XA haverá a possibilidade de uma falha confusa no gerenciador de filas.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Um exemplo comum é se você esquecer-se de configurar as variáveis de ambiente do banco de dados corretamente antes de iniciar o gerenciador de filas, as chamadas do gerenciador de fila para o `xa_open` falhará. Não haverá nenhuma unidade global de trabalho que possa ser usada.

Para evitar isso, assegure-se que você tenha configurado as variáveis de ambiente relevantes antes de iniciar o gerenciador de filas. Revise a documentação do produto do banco de dados e o conselho fornecido em “Configurando Db2” na página 52, “Configurando o Oracle” na página 55 e “Configuração do Sybase” na página 60.

Com todos os produtos de banco de dados, o gerenciador de filas chama o `xa_open` uma vez na inicialização do gerenciador de filas, como parte da sessão de recuperação (conforme explicado em “Considerações quando o contato estiver perdido com o gerenciador de recursos XA” na página 63). Essa chamada `xa_open` falha se suas variáveis de ambiente de banco de dados forem configuradas incorretamente, mas não fará com que o gerenciador de filas falhe ao iniciar. Isso ocorre porque o mesmo código de erro `xa_open` é usado pela biblioteca do cliente do banco de dados para indicar que o servidor de banco de dados estará indisponível. O WebSphere MQ não trata isso como um erro grave, pois o gerenciador de fila deve ser capaz de iniciar o processamento de dados fora das unidades globais de trabalho que envolvem esse banco de dados

Chamadas subsequentes para `xa_open` são feitas a partir do gerenciador de filas durante o primeiro MQBEGIN em uma conexão do WebSphere MQ (se o registro dinâmico não estiver sendo usado) ou durante uma chamada pelo código do cliente de banco de dados para a função WebSphere MQfornecida `ax_reg` (se o registro dinâmico estiver sendo usado).

A **sincronização** de quaisquer condições de erro (ou, ocasionalmente, relatórios FFST) depende se você está usando o registro dinâmico:

- Se você estiver usando o registro dinâmico, sua chamada MQBEGIN poderia ser bem-sucedida, mas a chamada do banco de dados do SQL UPDATE (ou semelhante) falhará.
- Se você não estiver usando o registro dinâmico, a chamada MQBEGIN falhará.

Assegure-se de que as variáveis de ambiente estejam configuradas corretamente no aplicativo e nos processos do gerenciador de filas.

Resumindo chamadas XA
Aqui está uma lista das chamadas que são feitas para as funções XA em uma biblioteca do cliente de banco de dados como resultado das várias chamadas de MQI que controlam unidades globais de

trabalho. Essa não é uma descrição completa do protocolo descrito na especificação XA; é fornecido como uma visão geral resumida.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Observe que as chamadas `xa_start` e `xa_end` são sempre chamadas pelo código WebSphere MQ no processo de aplicativo, enquanto `xa_prepare`, `xa_commite` `xa_rollback` são sempre chamadas a partir do processo do agente do gerenciador de filas, amqzlaa0.

As chamadas `xa_open` e `xa_close` mostradas nessa tabela são todas feitas a partir do processo do aplicativo. O processo do agente do gerenciador de filas chamará o `xa_open` nas circunstâncias descritas em “Condições de erro” na página 69.

Tabela 7. Resumo de Chamadas de Função XA

Chamada MQI	**Chamadas XA feitas com o registro dinâmico**	**Chamadas XA feitas sem o registro dinâmico**
Primeiro MQBEGIN	`xa_open`	`xa_open` `xa_start`
MQBEGIN subsequente	Nenhuma chamada XA	`xa_start`

MQCMIT (**sem** `ax_reg` ser chamado durante a unidade global de trabalho atual)	Nenhuma chamada XA	`xa_end`
		`xa_prepare`
		`xa_commit`
		`xa_rollback`

MQCMIT (**com** `ax_reg` sendo chamado durante a unidade global de trabalho atual)	`xa_end`
	`xa_prepare`
	`xa_commit`
	`xa_rollback`
		Não aplicável. Nenhuma chamada será feita para ax_reg em modo não dinâmico.

MQBACK (**sem** `ax_reg` ser chamado durante a unidade global de trabalho atual)	Nenhuma chamada XA	`xa_end` `xa_rollback`
MQBACK (**com** `ax_reg` sendo chamado durante a unidade de trabalho global atual)	`xa_end` `xa_rollback`	Não aplicável. Nenhuma chamada será feita para ax_reg em modo não dinâmico.
MQDISC, em que MQCMIT ou MQBACK foi chamado primeiro. Se eles não foram, o processamento MQCMIT é feito primeiro durante o MQDISC.	`xa_close`	`xa_close`

Notes:

1. Para MQCMIT, `xa_commit` é chamado se o `xa_prepare` for bem-sucedido. Caso contrário, `xa_rollback` será chamado.

Cenário 2: Outro Software Fornece a Coordenação

No cenário 2, um gerenciador de transações externo coordena as unidades globais de trabalho, os iniciando e confirmando sob o controle da API do gerenciador de transações. Os verbos MQBEGIN, MQCMIT e MQBACK estão indisponíveis.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Essa seção descreve esse cenário, incluindo:

- "Coordenação do ponto de sincronização externa" na página 71
- "Usando CICS" na página 73
- "Usando o Microsoft Transaction Server (COM +)" na página 78

O cliente do IBM WebSphere MQ para o HP Integrity NonStop Server pode usar o HP NonStop Transaction Management Facility (TMF) para coordenar as unidades globais de trabalho. Para obter mais informações, consulte Utilizando HP NonStop TMF.

Coordenação do ponto de sincronização externa

Uma unidade global de trabalho também pode ser coordenada por um gerenciador de transações compatíveis com o X/Open XA externo. Aqui, o gerenciador de filas do WebSphere MQ participa, mas não coordena, a unidade de trabalho

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

O fluxo de controle em uma unidade global de trabalho coordenada por um gerenciador de transações externas é o seguinte:

1. Um aplicativo informa ao coordenador do ponto de sincronização externo (por exemplo, TXSeries) que deseja iniciar uma transação.
2. O coordenador do ponto de sincronização informa aos gerenciadores de recursos conhecidos, como WebSphere MQ, sobre a transação atual.
3. O aplicativo emite as chamadas para os gerenciadores de recursos associados à transação atual. Por exemplo, o aplicativo poderia emitir MQGET chamadas para WebSphere MQ.
4. O aplicativo emite uma solicitação de confirmação ou recuperação para o coordenador do ponto de sincronização externa.
5. O coordenador do ponto de sincronização conclui a transação, emitindo as chamadas apropriadas para cada gerenciador de recursos, geralmente usando os protocolos two-phase commit.

Os níveis suportados de coordenadores de ponto de sincronização externos que podem fornecer um processo two-phase commit para transações nas quais o WebSphere MQ participa são definidos em IBM WebSphere MQ requisitos detalhados do sistema.

O restante dessa seção descreve como ativar as unidades de trabalho externo.

A estrutura do comutador XA IBM WebSphere MQ

Cada gerenciador de recursos participando de uma unidade de trabalho coordenado externamente deve fornecer uma estrutura do comutador XA. Essa estrutura define os recursos do gerenciador de recursos e as funções que devem ser chamadas pelo coordenador do ponto de sincronização.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

O IBM WebSphere MQ fornece duas versões dessa estrutura:

- *MQRMIXASwitch* para o gerenciamento de recurso XA estático
- *MQRMIXASwitchDynamic* para o gerenciamento de recurso XA dinâmico

Consulte a documentação do gerenciador de transação para determinar se deve usar a interface de gerenciamento de recurso estático ou dinâmico. Sempre que um gerenciador de transações o suportar, recomendamos que use o gerenciamento de recurso XA dinâmico.

Alguns gerenciadores de transação de 64 bits tratam o tipo *longo* na especificação XA como 64 bits e alguns o tratam como 32 bits. WebSphere MQ suporta ambos os modelos:

- Se o gerenciador de transações for de 32 bits ou 64 bits, mas trata o tipo *longo* como 32 bits, use o arquivo de carregamento do comutador listado no Tabela 8 na página 72.
- Se o gerenciador de transações for de 64 bits e trata o tipo *longo* como 64 bits, use o arquivo de carregamento do comutador listado no Tabela 9 na página 72.

Uma lista de gerenciadores de transações de 64 bits conhecidos que tratam o tipo *longo* como 64 bits será fornecido em Tabela 10 na página 73. Consulte a documentação do gerenciador de transações se não estiver seguro sobre qual modelo seu gerenciador de transações usa.

Tabela 8. Nomes de Arquivo de Carregamento do Comutador XA

Plataforma	**Nome do arquivo de carregamento de comutador (servidor)**	**Nome do arquivo de carregamento de comutador (cliente transacional estendido)**
Windows	*mqmxa.dll*	*mqcxa.dll*
AIX (não encadeado).	*libmqmxa.a*	*libmqcxa.a*
AIX (encadeado)	*libmqmxa_r.a*	*libmqcxa_r.a*
HP-UX (não encadeado)	*libmqmxa.so*	*libmqcxa.so*
HP-UX (encadeado)	*libmqmxa_r.so*	*libmqcxa_r.so*
Linux (não encadeado)	*libmqmxa.so*	*libmqcxa.so*
Linux (encadeado)	*libmqmxa_r.so*	*libmqcxa_r.so*
Solaris	*libmqmxa.so*	*libmqcxa.so*

Tabela 9. Nomes de Arquivo de Carregamento do Comutador XA de 64 Bits Alternativo

Plataforma	**Nome do arquivo de carregamento de comutador (servidor)**	**Nome do arquivo de carregamento de comutador (cliente transacional estendido)**
AIX (não encadeado).	*libmqmxa64.a*	*libmqcxa64.a*
AIX (encadeado)	*libmqmxa64_r.a*	*libmqcxa64_r.a*
HP-UX (não encadeado)	*libmqmxa64.so*	*libmqcxa64.so*
HP-UX (encadeado)	*libmqmxa64_r.so*	*libmqcxa64_r.so*
Linux (não encadeado)	*libmqmxa64.so*	*libmqcxa64.so*
Linux (encadeado)	*libmqmxa64_r.so*	*libmqcxa64_r.so*
Solaris	*libmqmxa64.so*	*libmqcxa64.so*

Tabela 10. Gerenciadores de Transações de 64 Bits que Requerem o Arquivo de Carregamento do Comutador de 64 Bits Alternativo
Gerenciador de Transações
Tuxedo

Alguns coordenadores de pontos de sincronização externos (não CICS) requerem que cada gerenciador de recursos que participa de uma unidade de trabalho forneça seu nome no campo de nome da estrutura do comutador XA O nome do gerenciador de recursos do WebSphere MQ é MQSeries_XA_RMI

O coordenador do ponto de sincronização define como a estrutura do comutador XA WebSphere MQ se vincula a ele. Informações sobre como vincular a estrutura do comutador XA do WebSphere MQ com CICS são fornecidas em "Usando CICS" na página 73. Para obter informações sobre como vincular a estrutura do comutador XA do WebSphere MQ com outros coordenadores de ponto de sincronização compatíveis com XA, consulte a documentação fornecida com esses produtos

As considerações a seguir se aplicam ao uso do WebSphere MQ com todos os coordenadores de ponto de sincronização compatíveis com XA:

- A estrutura xa_info transmitida em qualquer chamada xa_open pelo coordenador do ponto de sincronização inclui o nome de um gerenciador de fila do WebSphere MQ O nome possui o mesmo formato que o nome do gerenciador de filas transmitido para a chamada `MQCONN`. Se o nome transmitido na chamada xa_open estiver em branco, o gerenciador de filas padrão será usado.

  Como alternativa, a estrutura xa_info pode conter valores para os parâmetros *TPM* e *AXLIB*. O parâmetro *TPM* especifica o gerenciador de transações que estiver sendo usado. Os valores válidos são CICS, TUXEDO e ENCINA. O parâmetro *AXLIB* especifica o nome da biblioteca que contém as funções ax_reg e ax_unreg do gerenciador de transações. Para obter mais informações sobre esses parâmetros, consulte Configurando um Cliente Transacional Estendido. Se a estrutura xa_info contiver qualquer um desses parâmetros, o nome do gerenciador de filas será especificado no parâmetro *QMNAME*, a menos que o gerenciador de filas padrão esteja sendo usado.
- Apenas um gerenciador de filas por vez poderá participar em uma transação coordenada por uma instância de um coordenador do ponto de sincronização externo. O coordenador do ponto de sincronização é efetivamente conectado ao gerenciador de filas e está sujeito à regra de que apenas uma conexão por vez seja suportada.
- Todos os aplicativos que incluem as chamadas para um coordenador do ponto de sincronização externa podem se conectar apenas ao gerenciador de filas que está participando na transação gerenciada pelo coordenador externo (porque eles já estão efetivamente conectados a esse gerenciador de filas). No entanto, tais aplicativos devem emitir uma chamada `MQCONN` para obter uma manipulação de conexões e uma chamada `MQDISC` antes de sair.
- Um gerenciador de filas com as atualizações de recursos coordenados por um coordenador do ponto de sincronização externa deve ser iniciado antes do coordenador do ponto de sincronização externa. Da mesma forma, o coordenador do ponto de sincronização deve terminar antes do gerenciador de filas.
- Se o coordenador do ponto de sincronização externa for finalizado de forma anormal, pare e reinicie o gerenciador de filas ***antes*** de reiniciar o coordenador do ponto de sincronização para assegurar-se de que quaisquer operações de mensagens não confirmadas no momento da falha sejam resolvidas corretamente.

Usando CICS

CICS é um dos elementos de TXSeries..

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

As versões do TXSeries que são compatíveis com XA (e usam um processo two-phase commit) são definidas em: IBM WebSphere MQ requisitos detalhados do sistema

WebSphere MQ também suporta outros gerenciadores de transações. Consulte IBM WebSphere MQ requisitos detalhados do sistema para as listas atuais de software suportado.

Requisitos do processo two-phase commit
Requisitos do processo two-phase commit ao usar o processo two-phase commit CICS com WebSphere MQ. Esses requisitos não se aplicam ao z/OS

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Observe os seguintes requisitos:

- WebSphere MQ e CICS devem residir na mesma máquina física.
- WebSphere MQ não suporta CICS em um cliente MQI WebSphere MQ .
- Deve-se iniciar o gerenciador de filas, com seu nome especificado na sub-rotina de definição de recurso XAD, ***antes*** de tentar iniciar o CICS. A falha ao fazer isso evitará que você inicie o CICS se você tiver incluído uma sub-rotina de definição de recurso XAD para WebSphere MQ na região CICS
- Apenas um gerenciador de filas WebSphere MQ pode ser acessado por vez a partir de uma única região CICS .
- Uma transação CICS deve emitir um pedido `MQCONN` antes de poder acessar os recursos do WebSphere MQ . A chamada `MQCONN` deve especificar o nome do gerenciador de fila do WebSphere MQ especificado na entrada XAOpen da sub-rotina de definição de recurso XAD para a região CICS . Se essa entrada estiver em branco, a solicitação `MQCONN` deve especificar o gerenciador de filas padrão.
- Uma transação CICS que acessa os recursos WebSphere MQ deve emitir uma chamada `MQDISC` da transação antes de retornar ao CICS. Falha ao fazer isso pode significar que o servidor de aplicativos CICS ainda está conectado, deixando filas abertas. Além disso, se você não instalar uma saída de finalização de tarefa (consulte “Amostra de Saída de Término da Tarefa” na página 77), o servidor de aplicativos CICS poderá terminar de forma anormal posteriormente, talvez durante uma transação subsequente.
- Deve-se assegurar que o ID do usuário do CICS (cics) seja um membro do grupo mqm, para que o código CICS tenha a autoridade para chamar WebSphere MQ.

  Para transações em execução em um ambiente CICS , o gerenciador de filas adapta seus métodos de autorização e determina o contexto da seguinte forma:

  – O gerenciador de filas consulta o ID do usuário no qual o CICS executa a transação. Esse é o ID de usuário verificado pelo Gerenciador de Autoridade de Objeto e é usado para informações de contexto.
  – No contexto da mensagem, o tipo de aplicativo é MQAT_CICS.
  – O nome do aplicativo no contexto é copiado do nome da transação CICS .

Suporte a XA Geral
O General XA não é suportado no IBM i. Um módulo de carregamento do comutador XA é fornecido para permitir que você vincule CICS com WebSphere MQ em sistemas UNIX and Linux . Além disso, arquivos de código-fonte de amostra são fornecidos para permitir que você desenvolva os comutadores XA para mensagens de outra transação.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Os nomes dos módulos de carregamento do comutador fornecidos são:

Tabela 11. Código essencial para aplicativos CICS : rotina de inicialização XA	
C (origem)	**C (executável) – inclua um dos seguintes para seu XAD.Stanza**
amqzscix.c	amqzsc - TXSeries para AIX, Versão 5.1, amqzsc - TXSeries para HP-UX, Versão 5.1 amqzsc - TXSeries para Sun Solaris, Versão 5.1
amqzscin.c	mqmc4swi - TXSeries para Windows, Versão 5.1

Construindo bibliotecas para uso com TXSeries for Multiplatforms
Use estas informações ao construir bibliotecas para uso com o TXSeries for Multiplatforms

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Arquivos de carregamento do comutador pré-construídos são bibliotecas compartilhadas (chamadas *DLLs* no sistema Windows) que podem ser usadas com programas CICS, que requerem uma transação de confirmação de duas fases usando o protocolo XA. Os nomes dessas bibliotecas pré-construídas estão na tabela Código essencial para aplicativos CICS : rotina de inicialização XA. O código fonte de amostra também é fornecido nos diretórios a seguir:

Tabela 12. Diretórios de instalação nos sistemas operacionais Windows, UNIX and Linux		
Plataforma	**Diretório**	**Arquivo de Origem**
UNIX and Linux	*MQ_INSTALLATION_PATH*/ samp/	amqzscix.c
Windows	*MQ_INSTALLATION_PATH*\Tools \c \ Amostras	amqzscin.c

em que*MQ_INSTALLATION_PATH* é o diretório no qual você instalou o IBM WebSphere MQ

Para construir o arquivo de carregamento do comutador a partir da origem de amostra, siga as instruções apropriadas para seu sistema operacional:

AIX
Emita o seguinte comando:

```
export MQM_HOME=/usr/mqm
echo "amqzscix" > tmp.exp
xlc_r $MQM_HOME/samp/amqzscix.c -I/usr/lpp/cics/include -I$MQM_HOME/inc -e amqzscix -bE:tmp.exp
-bM:SRE -o amqzsc /usr/lpp/cics/lib/regxa_swxa.o -L$MQM_HOME/lib -L/usr/lpp/cics/lib -lcicsrt -lEncina
-lEncServer -lpthreads -lsarpc -lmqmcics_r -lmqmxa_r -lmqzi_r -lmqmcs_r
rm tmp.exp
```

Solaris
Emita o seguinte comando:

```
/opt/SUNWspro/bin/cc -s -l/opt/encina/include amqzscix.c -G -o amqzscix  -e
    CICS_XA_Init -LMQ_INSTALLATION_PATH/lib -L/opt/encina/lib
    -L/opt/dcelocal/lib /opt/cics/lib/reqxa_swxa.o
    -lmqmcics -lmqmxa -lmqzi -lmqmcs -lmqmzse -lcicsrt -lEncina -lEncSfs -ldce
```

HP-UX
Emita o seguinte comando:

```
cc -c -s -I/opt/encina/include MQ_INSTALLATION_PATH/samp/amqzscix.c -Aa +z -o  amqzscix.o ld -b
    -o amqzscix amqzscix.o /opt/cics/lib/regxa_swxa.o +e CICS_XA_Init \
    -LMQ_INSTALLATION_PATH/lib -L/opt/encina/lib -L/opt/cics/lib
    -lmqmxa_r -lmqzi_r -lmqmcs_r -lmqmzse -ldbm -lc -lm
```

Plataformas Linux
Emita o seguinte comando:

```
gcc -m32 -shared -fPIC -o amqzscix amqzscix.c
    \-IMQ_INSTALLATION_PATH/inc -I CICS_INSTALLATION_PATH/include
    \-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
    \-Wl,-rpath=/usr/lib -Wl,-rpath-link,/usr/lib -Wl,--no-undefined
    -Wl,--allow-shlib-undefined \-L CICS_LIB_PATH/regxa_swxa.o \-lpthread -ldl -lc
    -shared -lmqzi_r -lmqmxa_r -lmqmcics_r -ldl -lc
```

Windows
Siga estas etapas:

1. Use o comando cl para construir amqzscin.obj, compilando pelo menos as seguintes variáveis:

   ```
   cl.exe -c -IEncinaPath\include -IMQ_INSTALLATION_PATH\include -Gz -LD amqzscin.c
   ```

2. Crie um arquivo de definição de módulo chamado `mqmc1415.def`, que contenha as seguintes linhas:

   ```
   LIBRARY MQMC4SWI
   EXPORTS
   CICS_XA_Init
   ```

3. Use o comando **lib** para construir um arquivo de exportação e uma biblioteca de importação, usando pelo menos a seguinte opção:

   ```
   lib -def:mqmc4swi.def -out:mqmc4swi.lib
   ```

   Se o comando lib for bem-sucedido, um arquivo `mqmc4swi.exp` também será construído.

4. Use o comando de link para construir `mqmc4swi.dll` usando pelo menos a seguinte opção:

   ```
   link.exe -dll -nod -out:mqmc4swi.dll
       amqzscin.obj CicsPath\lib\regxa_swxa.obj
       mqmc4swi.exp mqmcics4.lib
      CicsPath\lib\libcicsrt.lib
      DcePath\lib\libdce.lib DcePath\lib\pthreads.lib
      EncinaPath\lib\libEncina.lib
      EncinaPath\lib\libEncServer.lib
      msvcrt.lib kernel32.lib
   ```

Suporte de IBM WebSphere MQ XA e Tuxedo
IBM WebSphere MQ no Windows,UNIX and Linux os sistemas podem bloquear aplicativos XA coordenados pelo Tuxedo indefinidamente no xa_start

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Isso pode ocorrer apenas quando dois ou mais processos coordenados pelo Tuxedo em uma única transação global tentam acessar o IBM WebSphere MQ usando o mesmo ID de seção de transação (XID). Se o Tuxedo fornecer a cada processo na transação global um XID diferente para usar com o IBM WebSphere MQ, isso não poderá ocorrer.

Para evitar o problema, configure cada aplicativo em Tuxedo que acessa o IBM WebSphere MQ sob um identificador de transação global único (gtrid), dentro de seu próprio grupo de servidores Tuxedo. Processos no mesmo grupo de servidores usam o mesmo XID ao acessar gerenciadores de recursos em nome de um gtrid único e são, portanto, vulneráveis a bloqueios em xa_start no IBM WebSphere

MQ. Processos em diferentes grupos de servidores usam XIDs separados ao acessar gerenciadores de recursos e, portanto, não necessitam serializar seu trabalho de transação no IBM WebSphere MQ.

Ativando o processo two-phase commit do CICS
Para permitir que o CICS use um processo two-phase commit para coordenar transações que incluem chamadas MQI, inclua uma entrada de sub-rotina de definição de recurso XAD CICS na região CICS . Observe que este tópico não é aplicável a z/OS

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Aqui está um exemplo de inclusão de uma entrada de sub-rotina XAD para WebSphere MQ para Windows, em que <Drive> é a unidade na qual o WebSphere MQ está instalado (por exemplo, D:).

```
cicsadd -cxad -r<cics_region> \
     ResourceDescription="MQM XA Product Description" \
     SwitchLoadFile="<Drive>:\Program Files\IBM\WebSphere MQ\bin\mqmc4swi.dll" \
     XAOpen=<queue_manager_name>
```

Para clientes transacionais estendidos, use o arquivo de carregamento do comutador mqcc4swi.dll.

Aqui está um exemplo de inclusão de uma entrada de sub-rotina XAD para sistemas WebSphere MQ para UNIX and Linux , em que *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado:

```
cicsadd -cxad -r<cics_region> \
      ResourceDescription="MQM XA Product Description" \
      SwitchLoadFile="MQ_INSTALLATION_PATH/lib/amqzsc" \
      XAOpen=<queue_manager_name>
```

Para clientes transacionais estendidos, use o arquivo de carregamento do comutador amqczsc.

Para obter informações sobre como usar o comando **cicsadd** , consulte o *CICS Administration Reference*ou *CICS Administration Guide* para sua plataforma.

As chamadas para o WebSphere MQ podem ser incluídas em uma transação do CICS e os recursos do WebSphere MQ serão confirmados ou revertidos conforme orientado pelo CICS. Esse suporte não está disponível para aplicativos cliente.

Você ***deve*** emitir um MQCONN de sua transação CICS para acessar os recursos do WebSphere MQ , seguido por um MQDISC correspondente na saída.

Ativando saídas de usuário do CICS
Uma CICS saída de usuário *ponto* (normalmente referida como *saída de usuário*) é um local em um módulo CICS no qual o CICS pode transferir o controle para um programa que você gravou (uma saída de usuário *programa*) e no qual o CICS pode retomar o controle quando seu programa de saída tiver concluído seu trabalho.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Antes de usar uma saída de usuário CICS , leia o *CICS Administration Guide* para sua plataforma.

Amostra de Saída de Término da Tarefa
WebSphere MQ fornece código de origem de amostra para uma saída de finalização da tarefa do CICS .

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

O código fonte de amostra está nos diretórios a seguir:

Tabela 13. Saídas de finalização da tarefa CICS		
Plataforma	**Diretório**	**Arquivo de Origem**
Sistemas UNIX and Linux	*MQ_INSTALLATION_PATH*/samp	amqzscgx.c
Windows	*MQ_INSTALLATION_PATH*\Tools \c \ Amostras	amqzscgn.c

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

As instruções de construção para a saída de término da tarefa de amostra estão contidas nos comentários próximos à parte superior de cada arquivo de origem

Essa saída é chamada pelo CICS na finalização normal e anormal da tarefa (após qualquer ponto de sincronização ser obtido). Nenhum trabalho recuperável é permitido no programa de saída.

Essas funções são usadas apenas em um contexto WebSphere MQ e CICS no qual a versão do CICS suporta a interface XA. CICS se refere a essas bibliotecas como "programas" ou "saídas de usuário".

O CICS possui um número de saídas de usuário e amqzscgx, se usado, é definido e ativado no CICS como a "Saída de usuário de finalização da tarefa (UE014015)", ou seja, número de saída 15.

Quando a saída de finalização da tarefa é chamada pelo CICS, CICS já informou WebSphere MQ do estado de finalização da tarefa e o WebSphere MQ tomou a ação apropriada (confirmação ou retrocesso). Tudo o que a saída faz é emitir um MQDISC para limpar.

Um propósito de instalar e configurar seu sistema CICS para usar uma saída de finalização de tarefa é proteger seu sistema contra algumas das consequências do código do aplicativo com falha. Por exemplo, se sua transação CICS terminar de forma anormal sem primeiro chamar MQDISC e não tiver nenhuma saída de finalização de tarefa instalada, você poderá ver (em cerca de 10 segundos) uma falha irrecuperável subsequente da região CICS . Isso ocorre porque o encadeamento de funcionamento do WebSphere MQ, que é executado no processo cicsas, não foi postado e recebeu tempo para limpar e retornar. Os sintomas podem ser que o processo cicsas termina imediatamente, tendo escrito relatórios FFST para /var/mqm/errors ou o local equivalente no Windows.

Usando o Microsoft Transaction Server (COM +)

COM + (Microsoft Transaction Server) é projetado para ajudar os usuários a executar aplicativos de lógica de negócios em um servidor de camada média típico..

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Consulte Recursos que podem ser usados somente com a instalação primária no Windows para obter informações importantes

O COM+ divide o trabalho em *atividades*, que normalmente são partes de lógica de negócios independentes curtas, como *transferência de fundos da conta A para a conta B*. O COM+ depende fortemente da orientação de objeto e, em especial, do COM; em geral uma atividade COM+ é representada por um objeto COM (negócios).

COM+ é uma parte integrada do sistema operacional. Para usar COM + no Windows 2000 e Windows XP, é necessário o Hotfix Q313582 (também conhecido como Pacote de Rolagem COM + 19.1).

O COM+ fornece três serviços para o administrador do objeto de negócios, removendo grande parte das preocupações do programador do objeto de negócios:

- Gerenciamento de Transação
- Segurança
- Conjunto de Recursos

Geralmente, você usa COM + com código de front-end que é um cliente COM para os objetos mantidos no COM + e serviços de backend, como um banco de dados, com WebSphere MQ de ponte entre o objeto de negócios COM + e o backend.

O código front-end pode ser um programa independente ou um Active Server Page (ASP) hospedado pelo Microsoft Internet Information Server (IIS). O código front-end pode estar no mesmo computador que COM + e seus objetos de negócios, com conexão através COM. Alternativamente, o código front-end pode estar em um computador diferente, com conexão através de DCOM. É possível usar diferentes clientes para acessar o mesmo objeto de negócios COM+ em diferentes situações.

O código de backend pode estar no mesmo computador que COM + e seus objetos de negócios, ou em um computador diferente com conexão por meio de qualquer um dos protocolos suportados do WebSphere MQ

Expirando Unidades de Trabalho Globais

O gerenciador de filas pode ser configurado para expirar unidades de trabalho globais após um intervalo de inatividade pré-configurado.

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

Para ativar este comportamento, configure as seguintes variáveis de ambiente:

- *AMQ_TRANSACTION_EXPIRY_RESCAN*= < intervalo de varredura em milissegundos>
- *AMQ_XA_TRANSACTION_EXPIRE*= < intervalo de tempo limite em milissegundos>

Atenção: As variáveis de ambiente afetam apenas as transações que estão no estado *Inativo* na tabela 6-4 da Especificação XA Ou seja, transações que não estão associadas a nenhum encadeamento de aplicativos, mas para as quais o software externo do Gerenciador de transações ainda não chamou a chamada de função **`xa_prepare`**.

Gerenciadores de transações externos mantêm apenas um log de transações preparadas, confirmadas ou recuperadas. Se o gerenciador de transações externas ficar inativo por algum motivo, em seu retorno ele direciona transações preparadas, confirmadas e recuperadas para conclusão, mas quaisquer transações ativas que ainda precisem ser preparadas se tornam órfãs. Para evitar isso, configure o *AMQ_XA_TRANSACTION_EXPIRY* para permitir que o intervalo esperado entre um aplicativo que esteja fazendo chamadas API transacionais do MQI e concluindo a transação, tenha realizado o trabalho transacional em outros gerenciadores de recursos.

Para assegurar uma limpeza em tempo hábil após o *AMQ_XA_TRANSACTION_EXPIRY* expirar, configure o valor *AMQ_TRANSACTION_EXPIRY_RESCAN* para um valor mais baixo que o intervalo de *AMQ_XA_TRANSACTION_EXPIRY*, idealmente para que a nova varredura ocorra mais de uma vez dentro do intervalo *AMQ_XA_TRANSACTION_EXPIRY*.

Unidade de Disposição de Recuperação

WebSphere MQ for z/OS fornece disposições de unidade de recuperação. Este recurso permite que você configure se a segunda fase das transações confirmadas de duas fases poderá ser orientada, por exemplo, durante a recuperação, quando conectada a outro gerenciador de filas dentro do mesmo grupo de filas compartilhadas (QSG).

Nota: Este tópico também está disponível no IBM MQ Version 8.0 e nas versões mais recentes No entanto, não é possível alternar para uma versão posterior usando a caixa de listagem "Alterar versão" . Para acessar o tópico em uma versão posterior, edite o número da versão na caixa URL em seu navegador.

WebSphere MQ for z/OS V7.0.1 e mais recente suporta a disposição da unidade de recuperação

Unidade de Disposição de Recuperação
A disposição da unidade de recuperação está relacionada a uma conexão do aplicativo e subsequentemente quaisquer transações iniciadas por ele. Há duas disposições de unidade de recuperação possíveis.

- Uma disposição da unidade de recuperação GROUP identifica que um aplicativo transacional está logicamente conectado ao grupo de filas compartilhadas não tem uma afinidade a qualquer gerenciador de filas específico. Qualquer transação de confirmação de 2 fases que ele iniciar e que tenha concluído a fase 1 do processo de confirmação ou seja, que esteja em dúvida, pode ser consultada e resolvida, quando conectada a qualquer gerenciador de filas no QSG. Em um cenário de recuperação isso significa que o coordenador de transação não precisa se reconectar ao mesmo gerenciador de filas, que pode estar indisponível.
- Uma disposição de unidade de recuperação QMGR identifica que um aplicativo possui uma afinidade direta com o gerenciador de filas ao qual ele está conectado e quaisquer transações que ele inicia também possuirão essa disposição.

  Em um cenário de recuperação, o coordenador de transação deverá se reconectar ao mesmo gerenciador de filas para consultar e resolver qualquer transação em dúvida, sem restrição com a possibilidade do gerenciador de filas pertencer a um grupo de filas compartilhadas.

Decidindo qual linguagem de programação usar

Use estas informações para descobrir sobre linguagens de programação e estruturas suportadas pelo IBM WebSphere MQe algumas considerações para usá-las.

O IBM WebSphere MQ fornece suporte para as seguintes linguagens processuais de programação:

- C
- Visual Basic (apenas sistemas Windows)
- COBOL

Esses idiomas usam a interface da fila de mensagens (MQI) para acessar serviços de enfileiramento de mensagens. Para obter mais informações sobre o suporte para esses idiomas, consulte “Usando linguagens processuais com o WebSphere MQ” na página 80

IBM WebSphere MQ fornece suporte para:

- .REDE
- ActiveX
- C++
- Java
- JMS

Essas linguagens usam o IBM WebSphere MQ Object Model, que fornece classes que fornecem a mesma funcionalidade que as chamadas e estruturas do WebSphere MQ , mas que são uma maneira mais natural de programação em um ambiente orientado a objetos. Algumas das linguagens que usam o IBM WebSphere MQ Object Model fornecem funções adicionais que não estão disponíveis na interface da fila de mensagens (MQI). Para obter mais informações sobre o suporte para esses idiomas, consulte “Programação orientada a objetos com WebSphere MQ” na página 81

Usando linguagens processuais com o WebSphere MQ

Para obter informações detalhadas sobre como gravar seus aplicativos no idioma escolhido, consulte os links a seguir:

- “Codificação em C” na página 84
- “Codificação em Visual Basic” na página 88
- “Codificação em COBOL” na página 87

Para obter uma visão geral da interface de chamada para linguagens processuais, consulte Descrições de chamada. Este tópico contém uma lista das chamadas MQI e cada chamada mostra como codificar as chamadas em cada um dessas linguagens.

WebSphere MQ fornece arquivos de definição de dados para ajudar a gravar seus aplicativos. Para obter uma descrição integral, consulte “IBM WebSphere MQ arquivos de definição de dados” na página 82.

Se você puder escolher em qual idioma codificar seus programas, considere o comprimento máximo das mensagens que seus programas processarão. Se seus programas processarem apenas mensagens de um comprimento máximo conhecido, você poderá codificá-las em qualquer uma das linguagens de programação suportadas. Mas se você não souber o comprimento máximo das mensagens que os programas terão que processar, o idioma escolhido dependerá se você estiver gravando um aplicativo CICS, IMSou em lote:

IMS e lote
Codifique os programas na linguagem C, PL/I ou assembler para usar os recursos que essas linguagens oferecem para obter e liberar quantias de memória arbitrárias. Como alternativa, você poderia codificar seus programas em COBOL, mas usar sub-rotinas na linguagem assembler, PL/I ou C para obter e liberar armazenamento.

CICS
Codifique os programas em qualquer idioma suportado por CICS A interface EXEC CICS fornece as chamadas para gerenciar memória, se necessário.

Programação orientada a objetos com WebSphere MQ

Algumas das linguagens e estruturas de programação que usam o IBM WebSphere MQ Object Model fornecem funções adicionais que não estão disponíveis na interface da fila de mensagens (MQI). Para obter detalhes das classes, métodos e propriedades fornecidos pelo IBM WebSphere MQ Modelo de Objeto, consulte “O Modelo de Objeto IBM WebSphere MQ” na página 89

.REDE
Consulte Usando .NET para obter informações sobre como codificar programas .NET usando as classes do WebSphere MQ .NET. Message Service Clients for C/C++ and .NET fornecem uma interface de programação de aplicativos (API) chamada XMS que possui o mesmo conjunto de interfaces que o Java Message Service (JMS) API.

C++
IBM WebSphere MQ fornece classes C++ equivalentes a objetos WebSphere MQ e algumas classes adicionais equivalentes aos tipos de dados da matriz. Ele fornece inúmeros recursos não disponíveis por meio do MQI. Consulte Using C++ para obter informações sobre como codificar programas usando o WebSphere MQ Object Model em C + +. Message Service Clients for C/C++ and .NET fornecem uma interface de programação de aplicativos (API) chamada XMS que possui o mesmo conjunto de interfaces que o Java Message Service (JMS) API.

Java
Consulte Usando Java para obter informações sobre como codificar programas usando o Modelo de Objeto WebSphere MQ em Java Para obter informações sobre as diferenças entre as classes IBM WebSphere MQ para Java e IBM WebSphere MQ para ajudar a decidir qual usar, consulte “Devo usar classes IBM WebSphere MQ para Java ou classes IBM WebSphere MQ para JMS.” na página 91.

JMS
O Websphere MQ também fornece classes que implementam a especificação Java Message Service (JMS). Para obter detalhes das classes do WebSphere MQ para JMS, consulte Usando Java. Para obter informações sobre as diferenças entre as classes IBM WebSphere MQ para Java e as classes IBM WebSphere MQ para ajudar a decidir qual usar, consulte “Devo usar classes IBM WebSphere MQ para Java ou classes IBM WebSphere MQ para JMS.” na página 91.

Message Service Clients for C/C++ and .NET fornecem uma interface de programação de aplicativos (API) chamada XMS que possui o mesmo conjunto de interfaces que o Java Message Service (JMS) API.

ActiveX
O WebSphere MQ ActiveX é comumente conhecido como MQAX. O MQAX é incluído como parte do WebSphere MQ para Windows.Suporte para ActiveX foi estabilizado no nível do WebSphere MQ Versão 6.0 . Para explorar recursos introduzidos no WebSphere MQ mais recente do que a Versão 6.0, considere usar .NET em vez disso. Consulte Usando a Interface do modelo de objeto componente

(Classes de automação do WebSphere MQ for ActiveX) para obter informações sobre como codificar programas usando o WebSphere MQ Object Model em ActiveX.

Conceitos relacionados

Visão geral técnica

“Desenvolvendo Aplicativos” na página 7
O IBM WebSphere MQ fornece várias maneiras nas quais é possível desenvolver aplicativos para enviar e receber mensagens necessárias para suportar seus processos de negócios. É possível também desenvolver os aplicativos para gerenciar os gerenciadores de fila e recursos relacionados.

“Conceitos de desenvolvimento de aplicativos” na página 8
É possível usar uma opção de linguagens orientadas a objeto ou processuais para gravar aplicativos IBM WebSphere MQ. Use os links neste tópico para obter informações sobre conceitos do IBM WebSphere MQ que são úteis para desenvolvedores de aplicativos.

Referências relacionadas

Referência de desenvolvimento de aplicativos

IBM WebSphere MQ arquivos de definição de dados

O IBM WebSphere MQ fornece arquivos de definição de dados que ajudam a gravar seus aplicativos.

Os arquivos de definição de dados também são conhecidos como:

Idioma	Definições de dados
C	Arquivos de inclusão ou arquivos de cabeçalho
Visual Basic	Arquivos de módulo (apenas versões de 32 bits)
COBOL	Arquivos de cópia
Assembler	Macros
PL/I	Arquivos de inclusão

Os arquivos de definição de dados para ajudá-lo a gravar saídas de canal são descritos em WebSphere MQ COPY, header, include e module files

Os arquivos de definição de dados que ajudam a gravar as saídas de serviços instaláveis estão descritos em “Saídas de usuário, saídas de API e WebSphere MQ serviços instaláveis” na página 379.

Para os arquivos de definição de dados suportados em C++, consulte Using C++.

Os nomes dos arquivos de definição de dados possuem o prefixo CMQ e um sufixo que é determinado pela linguagem de programação:

Sufixo	Idioma
a	Linguagem assembler
b	Visual Basic
c	C
l	COBOL (sem valores inicializados)
p	PL/I
v	COBOL (com valores padrão configurados)

Biblioteca de instalação

O nome **thlqual** é o qualificador de alto nível da biblioteca de instalação no z/OS

Este tópico apresenta arquivos de definição de dados do WebSphere MQ , sob estes títulos:

Arquivos de inclusão da Linguagem C

Os arquivos de inclusão do WebSphere MQ C são listados em Arquivos de cabeçalho C Eles são instalados nos diretórios ou bibliotecas a seguir:

Plataforma	Diretório de instalação ou biblioteca
Plataformas UNIX	*MQ_INSTALLATION_PATH*/inc/
Sistemas Windows	*MQ_INSTALLATION_PATH*\Tools\c\include

em que *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado

Nota: Para plataformas UNIX , os arquivos include são simbolicamente vinculados ao `/usr/include`

Para obter mais informações sobre a estrutura de diretórios, consulte Planejando o suporte do sistema de arquivos

Arquivos de módulo Visual Basic

WebSphere MQ para Windows fornece quatro arquivos do módulo Visual Basic.

Ele são listados em Arquivos de módulo do Visual Basic e instalados em

```
MQ_INSTALLATION_PATH\Tools\Samples\VB\Include
```

Arquivos de cópia COBOL

Para COBOL, WebSphere MQ fornece arquivos de cópia separados contendo as constantes nomeadas e dois arquivos de cópia para cada uma das estruturas.

Existem dois arquivos de cópia para cada estrutura porque cada um é fornecido com e sem os valores iniciais:

- Em WORKING-STORAGE SECTION de um programa COBOL, use os arquivos que inicializam os campos de estrutura para os valores padrão. Essas estruturas são definidas nos arquivos de cópia que possuem nomes com sufixo de letra V (valores).
- Em LINKAGE SECTION de um programa COBOL, use as estruturas sem os valores iniciais. Essas estruturas são definidas nos arquivos de cópia que possuem nomes com sufixo de letra L (ligação).

Os arquivos de cópia COBOL do WebSphere MQ são listados em Arquivos COBOL COPY Eles são instalados nos diretórios a seguir:

Plataforma	Diretório de instalação ou biblioteca
Outras plataformas UNIX	*MQ_INSTALLATION_PATH*/inc/
Windows	*MQ_INSTALLATION_PATH*\Tools\cobol\copybook (para Micro Focus COBOL) *MQ_INSTALLATION_PATH*\Tools\cobol\copybook\VAcobol (para IBM VisualAge COBOL)

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

Inclua em seu programa apenas os arquivos que precisar. Faça isso com uma ou mais instruções COPY após uma declaração de nível 01. Isso significa que é possível incluir diversas versões das estruturas em um programa, se necessário. Observe que CMQV é um grande arquivo.

Segue um exemplo de código COBOL para incluir o arquivo de cópia CMQMDV:

```
01 MQM-MESSAGE-DESCRIPTOR.
   COPY CMQMDV.
```

Cada declaração de estrutura inicia com um item de nível 01; é possível declarar diversas instâncias da estruturas codificando a declaração de nível 01 seguida por uma instrução COPY para copiar no restante da declaração de estrutura. Para consultar a instância apropriada, use a palavra-chave IN.

Segue um exemplo de código COBOL para incluir duas instâncias de CMQMDV:

```
* Declare two instances of MQMD
 01  MY-CMQMD.
     COPY CMQMDV.
 01  MY-OTHER-CMQMD.
     COPY CMQMDV.
*
* Set MSGTYPE field in MY-OTHER-CMQMD
     MOVE MQMT-REQUEST TO MQMD-MSGTYPE IN MY-OTHER-CMQMD.
```

Alinhe as estruturas em limites de 4 bytes. Se você usar a instrução COPY para incluir uma estrutura que segue um item que não seja o item de nível 01, certifique-se de que a estrutura seja um múltiplo de 4 bytes do início do item de nível 01. Se isso não for feito, você pode reduzir o desempenho do seu aplicativo.

As estruturas são descritas em Tipos de dados usados no MQI. As descrições dos campos nas estruturas mostram os nomes de campos sem um prefixo. Em programas COBOL, prefixe os nomes de campo com o nome da estrutura seguido por um hífen, conforme mostrado nas declarações COBOL. Os campos nos arquivos de cópia de estrutura recebem o prefixo desta maneira.

Os nomes de campo nas declarações nos arquivos de cópia da estrutura estão em maiúsculas. É possível usar maiúsculas ou minúsculas mistas no lugar. Por exemplo, o campo *StrucId* da estrutura MQGMO é mostrado como MQGMO-STRUCID na declaração COBOL e no arquivo de cópia.

As estruturas de sufixo V são declaradas com valores iniciais para todos os campos, portanto, você precisa configurar apenas os campos nos quais o valor necessário seja diferente do valor inicial.

Codificação em C

Observe as informações nas seções a seguir ao codificar programas WebSphere MQ em C.

Parâmetros das chamadas MQI

Os parâmetros que são *somente entrada* e do tipo MQHCONN, MQHOBJ, MQHMSG ou MQLONG são passados por valor; para todos os outros parâmetros, o *endereço* do parâmetro é passado por valor.

Nem todos os parâmetros que são passados por endereço precisam ser especificados toda vez que uma função for chamada. Quando um parâmetro específico não for necessário, um ponteiro nulo pode ser especificado como o parâmetro na chamada de função, no lugar do endereço dos dados do parâmetro. Parâmetros para os quais isso é possível estão identificados nas descrições de chamada.

Nenhum parâmetro é retornado como o valor da função; na terminologia de C, isso significa que todas as funções retornam nulo.

Os atributos da função são definidos pela variável de macro MQENTRY; o valor dessa variável de macro depende do ambiente.

Parâmetros com o tipo de dados indefinido

As funções MQGET, MQPUT e MQPUT1 têm, cada uma delas, um parâmetro *Buffer* que tem um tipo de dados indefinido. Esse parâmetro é usado para enviar e receber os dados da mensagem do aplicativo.

Parâmetros desse tipo são mostrados nos exemplos de C como matrizes de MQBYTE. É possível declarar os parâmetros dessa maneira, mas geralmente é mais conveniente declará-los como a estrutura que descreve o layout dos dados na mensagem. O parâmetro da função é declarado como um ponteiro para nulo e, portanto, o endereço de quaisquer dados pode ser especificado como o parâmetro na chamada de função.

Tipos de dados

Todos os tipos de dados são definidos com a instrução `typedef`.

Para cada tipo de dados, o tipo de dados do ponteiro correspondente também é definido. O nome do tipo de dados do ponteiro é o nome do tipo de dados elementar ou de estrutura com o prefixo P para denotar um ponteiro. Os atributos do ponteiro são definidos pela variável de macro MQPOINTER; o valor dessa variável de macro depende do ambiente. O código a seguir ilustra como declarar tipos de dados do ponteiro:

```
#define MQPOINTER                   /* depends on environment */
...
typedef MQLONG  MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD   MQPOINTER PMQMD;     /* pointer to MQMD */
```

Manipulando sequências binárias

Sequências de dados binários são declaradas como um dos tipos de dados MQBYTEn.

Sempre que copiar, comparar ou configurar campos desse tipo, use as funções C `memcpy`, `memcmp`ou `memset`:

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

MQMD MyMsgDesc;

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

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

Não use as funções de sequência strcpy, strcmp, strncpy ou strncmp, pois elas não funcionam corretamente com os dados declarados como MQBYTE24.

Manipulação de sequências de caracteres

Quando o gerenciador de filas retornar dados de caracteres para o aplicativo, o gerenciador de filas sempre preencherá os dados de caracteres com espaços em branco para o comprimento definido do campo. O gerenciador de filas não retorna sequências terminadas em nulo, mas é possível usá-las em sua entrada. Portanto, ao copiar, comparar concatenar essas sequências, use as funções de sequência strncpy, strncmp ou strncat.

Não use as funções de sequência que requerem a sequência a ser finalizada por um nulo (strcpy, strcmp e strxfrm). Além disso, não use a função strlen para determinar o comprimento da sequência; use em vez disso a função sizeof para determinar o comprimento do campo.

Valores iniciais para estruturas

O arquivo include <cmqc.h> define várias variáveis de macro que podem ser usadas para fornecer valores iniciais para as estruturas ao declarar instâncias dessas estruturas. Essas variáveis de macro têm nomes no formato MQxxx_DEFAULT, em que MQxxx representa o nome da estrutura. Use-as desta forma:

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

Para alguns campos de caracteres, a MQI define valores específicos válidos (por exemplo, para os campos *StrucId* ou para o campo *Format* no MQMD). Para cada um dos valores válidos, duas variáveis de macro são fornecidas:

- Uma variável de macro define o valor como uma sequência com um comprimento, excluindo o nulo implícito, que corresponda exatamente ao comprimento definido do campo. Por exemplo, o símbolo ¬ representa um caractere em branco:

  ```
  #define MQMD_STRUC_ID "MD¬¬"
  #define MQFMT_STRING "MQSTR¬¬¬"
  ```

  Use esse formato com as funções memcpy e memcmp.
- A outra variável de macro define o valor como uma matriz de char; o nome dessa variável de macro é o nome da forma de sequência com o sufixo _ARRAY. Por exemplo:

  ```
  #define MQMD_STRUC_ID_ARRAY 'M','D','¬','¬'
  #define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
  ```

  Use esse formato para inicializar o campo quando uma instância da estrutura for declarada com valores diferentes dos fornecidos pela variável de macro MQMD_DEFAULT.

Valores iniciais para as estruturas dinâmicas

Quando um número variável de instâncias de uma estrutura é necessário, as instâncias são geralmente criadas no armazenamento principal obtido dinamicamente usando as funções calloc ou malloc.

Para inicializar os campos nessas estruturas, a técnica a seguir é recomendada:

1. Declare uma instância da estrutura usando a variável de macro MQxxx_DEFAULT apropriada para inicializar a estrutura. Esta instância se torna o *modelo* para outras instâncias:

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

   Codifique as palavras-chave static e auto na declaração para fornecer à instância modelo tempo de vida estático ou dinâmico, conforme necessário.
2. Use as funções calloc ou malloc para obter armazenamento para uma instância dinâmica da estrutura:

   ```
   PMQMD InstancePtr;
   InstancePtr = malloc(sizeof(MQMD));
                                    /* get storage for dynamic instance */
   ```
3. Use a função memcpy para copiar a instância modelo para a instância dinâmica:

   ```
   memcpy(InstancePtr,&ModelMsgDesc,sizeof(MQMD));
                                    /* initialize dynamic instance */
   ```

Uso de C++

Para a linguagem de programação C++, os arquivos de cabeçalho contêm as instruções adicionais a seguir que são incluídas somente quando um compilador C++ é usado:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Codificação em COBOL

Observe as informações na seção a seguir ao codificar programas do WebSphere MQ em COBOL.

Constantes nomeadas

Os nomes de constantes são mostrados contendo o caractere sublinhado (_) como parte do nome. Em COBOL, deve-se usar o caractere de hífen (-) no lugar do sublinhado. As constantes que possuem valores de sequência de caracteres usam o caractere de aspas simples (') como o delimitador de sequência. Para fazer o compilador aceitar esse caractere, use a opção do compilador APOST.

O arquivo de cópia CMQV contém declarações de constantes nomeadas como itens de nível 10. Para usar as constantes, declare o item nível 01 explicitamente, em seguida, use a instrução COPY para copiar nas declarações das constantes:

```
 WORKING-STORAGE SECTION.
 01  MQM-CONSTANTS.
     COPY CMQV.
```

No entanto, este método faz com que as constantes ocupem o armazenamento no programa, mesmo se elas não forem referidas. Se as constantes forem incluídas em muitos programas separados dentro da mesma unidade de execução, diversas cópias de constantes existirão; isso pode resultar em uma quantidade significativa de armazenamento principal que está sendo usado. É possível evitar isso incluindo a cláusula GLOBAL na declaração de nível 01:

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

Isso aloca armazenamento para apenas *um* conjunto de constantes dentro da unidade de execução; as constantes, no entanto, podem ser referenciadas por *qualquer* programa na unidade de execução, não apenas o programa que contém a declaração nível 01.

Assegurando o alinhamento da estrutura

Deve-se tomar cuidado para assegurar que as estruturas do IBM WebSphere MQ que são transmitidas para iniciar nas chamadas do MQ devem ser alinhadas nos limites de palavra. Um limite de palavra é 4 bytes para processos de 32 bits, 8 bytes para processos de 64 bits e 16 bytes para processos de 128 bits (IBM i).

Quando possível, coloque todas as estruturas do IBM WebSphere MQ juntas para que sejam todas alinhadas por limite.

Codificação em pTAL

Observe as informações na seção a seguir ao codificar programas do IBM WebSphere MQ em pTAL.

HP Integrity NonStop Server

Definindo e inicializando estruturas de IBM WebSphere MQ

As definições de estrutura pTAL para estruturas de IBM WebSphere MQ são fornecidas com nomes que terminam com ^DEF. Por exemplo, as seguintes declarações pTAL seriam codificadas para criar uma estrutura do IBM WebSphere MQ Message Descriptor (MQMD) e uma estrutura do IBM WebSphere MQ Put Message Options (MQPMO).

```
STRUCT MYMD(MQMD^DEF);        ! Declare an MQMD structure
STRUCT MYPMO(MQPMO^DEF);      ! Declare an MQPMO structure
```

IBM WebSphere MQ fornece pTAL DEFINE com nomes que terminam com ^DEFAULT para inicializar as estruturas de IBM WebSphere MQ com valores padrão. As seguintes instruções de pTAL são codificadas para designar valores padrão para o MQMD declarado e as estruturas MQPMO:

```
MQMD^DEFAULT(MYMD);           ! Assign default values to an MQMD structure
MQPMO^DEFAULT(MYPMO);         ! Assign default values to an MQPMO structure
```

É possível declarar e inicializar outras estruturas de IBM WebSphere MQ usando código similar.

pTAL e o CRE

Os programas pTAL não podem inicializar o Ambiente de tempo de execução comum e, portanto, eles devem ser usados com uma rotina principal em linguagem C ou COBOL.

As amostras de pTAL fornecidas com o IBM WebSphere MQ usam uma rotina de linha principal da linguagem C é chamada AMQSPTM0.C

Parâmetros com tipo de dados MQCHAR

Os procedimentos MQGET, MQPUT e MQPUT1 cada possui um parâmetro **Buffer** que possui um tipo de dados MQCHAR .EXT. Esse parâmetro é usado para enviar e receber os dados da mensagem do aplicativo.

Os parâmetros dessa classificação são mostrados nas amostras de pTAL como matrizes de sequência. É possível declarar os parâmetros dessa maneira, mas geralmente é mais conveniente declará-los como a estrutura que descreve o layout dos dados na mensagem. O parâmetro do procedimento é declarado como um MQCHAR .EXT, mas o endereço de quaisquer dados pode ser especificado como o parâmetro na chamada de procedimento.

Manipulação de sequências de caracteres

Quando o gerenciador de filas retornar dados de caracteres para o aplicativo, o gerenciador de filas sempre preencherá os dados de caracteres com espaços em branco para o comprimento definido do campo. O gerenciador de filas não retorna sequências terminadas em nulo, mas é possível usá-las em sua entrada.

Codificação em Visual Basic

Observe as informações na seção a seguir ao codificar programas WebSphere MQ no Visual Basic.

Nota: Fora do ambiente .NET, o suporte para Visual Basic (VB) no WebSphere MQ foi estabilizado no nível V6.0 . A maioria das novas funções incluídas no WebSphere MQ 7.0 ou mais recente não está disponível para aplicativos VB Se você estiver programando no VB.NET, use as classes .NET do WebSphere MQ Para obter mais informações, consulte Usando .NET..

O Visual Basic é suportado somente no Windows

Para evitar a conversão indesejada de dados binários que passam entre o Visual Basic e o WebSphere MQ, use uma definição MQBYTE em vez de MQSTRING CMQB.BAS define vários novos tipos MQBYTE que são equivalentes a uma definição de byte C e os usa nas estruturas WebSphere MQ . Por exemplo,

para a estrutura MQMD (descritor de mensagens), MsgId (identificador de mensagem) é definido como MQBYTE24.

O Visual Basic não possui um tipo de dados de ponteiro, portanto, as referências a outras estruturas de dados WebSphere MQ são por deslocamento em vez de ponteiro. Declare uma estrutura composta que consiste das duas estruturas de componente e especifique a estrutura composta na chamada. O suporte do WebSphere MQ para Visual Basic fornece uma chamada MQCONNXAny para tornar isso possível e permitir que aplicativos clientes especifiquem as propriedades do canal em uma conexão do cliente. Ele aceita uma estrutura sem tipo (MQCNOCD) no lugar da estrutura MQCNO típica.

A estrutura MQCNOCD é uma estrutura composta que consiste em um MQCNO seguido por um MQCD. Esta estrutura é declarada no arquivo de cabeçalho saídas CMQXB. Use a rotina MQCNOCD_DEFAULTS para inicializar uma estrutura MQCNOCD. Uma amostra realizando chamadas MQCONNX fornecida (amqscnxb.vbp).

MQCONNXAny possui os mesmos parâmetros que MQCONNX, exceto o fato de que o parâmetro *ConnectOpts* é declarado como sendo do tipo qualquer tipo de dados em vez de dados de tipo MQCNO. Isso permite que a função aceite tanto a estrutura MQCNO como a MQCNOCD. Essa função é declarada no arquivo de cabeçalho principal CMQB.

O Modelo de Objeto IBM WebSphere MQ

O IBM WebSphere MQ Object Model consiste em classes, métodos e propriedades. Use estas informações para aprender sobre cada um desses conceitos.

O IBM WebSphere MQ Object Model consiste no seguinte:

- *Classes* que representam conceitos familiares do WebSphere MQ , como gerenciadores de fila, filas e mensagens
- *Métodos* em cada classe correspondendo a chamadas MQI.
- *Propriedades* em cada classe correspondente a atributos de objetos do WebSphere MQ

Ao criar um aplicativo do WebSphere MQ usando o Modelo de Objeto do WebSphere MQ , você cria instâncias dessas classes no programa. Uma instância de uma classe em programação orientada a objetos é chamada de *objeto*. Quando um objeto tiver sido criado, você interage com o objeto examinando ou configurando os valores das propriedades do objeto (o equivalente a emitir uma chamada MQINQ ou MQSET) e fazendo chamadas de método com relação ao objeto (o equivalente a emitir as outras chamadas MQI).

Estes tópicos descrevem cada um dos Modelos de Objeto do WebSphere MQ em detalhes:

- "Classes" na página 89
- "Referências do objeto" na página 90
- "Códigos de retorno" na página 90

Classes

O Modelo de Objeto do WebSphere MQ fornece o conjunto base de classes a seguir:

A implementação real do modelo varia um pouco entre os diferentes ambientes suportados orientados a objetos.

MQQueueManager
Um objeto da classe MQQueueManager representa uma conexão com um gerenciador de filas. Ele tem métodos para Connect(), Disconnect(), Commit() e Backout() (o equivalente de MQCONN ou MQCONNX, MQDISC, MQCMIT e MQBACK). Ele tem propriedades correspondentes aos atributos de um gerenciador de filas. Acessar uma propriedade de atributo do gerenciador de filas implicitamente conecta ao gerenciador de filas se ainda não estiver conectado. Destruir um objeto MQQueueManager implicitamente desconecta do gerenciador de filas.

MQQueue
Um objeto da classe MQQueue representa uma fila. Ele tem métodos para efetuar Put() e Get() de mensagens na e da fila (o equivalente de MQPUT e MQGET). Ele tem propriedades correspondentes aos atributos de uma fila. Acessar uma propriedade de atributo da fila ou emitir uma chamada de método Put() ou Get(), implicitamente abre a fila (o equivalente de MQOPEN). Destruir um objeto MQQueue implicitamente fecha a fila (o equivalente de MQCLOSE).

MQTopic
Um objeto da classe MQTopic representa um tópico. Ele tem métodos para efetuar Put() (publicar) e Get() (receber ou assinar) mensagens do tópico (o equivalente de MQPUT e MQGET). Ele tem propriedades correspondentes aos atributos de um tópico. Um objeto MQTopic só pode ser acessado para publicação ou assinatura, não ambas simultaneamente. Quando usado para receber mensagens, o objeto MQTopic pode ser criado com uma assinatura não gerenciada ou gerenciada e como um assinante durável ou não durável - diversos construtores sobrecarregados são fornecidos para esses cenários diferentes.

MQMessage
Um objeto da classe MQMessage representa uma mensagem a ser colocada em uma fila ou obtida de uma fila. Ele contém um buffer e engloba dados do aplicativo e MQMD. Ele tem propriedades correspondentes aos campos do MQMD e métodos que permitem gravar e ler dados do usuário de tipos diferentes (por exemplo, sequências, números inteiros longos, números inteiros curtos, bytes únicos) no buffer e a partir dele.

MQPutMessageOptions
Um objeto da classe MQPutMessageOptions representa a estrutura MQPMO. Ele tem propriedades correspondentes aos campos do MQPMO.

MQGetMessageOptions
Um objeto da classe MQGetMessageOptions representa a estrutura MQGMO. Ele tem propriedades correspondentes aos campos do MQGMO.

MQProcess
Um objeto da classe MQProcess representa uma definição de processo (usada com acionamento). Ele tem propriedades que representam os atributos de uma definição de processo.

MQDistributionList
Um objeto da classe MQDistributionList representa uma lista de distribuição (usada para enviar múltiplas mensagens com um único MQPUT). Ele contém uma lista de objetos MQDistributionListItem.

MQDistributionListItem
Um objeto da classe MQDistributionListItem representa um único destino de lista de distribuição. Ele contém as estruturas MQOR, MQRR e MQPMR e tem propriedades correspondentes aos campos dessas estruturas.

Referências do objeto

Em um programa do WebSphere MQ que usa o MQI, o WebSphere MQ retorna identificações de conexões e identificações de objetos para o programa

Esses identificadores devem ser transmitidos como parâmetros em chamadas subsequentes do WebSphere MQ . Com o Modelo de Objeto do WebSphere MQ , essas identificações são ocultadas do programa de aplicativo Em vez disso, a criação de um objeto a partir de uma classe resulta em uma referência do objeto que está sendo retornada ao programa de aplicativo. É esta referência do objeto que é usada ao fazer chamadas de método e acessos de propriedade com relação ao objeto.

Códigos de retorno

Emitir uma chamada de método ou configurar um valor de propriedade resulta em códigos de retorno serem configurados.

Esses códigos de retorno são um código de conclusão e um código de razão e são eles próprios propriedades do objeto. Os valores de código de conclusão e de código de razão são os mesmos que

aqueles definidos para a MQI, com alguns valores adicionais específicos para o ambiente orientado a objetos.

Devo usar classes IBM WebSphere MQ para Java ou classes IBM WebSphere MQ para JMS.

Um aplicativo Java pode usar classes IBM WebSphere MQ para Java ou classes IBM WebSphere MQ para JMS para acessar recursos IBM WebSphere MQ . Cada abordagem tem suas vantagens.

As classes IBM WebSphere MQ para Java encapsulam a Message Queue Interface (MQI), a API nativa do IBM WebSphere MQ e usam o mesmo modelo de objeto que outras interfaces orientadas a objetos, enquanto as classes IBM WebSphere MQ para o Java Message Service implementa as interfaces do Serviço de Mensagens Java (JMS) da Sun

Se você estiver familiarizado com IBM WebSphere MQ em ambientes diferentes de Java, usando linguagens processuais ou orientadas a objetos, será possível transferir seu conhecimento existente para o ambiente Java usando classes IBM WebSphere MQ para Java. Também é possível explorar o intervalo completo de recursos do IBM WebSphere MQ, nem todos os quais estão disponíveis nas classes IBM WebSphere MQ para JMS

Se você não estiver familiarizado com o IBM WebSphere MQou já tiver experiência JMS, poderá achar mais fácil usar a API JMS familiar para acessar recursos IBM WebSphere MQ , usando classes IBM WebSphere MQ para JMS. O JMS também é uma parte integral da plataforma Java Platform, Enterprise Edition (Java EE). Os aplicativos Java EE podem usar beans acionados por mensagens (MDBs) para processar mensagens de forma assíncrona e MDBs podem processar apenas mensagens JMS. JMS também é o mecanismo padrão para Java EE para interagir com sistemas de mensagens assíncronos, como IBM WebSphere MQ. Cada servidor de aplicativos que é compatível com Java EE deve incluir um provedor JMS, portanto, é possível usar JMS para se comunicar entre diferentes servidores de aplicativos ou é possível portar um aplicativo de um provedor JMS para outro sem qualquer mudança no aplicativo.

Projetando aplicativos IBM WebSphere MQ

Quando você tiver decidido como seus aplicativos podem aproveitar as plataformas e ambientes disponíveis para você, será necessário decidir como usar os recursos oferecidos pelo WebSphere MQ.

Ao projetar um aplicativo IBM WebSphere MQ considere as questões e opções a seguir:

Tipo de aplicativo

Qual é o propósito do seu aplicativo? Consulte os links a seguir para obter informações sobre os diferentes tipos de aplicativos que é possível desenvolver:

- Servidor
- Client
- Publicação/assinatura
- Serviços da Web
- Saídas de usuário, saídas de API e serviços instaláveis

Além disso, também é possível escrever seus próprios aplicativos para automatizar a administração do IBM WebSphere MQ. Para obter mais informações, consulte Introdução ao WebSphere MQ Administration Interface (MQAI) e Automatizando tarefas de administração .

Linguagem de programação

O IBM WebSphere MQ suporta inúmeras linguagens de programação processuais e orientadas por objetos para escrever aplicativos. Para obter mais informações, consulte, “Decidindo qual linguagem de programação usar” na página 80

Aplicativos para mais de uma plataforma

Seu aplicativo será executado em mais de uma plataforma? Você tem uma estratégia para mudar para uma plataforma diferente daquela que usa hoje? Se a resposta para uma dessas perguntas for sim, assegure que você codifique seus programas para independência de plataforma.

Se você estiver usando C, código no padrão ANSI C. Use uma função padrão de biblioteca C em vez de uma função específica de plataforma equivalente, mesmo se a função específica da plataforma for mais rápida ou mais eficiente. A exceção é quando eficiência no código é o ponto mais importante, nesse caso, é necessário codificar para ambas as situações usando #ifdef. Por exemplo:

```
#ifdef _AIX
     AIX specific code
#else
     generic code
#endif
```

Tipos de filas
Deseja criar uma fila toda vez que precisar de uma ou deseja usar as filas que já foram configuradas? Deseja excluir uma fila quando tiver terminado de usá-la ou ela será usada novamente? Deseja usar as filas de alias para independência do aplicativo? Para ver quais tipos de fila são suportados, consulte Filas

Usando Clusters do Gerenciador de Filas
Talvez você deseje tirar proveito da administração do sistema simplificada e maior disponibilidade, escalabilidade e balanceamento de carga de trabalho que são possíveis ao usar clusters. Consulte Clusters do Gerenciador de Filas para obter mais informações

Tipos de mensagens
Talvez você deseje usar datagramas para mensagens simples, mas mensagens de solicitação (para as quais se espera resposta) para outras situações. Pode ser que deseje designar prioridades diferentes a algumas de suas mensagens. Para obter mais informações sobre como projetar mensagens, consulte “Projetando suas mensagens” na página 94.

Usando sistema de mensagens publicar/assinar ou ponto a ponto
Usando sistema de mensagens publicar/assinar, um aplicativo de envio envia as informações que deseja compartilhar em uma mensagem do IBM WebSphere MQ para um destino padrão gerenciado por publicar/assinar do IBM WebSphere MQ e permite que o IBM WebSphere MQ manipule distribuição dessas informações. O aplicativo de destino não precisa saber nada sobre a origem das informações que recebe, ele apenas registra um interesse em um ou mais tópicos e recebe as informações quando estiverem disponíveis. Para obter mais informações sobre o sistema de mensagens de publicação / assinatura, consulte Introdução ao IBM WebSphere MQ sistema de mensagens de publicação / assinatura

Usando o sistema de mensagens ponto a ponto, um aplicativo de envio envia uma mensagem a uma fila específica, de onde sabe que um aplicativo de recebimento irá recuperá-la. Um aplicativo de recebimento recebe mensagens de uma fila específica e age em seu conteúdo. Um aplicativo frequentemente funcionará como um emissor e um receptor, enviando uma consulta a outro aplicativo e recebendo uma resposta.

Controlando seus programas IBM WebSphere MQ
Você pode desejar iniciar alguns programas automaticamente ou fazer programas esperarem até que uma mensagem específica chegue em uma fila (usando o recurso do IBM WebSphere MQ de *acionamento*, consulte “Iniciando aplicativos IBM WebSphere MQ usando acionadores” na página 333). Como alternativa, você pode desejar iniciar outra instância de um aplicativo quando as mensagens em uma fila não estiverem sendo processadas com rapidez suficiente (usando o recurso IBM WebSphere MQ *eventos de instrumentação* conforme descrito em Eventos de instrumentação).

Executando seu aplicativo em um cliente IBM WebSphere MQ
O MQI completo é suportado no ambiente do cliente e isso permite que quase qualquer aplicativo IBM WebSphere MQ seja vinculado novamente à execução em um cliente MQI do IBM WebSphere MQ . Vincule o aplicativo no cliente MQI do IBM WebSphere MQ à biblioteca MQIC, em vez de à biblioteca MQI.

Nota: Um aplicativo em execução em um cliente IBM WebSphere MQ pode se conectar a mais de um gerenciador de filas simultaneamente ou usar um nome de gerenciador de filas com um asterisco (*) em uma chamada MQCONN ou MQCONNX. Mude o aplicativo se desejar vincular a bibliotecas do gerenciador de filas em vez a bibliotecas do cliente, porque essa função não estará disponível.

Veja “Executando aplicativos no ambiente do cliente MQI do IBM WebSphere MQ” na página 364 para obter mais informações.

Desempenho do aplicativo
As decisões de design podem afetar o desempenho do aplicativo, para obter sugestões para aprimorar o desempenho de aplicativos IBM WebSphere MQ, consulte “Design e desempenho do aplicativo” na página 95 .

Técnicas avançadas do IBM WebSphere MQ
Para aplicativos mais avançados, você pode querer usar algumas técnicas avançadas do IBM WebSphere MQ, como correlacionar respostas e gerar e enviar informações de contexto do IBM WebSphere MQ. Para obter informações adicionais, consulte “Técnicas avançadas do IBM WebSphere MQ” na página 96.

Protegendo dados e mantendo sua integridade
É possível usar as informações de contexto que são passadas com uma mensagem para testar se a mensagem foi enviada a partir de uma origem aceitável. É possível usar recursos de definição de ponto de sincronização fornecidos pelo IBM WebSphere MQ ou seu sistema operacional para assegurar que seus dados permaneçam consistentes com outros recursos (consulte “Confirmando e fazendo backup de unidades de trabalho” na página 327 para obter detalhes adicionais). É possível usar o recurso de *persistência* de mensagens do IBM WebSphere MQ para assegurar a entrega de mensagens importantes.

Testando aplicativos IBM WebSphere MQ

O ambiente de desenvolvimento de aplicativos para programas IBM WebSphere MQ não é diferente daquele de qualquer outro aplicativo, portanto, é possível usar as mesmas ferramentas de desenvolvimento, assim como os recursos de rastreio do IBM WebSphere MQ.

Manipulando exceções e erros
É necessário considerar como processar as mensagens que não podem ser entregues e como resolver situações de erro que são relatadas para você pelo gerenciador de filas. Para alguns relatórios, deve-se configurar opções de relatório em MQPUT.

Conceitos relacionados
Visão geral técnica do IBM WebSphere MQ
“Conceitos de desenvolvimento de aplicativos” na página 8
É possível usar uma opção de linguagens orientadas a objeto ou processuais para gravar aplicativos IBM WebSphere MQ. Use os links neste tópico para obter informações sobre conceitos do IBM WebSphere MQ que são úteis para desenvolvedores de aplicativos.

“Gravando um Aplicativo de Enfileiramento” na página 197
Use essas informações para aprender como escrever aplicativos de enfileiramento, conectar e desconectar de um gerenciador de filas, publicar/assinar e abrir e fechar objetos.

“Gravando aplicativos clientes” na página 356
O que você precisa saber para gravar os aplicativos clientes no WebSphere MQ

“Usando .NET” na página 567
WebSphere MQ classes para .NET permitem que um programa gravado na estrutura de programação .NET se conecte ao WebSphere MQ como um cliente MQI do WebSphere MQ ou se conecte diretamente a um servidor WebSphere MQ .

“Usando C++” na página 636
WebSphere MQ fornece classes C++ equivalentes a objetos WebSphere MQ e algumas classes adicionais equivalentes aos tipos de dados da matriz. Ele fornece inúmeros recursos não disponíveis por meio do MQI.

“Usando classes do WebSphere MQ para JMS” na página 724
WebSphere MQ classes para Java Message Service (WebSphere MQ classes para JMS) é o provedor JMS que é fornecido com o WebSphere MQ. Além de implementar as interfaces definidas no pacote javax.jms , as classes WebSphere MQ para JMS fornecem dois conjuntos de extensões para a API JMS.

“Usando classes do WebSphere MQ para Java” na página 660

As classes do WebSphere MQ para Java permitem usar o WebSphere MQ em um ambiente Java. Um aplicativo Java pode usar classes WebSphere MQ para Java ou classes WebSphere MQ para JMS para acessar recursos do WebSphere MQ .

“Usando o Component Object Model Interface (WebSphere MQ Classes de Automação para ActiveX)” na página 1045
O WebSphere MQ Automation Classes para ActiveX (MQAX) são componentes ActiveX que fornecem classes que podem ser usadas em seu aplicativo para acessar o WebSphere MQ

Projetando suas mensagens

Considere os aspectos fornecidos nestas informações para ajudá-lo a projetar mensagens.

Você cria uma mensagem ao usar uma chamada MQI para colocar a mensagem em uma fila. Como entrada para a chamada, você fornece algumas informações de controle em um *descritor de mensagens* (MQMD) e os dados que você deseja enviar para outro programa. Mas no estágio de design, é necessário considerar o seguinte, pois afetam a maneira que você cria suas mensagens:

Tipo de mensagem a usar
Você está projetando um aplicativo simples no qual é possível enviar uma mensagem e depois não executar nenhuma ação adicional? Ou está solicitando uma resposta a uma pergunta? Se estiver fazendo uma pergunta, você pode incluir no descritor de mensagens o nome da fila na qual deseja receber a resposta.

Deseja que suas mensagens de solicitação e de resposta sejam síncronas? Isso sugere que você configura um período de tempo limite para a resposta à sua solicitação e, se não receber a resposta dentro desse período, isso será tratado como um erro.

Ou preferiria trabalhar de forma assíncrona, para que seus processos não precisem depender da ocorrência de eventos específicos, como sinais de sincronização comuns?

Outra consideração é se você tem todas as suas mensagens dentro de uma unidade de trabalho.

Atribuindo prioridades diferentes a mensagens
É possível designar um valor de prioridade a cada mensagem e definir a fila de forma que ela mantenha suas mensagens em ordem de prioridade. Se fizer isto, quando outro programa recuperar uma mensagem da fila, sempre obterá a mensagem com a prioridade mais alta. Se a fila não mantiver suas mensagens em ordem de prioridade, um programa que recupera mensagens da fila irá recuperá-las na ordem em que foram incluídas na fila.

Programas também podem selecionar uma mensagem usando o identificador que o gerenciador de filas designou quando a mensagem foi colocada na fila. Como alternativa, é possível gerar seus próprios identificadores para cada uma de suas mensagens.

O efeito de reiniciar o gerenciador de filas nas mensagens
O gerenciador de filas preserva todas as mensagens persistentes, recuperando-as quando necessário dos arquivos de log do WebSphere MQ , quando ele é reiniciado. Mensagens não persistentes e filas dinâmicas temporárias não são preservadas. Quaisquer mensagens que não deseje descartar devem ser definidas como persistentes quando forem criadas. Ao gravar um aplicativo para o WebSphere MQ para Windows ou WebSphere MQ em sistemas UNIX and Linux , certifique-se de saber como seu sistema foi configurado em relação à alocação de arquivo de log para reduzir o risco de projetar um aplicativo que será executado para os limites do arquivo de log.

Fornecendo informações sobre si mesmo ao destinatário de mensagens
Geralmente, o gerenciador de filas configura o ID do usuário, mas os aplicativos devidamente autorizados também podem configurar esse campo, para que seja possível incluir seu próprio ID do usuário e outras informações que o programa de recebimento pode usar para propósitos de contabilidade ou segurança.

Quantia de filas de recebimento
Se uma mensagem precisar ser colocada em várias filas, será possível usar uma lista de distribuiçãoou publicar em um tópico.

Design e desempenho do aplicativo

Há várias maneiras em que um design ruim de programa pode afetar o desempenho. Elas podem ser difíceis de detectar porque o programa pode parecer executar bem, mas afetar o desempenho de outras tarefas. Vários problemas específicos de programas que fazem chamadas do WebSphere MQ são explicados neste tópico

Aqui estão algumas ideias para ajudar a projetar os aplicativos eficientes:

- Projete seu aplicativo de forma que o processamento ocorra em paralelo ao tempo de reflexão do usuário:
  - Exiba um painel e permita que o usuário comece a digitar enquanto o aplicativo ainda está inicializando.
  - Obtenha os dados que precisa em paralelo a partir de diferentes servidores.
- Mantenha conexões e filas abertas se for reutilizá-las em vez de repetidamente abrir e fechar, conectar e desconectar.
- No entanto, um aplicativo do servidor que está efetuando put somente de uma mensagem deve usar MQPUT1.
- Os gerenciadores de filas são otimizados para mensagens com tamanho entre 4 KB e 100 KB. Mensagens muito grandes são ineficientes; é provavelmente melhor enviar 100 mensagens de 1 MB cada do que uma única mensagem de 100 MB. Mensagens muito pequenas também são ineficientes. O gerenciador de filas executa a mesma quantia de trabalho para uma mensagem de byte único que para uma mensagem de 4 KB.
- Mantenha suas mensagens em uma unidade de trabalho, para que possam ser confirmadas ou restauradas simultaneamente.
- Use a opção não persistente para mensagens que não precisam ser recuperáveis.
- Se precisar enviar uma mensagem para diversas filas de destino, considere usar uma lista de distribuição.

Efeito do comprimento da mensagem

A quantia de dados em uma mensagem pode afetar o desempenho do aplicativo que processa a mensagem. Para obter o melhor desempenho de seu aplicativo, envie somente os dados essenciais em uma mensagem. Por exemplo, em uma solicitação para debitar uma conta bancária, as únicas informações que podem precisar ser passadas do cliente para o aplicativo do servidor são o número da conta e o valor do débito.

Efeito de persistência de mensagem

Mensagens persistentes são geralmente registradas. Registrar mensagens reduz o desempenho de seu aplicativo, portanto, use mensagens persistentes somente para dados essenciais. Se os dados de uma mensagem puderem ser descartados se o gerenciador de filas parar ou falhar, use uma mensagem não persistente.

Procurando uma mensagem específica

A chamada MQGET, geralmente, recupera a primeira mensagem de uma fila. Se você usar a mensagem e os identificadores de correlação (*MsgId* e *CorrelId*) no descritor de mensagens para especificar uma mensagem específica, o gerenciador de filas precisa procurar na fila até localizar essa mensagem. Usar a chamada MQGET dessa forma afeta o desempenho de seu aplicativo.

Filas que contêm mensagens de comprimentos diferentes

Se seu aplicativo não puder usar mensagens de um comprimento fixo, aumente e diminua os buffers dinamicamente para ajustar o tamanho típico de mensagem. Se o aplicativo emitir uma chamada MQGET que falha porque o buffer é muito pequeno, o tamanho dos dados da mensagem é retornado. Inclua

código em seu aplicativo para que o buffer seja redimensionado apropriadamente e a chamada MQGET emitida novamente.

Nota: se não configurar o atributo *MaxMsgLength* explicitamente, ele usa como padrão 4 MB, o que pode ser muito ineficiente se for usado para influenciar o tamanho do buffer do aplicativo.

Frequência de pontos de sincronização

Os programas que emitem um número muito grande de chamadas MQPUT ou MQGET no ponto de sincronização, sem confirmá-las, podem causar problemas de desempenho. As filas afetadas podem se ficar cheias de mensagens atualmente inacessíveis, enquanto que outras tarefas podem estar esperando para obter essas mensagens. Isso tem implicações em termos de armazenamento e em termos de encadeamentos ligados a tarefas que estão tentando obter mensagens.

Uso da chamada MQPUT1

Use a chamada MQPUT1 somente se tiver uma única mensagem para colocar em uma fila. Se desejar colocar mais de uma mensagem, use a chamada MQOPEN, seguida por uma série de chamadas MQPUT e uma única chamada MQCLOSE.

Número de encadeamentos em uso

Para o WebSphere MQ para Windows, um aplicativo pode requerer um grande número de encadeamentos Cada processo do gerenciador de filas tem alocado um número máximo permitido de encadeamentos do aplicativo.

Os aplicativos podem usar encadeamentos em excesso. Considere se o aplicativo leva em consideração essa possibilidade e se executa ações para parar ou relatar esse tipo de ocorrência.

Técnicas avançadas do IBM WebSphere MQ

Para um aplicativo IBM WebSphere MQ simples, é necessário decidir quais objetos do WebSphere MQ usar em seu aplicativo e quais tipos de mensagens você deseja usar Para um aplicativo mais avançado, pode desejar usar algumas das técnicas introduzidas nas seções a seguir.

Esperando mensagens

Um programa que está atendendo a uma fila pode esperar mensagens da seguinte forma:

- Esperando até uma mensagem chegar ou um intervalo de tempo especificado expirar (consulte “Esperando mensagens” na página 271).
- Estabelecendo uma saída de retorno de chamada para ser acionada quando uma mensagem chegar; consulte “Consumo assíncrono de mensagens do IBM WebSphere MQ” na página 34.
- Fazendo chamadas periódicas na fila para ver se uma mensagem chegou (*pesquisa*). Isso geralmente não é aconselhável pois pode ter implicações no desempenho.

Correlacionando respostas

Nos aplicativos WebSphere MQ , quando um programa recebe uma mensagem que solicita que ele execute algum trabalho, o programa geralmente envia uma ou mais mensagens de resposta para o solicitante

Para ajudar o solicitante a associar essas respostas à sua solicitação original, um aplicativo pode configurar um campo *correlation identifier* no descritor de cada mensagem. Os programas copiam, então, o identificador de mensagem da mensagem de solicitação para o campo do identificador de correlação de suas mensagens de resposta.

Configurando e usando informações de contexto

Informações de contexto são usadas para associar mensagens ao usuário que as gerou e para identificar o aplicativo que gerou a mensagem. Essas informações são úteis para segurança, contabilidade, auditoria e determinação de problemas.

Ao criar uma mensagem, é possível especificar uma opção que solicita que o gerenciador de filas associa informações de contexto padrão à sua mensagem.

Para obter mais informações sobre como usar e configurar informações de contexto, consulte "Contexto da mensagem" na página 39.

Iniciando programas do WebSphere MQ automaticamente

Use o WebSphere MQ *acionamento* para iniciar um programa automaticamente quando as mensagens chegarem em uma fila

É possível configurar condições de acionamento em uma fila para que um programa comece a processar essa fila:

- Toda vez que uma mensagem chegar na fila
- Quando a primeira mensagem chegar na fila
- Quando o número de mensagens na fila atingir um número predefinido

Para obter mais informações sobre acionamento, consulte "Iniciando aplicativos IBM WebSphere MQ usando acionadores" na página 333. O acionamento é apenas uma da maneira de iniciar um programa automaticamente. Por exemplo, é possível iniciar um programa automaticamente em um cronômetro usando recursos nãoWebSphere MQ .

WebSphere MQ pode definir objetos de serviço para iniciar programas do WebSphere MQ quando o gerenciador de filas for iniciado; consulte Objetos de serviço.

Gerando relatórios do WebSphere MQ

É possível solicitar relatórios a seguir em um aplicativo:

- Relatórios de exceções
- Relatórios de expiração
- Relatórios de confirmação de chegada (COA)
- Relatórios de confirmação de entrega (COD)
- Relatórios de positive action notification (PAN)
- Relatórios de negative action notification (NAN)

Eles são descritos na seção "Mensagens de relatório" na página 11.

Clusters e afinidades de mensagens

Antes de começar a usar clusters com diversas definições para a mesma fila, examine seus aplicativos para ver se há algum que requeira uma troca de mensagens relacionadas.

Em um cluster, uma mensagem pode ser roteada para qualquer gerenciador de filas que hospede uma instância da fila apropriada. Portanto, a lógica dos aplicativos com afinidades de mensagens pode ser afetada.

Por exemplo, você pode ter dois aplicativos que dependem de uma série de mensagens que fluem entre eles na forma de perguntas e respostas. Pode ser importante que todas as perguntas sejam enviadas ao mesmo gerenciador de filas e que todas as respostas sejam enviadas de volta ao outro gerenciador de filas. Nessa situação, é importante que a rotina de gerenciamento de carga de trabalho não envie as mensagens para qualquer gerenciador de filas que simplesmente hospeda uma instância da fila adequada.

Sempre que possível, remova as afinidades. Remover as afinidades de mensagens melhora a disponibilidade e escalabilidade de aplicativos.

Para obter mais informações, consulte Manipulando afinidades de mensagens ..

Programas de amostra do WebSphere MQ

Use esta coleção de tópicos para aprender sobre programas de amostra do WebSphere MQ em diferentes plataformas

- “Programas de amostra para plataformas distribuídas” na página 98

Conceitos relacionados

“Conceitos de desenvolvimento de aplicativos” na página 8
É possível usar uma opção de linguagens orientadas a objeto ou processuais para gravar aplicativos IBM WebSphere MQ. Use os links neste tópico para obter informações sobre conceitos do IBM WebSphere MQ que são úteis para desenvolvedores de aplicativos.

“Decidindo qual linguagem de programação usar” na página 80
Use estas informações para descobrir sobre linguagens de programação e estruturas suportadas pelo IBM WebSphere MQe algumas considerações para usá-las.

“Projetando aplicativos IBM WebSphere MQ” na página 91
Quando você tiver decidido como seus aplicativos podem aproveitar as plataformas e ambientes disponíveis para você, será necessário decidir como usar os recursos oferecidos pelo WebSphere MQ.

“Gravando um Aplicativo de Enfileiramento” na página 197
Use essas informações para aprender como escrever aplicativos de enfileiramento, conectar e desconectar de um gerenciador de filas, publicar/assinar e abrir e fechar objetos.

“Gravando aplicativos clientes” na página 356
O que você precisa saber para gravar os aplicativos clientes no WebSphere MQ

“Usando serviços da Web no WebSphere MQ” na página 958
É possível desenvolver aplicativos IBM WebSphere MQ para serviços da web usando o transporte IBM WebSphere MQ para SOAP ou a ponte IBM WebSphere MQ para HTTP (Protocolo de Transporte de Hipertexto)

“Escrevendo aplicativos de publicar/assinar” na página 281
Inicie a gravação de aplicativos de publicação / assinatura do WebSphere MQ

“Construindo um aplicativo IBM WebSphere MQ” na página 435
Use estas informações para saber como construir um aplicativo IBM WebSphere MQ em diferentes plataformas.

“Manipulando erros do programa” na página 555
Estas informações explicarão os erros associados às chamadas MQI de aplicativos, ao realizar uma chamada ou quando sua mensagem for entregue para seu destino final.

Programas de amostra para plataformas distribuídas

Este tópico descreve os programas de amostra entregues com o IBM WebSphere MQ, gravados em C e COBOL As amostras demonstram usos típicos do Message Queue Interface (MQI).

As amostras não são destinados a demonstrar técnicas de programação geral, portanto, alguma verificação de erro que você queira incluir em um programa de produção será omitida. No entanto, essas amostras são adequadas para uso como uma base para seus próprios programas de fila de mensagens.

O código de origem para todas as amostras é fornecido com o produto; esta origem inclui comentários que explicam as técnicas de enfileiramento de mensagens demonstradas nos programas.

Programas de amostra C++: Consulte Using C++ para obter uma descrição dos programas de amostra disponíveis em C + +.

Os nomes das amostras de iniciam com o prefixo `amq`. O quarto caractere indica a linguagem de programação e o compilador, quando necessário.

s	Linguagem C
0	idioma COBOL nos compiladores IBM e Micro Focus
i	Linguagem COBOL somente em compiladores IBM
m	Linguagem COBOL somente em compiladores Micro Focus

O oitavo caractere do executável indica se a amostra é executada no modo cliente ou no modo de ligação local. Se não houver um oitavo caractere, então, a amostra será executada no modo de ligações locais. Se o oitavo caractere for 'c', então, a amostra será executada no modo cliente. Para configurar o gerenciador de filas para aceitar conexões do cliente, consulte “Preparando e executando os programas de amostra” na página 111 para obter detalhes.

Use os seguintes links para descobrir mais sobre os programas de amostra:

- “Recursos demonstrados nos programas de amostra” na página 99
- “Os programas de amostra Publish/Subscribe” na página 137
- “Os programas de amostra Put” na página 142
- “O programa de amostra Distribution List” na página 129
- “Os programas de amostra Browse” na página 118
- “O programa de amostra Browser” na página 119
- “Os programas de amostra Get” na página 131
- “Os programas de amostra Reference Message” na página 143
- “Os programas de amostra Request” na página 149
- “Os programas de amostra Inquire” na página 136
- “O programa de amostra Inquire Properties of a Message Handle” na página 137
- “Os programas de amostra Set” na página 153
- “Os programas de amostra Echo” na página 130
- “O programa de amostra Data-Conversion” na página 121
- “Os programas de amostra Triggering” na página 157
- “O programa de amostra Asynchronous Put” na página 117
- “Amostras de coordenação de banco de dados” na página 122
- “A amostra de transação CICS” na página 120
- “Amostras do TUXEDO” na página 158
- “Amostra do manipulador de filas de mensagens não entregues” na página 129
- “O programa de amostra Connect” na página 120
- “O programa de amostra de saída de API” na página 115
- “Usando a saída de segurança SSPI em sistemas Windows” na página 171
- “Executando as amostras usando filas remotas” na página 172
- “O programa de amostra de Cluster Queue Monitoring (AMQSCLM)” na página 172
- “Programa de amostra para Connection Endpoint Lookup (CEPL)” na página 182

Recursos demonstrados nos programas de amostra

Uma coleção de tabelas que mostram as técnicas demonstradas pelo WebSphere MQ programas de amostra.

Todas as amostras abrem e fecham filas usando as chamadas MQOPEN e MQCLOSE, portanto, essas técnicas não estão listados separadamente nas tabelas. Veja o título que inclui a plataforma na qual você está interessado.

Amostras para sistemas UNIX and Linux

Este tópico mostra as técnicas demonstradas pelos programas de amostra para os sistemas WebSphere MQ no UNIX and Linux

Consulte “Preparando e executando programas de amostra em sistemas UNIX” na página 113 para descobrir onde os programas de amostra para WebSphere MQ em sistemas UNIX e Linux estão armazenados..

Tabela 14 na página 100 A tabela lista quais arquivos de origem C e COBOL são fornecidos e se um executável do servidor ou cliente é incluído.

Tabela 14. WebSphere MQ em UNIX and Linux programas de amostra que demonstram o uso do MQI (C e COBOL)

Técnica	**C (origem) (“1” na página 102)**	**COBOL (origem) (“2” na página 102)**	**Servidor (executável C)**	**Cliente (executável C) (“3” na página 102)**
Usando a interface de publicação/assinatura	amqssbxa amqssuba amqspuba	sem amostra	amqssbx amqssub amqspub	sem amostra
Colocando mensagens usando a chamada MQPUT	amqsput0	amq0put0	amqsput	amqsputc
Colocando uma única mensagem usando a chamada MQPUT1	amqsecha amqsinqa	amqiechx amqiinqx amqmechx amqminqx	amqsech amqsinq	amqsechc
Colocando mensagens em uma lista de distribuição (“4” na página 102)	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Respondendo a uma mensagem de solicitação	amqsinqa	amqiinqx amqminqx	amqsinq	sem amostra
Obtendo mensagens (nenhuma espera)	amqsgbr0	amq0gbr0	amqsgbr	sem amostra
Obtendo mensagens (aguardar com um limite de tempo)	amqsget0	amq0get0	amqsget	amqsgetc
Obtendo mensagens (espera ilimitada)	amqstrg0	sem amostra	amqstrg	amqstrgc
Obtendo mensagens (com conversão de dados)	amqsecha	sem amostra	amqsech	sem amostra
Colocando mensagens de referência em uma fila (“4” na página 102)	amqsprma	sem amostra	amqsprm	amqsprmc
Obtendo mensagens de referência de uma fila (“4” na página 102)	amqsgrma	sem amostra	amqsgrm	amqsgrmc
Saída do canal de mensagem de referência (“4” na página 102)	amqsxrma amqsqrma	sem amostra	amqsxrm	sem amostra
Procurando 20 primeiros caracteres de uma mensagem	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Procurando mensagens completas	amqsbcg0	sem amostra	amqsbcg	amqsbcgc
Usando uma fila de entrada compartilhada	amqsinqa	amqiinqx amqminqx	amqsinq	amqsinqc
Usando uma fila de entrada exclusiva	amqstrg0	amq0req0	amqstrg	amqstrgc

Tabela 14. WebSphere MQ em UNIX and Linux programas de amostra que demonstram o uso do MQI (C e COBOL) (continuação)

Técnica	**C (origem) ("1" na página 102)**	**COBOL (origem) ("2" na página 102)**	**Servidor (executável C)**	**Cliente (executável C) ("3" na página 102)**
Usando a chamada MQINQ	amqsinqa	amqiinqx amqminqx	amqsinq	sem amostra
Usando a chamada MQSET	amqsseta	amqmsetx amqisetx	amqsset	amqssetc
Usando uma fila de resposta	amqsreq0	amq0req0	amqsreq	amqsreqc
Solicitando exceções de mensagens	amqsreq0	amq0req0	amqsreq	sem amostra
Aceitando uma mensagem truncada	amqsgbr0	amq0gbr0	amqsgbr	sem amostra
Usando um nome de fila resolvido	amqsgbr0	amq0gbr0	amqsgbr	sem amostra
Acionando um processo	amqstrg0	sem amostra	amqstrg	amqstrgc
Usando a conversão de dados	("5" na página 102)	sem amostra	sem amostra	sem amostra
WebSphere MQ (coordenando gerenciadores de banco de dados compatíveis com XA) acessando um único banco de dados usando SQL	amqsxas0.sqc DB2 amqsxas0.ec Informix	amq0xas0.sqb	sem amostra	sem amostra
WebSphere MQ (coordenando gerenciadores de banco de dados compatíveis com XA) acessando dois bancos de dados usando SQL	amqsxag0.c amqsxab0.sqc amqsxaf0.sqc	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	sem amostra	sem amostra
Transação CICS ("6" na página 102)	amqscic0.ccs	sem amostra	amqscic0	sem amostra
Transação Encina ("4" na página 102)	amqsxae0	sem amostra	amqsxae0	sem amostra
Transação TUXEDO para colocar mensagens ("7" na página 102)	amqstxpx	sem amostra	sem amostra	sem amostra
Transação TUXEDO para obter mensagens ("7" na página 102)	amqstxgx	sem amostra	sem amostra	sem amostra
Servidor para TUXEDO ("7" na página 102)	amqstxsx	sem amostra	sem amostra	sem amostra
Manipulador da fila de devoluções	Diretório `./tools/c/Samples/dlq` ("8" na página 102)	sem amostra	amqsdlq	sem amostra
A partir de um cliente MQI, a colocação de uma mensagem	sem amostra	sem amostra	sem amostra	amqsputc
A partir de um cliente MQI, obter uma mensagem	sem amostra	sem amostra	sem amostra	amqsgetc
Conectando-se ao gerenciador de filas usando MQCONNX	amqscnxc	sem amostra	sem amostra	amqscnxc
Usando saídas de API	amqsaxe0	sem amostra	amqsaxe	sem amostra

Tabela 14. WebSphere MQ em UNIX and Linux programas de amostra que demonstram o uso do MQI (C e COBOL) (continuação)

Técnica	C (origem) (“1” na página 102)	COBOL (origem) (“2” na página 102)	Servidor (executável C)	Cliente (executável C) (“3” na página 102)
Saída de balanceamento de carga de trabalho do cluster	amqswlm0	sem amostra	amqswlm	sem amostra
Colocando mensagens de maneira assíncrona e obtendo o status usando a chamada MQSTAT	amqsapt0	sem amostra	amqsapt	amqsaptc
Clientes reconectáveis	amqsphac amqsghac amqsmhac	sem amostra	não aplicável	amqsphac amqsghac amqsmhac
Utilizando os consumidores de mensagens para consumir mensagens de várias filas de maneira assíncrona	amqscbf0	sem amostra	amqscbf	amqscbfc
Especificando informações de conexão SSL/TLS no MQCONNX	amqssslc	sem amostra	não aplicável	amqssslc

Notes:

1. A versão executável das amostras do cliente MQI do WebSphere MQ compartilham a mesma origem que as amostras executadas em um ambiente do servidor.
2. Compile programas iniciando 'amqm' com o compilador COBOL Micro Focus, aqueles iniciando 'amqi' com o compilador COBOL IBM e aqueles iniciando 'amq0' com um deles.
3. As versões executáveis das amostras do cliente MQI do WebSphere MQ não estão disponíveis no WebSphere MQ para o HP-UX
4. Suportado no WebSphere MQ para AIX, WebSphere MQ para HP-UXe WebSphere MQ apenas para Solaris
5. No WebSphere MQ para AIX, WebSphere MQ para HP-UXe WebSphere MQ para Solaris, esse programa é chamado `amqsvfc0.c`
6. O CICS é suportado pelo WebSphere MQ para AIX e pelo WebSphere MQ para HP-UX apenas
7. TUXEDO não é suportado pelo WebSphere MQ para Linux no System p.
8. A origem para o manipulador da fila de mensagens não entregues consiste em vários arquivos e é fornecida em um diretório separado.

Informações detalhadas sobre o suporte para sistemas UNIX and Linux estão disponíveis na página de requisitos de sistemas do WebSphere MQ em Requisitos do Sistema para IBM WebSphere MQ.

Amostras para o cliente IBM WebSphere MQ para HP Integrity NonStop Server

Este tópico mostra as técnicas demonstradas pelos programas de amostra para o cliente IBM WebSphere MQ em sistemas HP Integrity NonStop Server.

Tabela 15 na página 103A tabela lista os programas de origem C, COBOL e pTAL que são fornecidos.

Tabela 15. IBM WebSphere MQ em programas de amostra do HP Integrity NonStop Server demonstrando o uso de C, COBOL e pTAL

Técnica	**C**				**COBOL**		**pTAL**	
	OSS (Origem)	OSS (Executável)	Guardian (Origem)	Guardian (Executável)	OSS (Origem)	Guardian (Origem)	OSS (Origem)	Guardian (Origem)
Usando a interface de publicação/ assinatura	amqspuba.c amqssbxa.c amqssuba.c amqspse0.c	amqspubc amqssbxc amqssubc	MQSPUBC MQSSBXC MQSSUBC	AMQSPUBC AMQSSBXC AMQSSUBC	amq0pub0.cbl amq0sub0.cbl	MQSPUBL MQSSUBL	amqtpub0.tal amqtsub0.tal	MQSPUBT MQSSUBT
Colocando mensagens usando a chamada MQPUT	amqsput0.c	amqsputc	MQSPUTC	AMQSPUTC	amq0put0.cbl	MQSPUTL	amqtput0.tal	MQSPUTT
Colocando uma única mensagem usando a chamada MQPUT1	amqsecha.c	amqsechc	MQSECHC	AMQSECHC			amqtech0.tal	MQSECHT
Colocando mensagens em uma lista de distribuição	amqsptl0.c	amqsptlc	MQSPTLC	AMQSPTLC	amq0ptl0.cbl	MQSPTLL		
Respondendo a uma mensagem de solicitação	amqsinqa.c	amqsinqc	MQSINQC	AMQSINQC				
Obtendo mensagens (nenhuma espera)	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		

Tabela 15. IBM WebSphere MQ em programas de amostra do HP Integrity NonStop Server demonstrando o uso de C, COBOL e pTAL (continuação)

Técnica	**C**				**COBOL**		**pTAL**	
Obtendo mensagens (aguardar com um limite de tempo)	amqsget0.c	amqsgetc	MQSGETC	AMQSGETC	amq0get0.cbl	MQSGETL	amqtget0.tal	MQSGETT
Obtendo mensagens (espera ilimitada)	amqstrg0.c	amqstrgc	MQSTRGC	AMQSTRGC				
Obtendo mensagens (com conversão de dados)	amqsecha.c	amqsechc	MQSECHC	AMQSECHC				
Colocando mensagens de referência em uma fila	amqsprma.c	amqsprmc	MQSPRMC	AMQSPRMC				
Obtendo Mensagens de Referência de uma fila	amqsgrma.c	amqsgrmc	MQSGRMC	AMQSGRMC				
Saída do canal de mensagem de referência	amqsqrma.c amqsxrma.c		MQSQRMC MQSXRMC					
Procurando 20 primeiros caracteres de uma mensagem	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		

Tabela 15. IBM WebSphere MQ em programas de amostra do HP Integrity NonStop Server demonstrando o uso de C, COBOL e pTAL (continuação)

Técnica	**C**				**COBOL**		**pTAL**	
Procurando mensagens completas	amqsbcg0.c	amqsbcgc	MQSBCGC	AMQSBCGC				
Usando uma fila de entrada compartilhada	amqsinqa.c	amqsinqc	MQSINQC	MQSINQC				
Usando uma fila de entrada exclusiva	amqstrg0.c	amqstrgc	MQSTRGC	AMQSTRGC				
Usando a chamada MQINQ	amqsinqa.c	amqsinqc	MQSINQC	AMQSINQC				
Usando a chamada MQSET	amqsseta.c	amqssetc	MQSSETC	AMQSSETC				
Usando uma fila de resposta	amqsreq0.c	amqsreqc	MQSREQC	AMQSREQC	amq0req0.cbl	MQSREQL		
Solicitando exceções de mensagens	amqsreq0.c	amqsreqc	MQSREQC	AMQSREQC	amq0req0.cbl	MQSREQL		
Aceitando uma mensagem truncada	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		
Usando um nome de fila resolvido	amqsgbr0.c	amqsgbrc	MQSGBRC	AMQSGBRC	amq0gbr0.cbl	MQSGBRL		
Acionando um processo	amqstrg0.c	amqstrgc	MQSTRGC	AMQSTRGC				

Tabela 15. IBM WebSphere MQ em programas de amostra do HP Integrity NonStop Server demonstrando o uso de C, COBOL e pTAL (continuação)

Técnica	**C**				**COBOL**		**pTAL**	
Usando a conversão de dados	amqsvfc0.c							
Manipulador de fila de devoluções (1)	Directory ./samp/dlq							
Conectando-se a um gerenciador de filas usando MQCONNX	amqscnxc.c	amqscnxc	MQSCNXC					
Usando saídas de API	amqsaxe0.c amqsaem0.c							
Saída de balanceamento de carga de trabalho do cluster	amqswlm0.c		MQSWLMC					
Monitor de fila de clusters	amqsclma.c							
Colocando mensagens de maneira assíncrona e obtendo o status usando a chamada MQSTAT	amqsapt0.c	amqsaptc	MQSAPTC	MQSAPTC				

Tabela 15. IBM WebSphere MQ em programas de amostra do HP Integrity NonStop Server demonstrando o uso de C, COBOL e pTAL (continuação)

Técnica	**C**				**COBOL**		**pTAL**	
Clientes reconectáveis	amqsghac.c amqsmhac.c amqsphac.c	amqsghac amqsmhac amqsphac	MQSGHAC MQSMHAC MQSPHAC MQSFHAC	AMQSGHAC AMQSMHAC AMQSPHAC AMQSFHAC				
Usando os consumidores de mensagens para consumir mensagens de forma assíncrona de várias filas	amqscbf0.c	amqscbfc						
Especificando informações de conexão SSL/TLS no MQCONNX	amqssslc.c	amqssslc	MQSSSLC	AMQSSSLC				
Rastreio de atividade	amqsact0.c	amqsactc	MQSACTC	AMQSACTC				
Propriedades da Mensagem	amqsiqma.c amqsstma.c	amqsiqmc amqsstmc	MQSIQMC MQSSTMC	AMQSIQMC AMQSSTMC				
Servidor de Comandos	amqsstop.c		MQSSTOC					
Eventos de log	amqslog0.c	amqslogc	MQSLOGC	AMQSLOGC				
Contabilidade	amqsmon0.c	amqsmonc	MQSMONC	AMQSMONC				

Tabela 15. IBM WebSphere MQ em programas de amostra do HP Integrity NonStop Server demonstrando o uso de C, COBOL e pTAL (continuação)

Técnica	C				COBOL		pTAL	
Interface de administração	amqsaicq.c amqsaiem.c amqsailq.c							
Um exemplo de uma função principal de linguagem C para chamada de pTAL			MQSPTMC					

Notes:

1. A origem para o manipulador da fila de mensagens não entregues consiste em vários arquivos e é fornecida em um diretório separado.
2. Para obter informações sobre o desenvolvimento de aplicativos para o seu cliente de IBM WebSphere MQ na HP Integrity NonStop Server plataforma, consulte:
   - "Construindo seu aplicativo no HP Integrity NonStop Server" na página 441
     - "Preparando programas C no HP Integrity NonStop Server" na página 443
     - "Preparando programas COBOL" na página 444
     - "Preparando programas pTAL" na página 446

Amostras para IBM WebSphere MQ para Windows

Este tópico mostra as técnicas demonstradas pelos programas de amostra para o IBM WebSphere MQ for Windows

Tabela 16 na página 108 A tabela lista quais arquivos de origem C e COBOL são fornecidos e se um executável do servidor ou cliente é incluído.

Tabela 16. IBM WebSphere MQ para Windows programas de amostra demonstrando o uso do MQI (C e COBOL)

Técnica	C (origem)	COBOL (origem)	Servidor (executável C)	Cliente (executável C)
Usando a interface de publicação/assinatura	amqssbxa amqssuba amqspuba	sem amostra	amqssbx amqssub amqspub	sem amostra
Colocando mensagens usando a chamada MQPUT	amqsput0	amq0put0	amqsput	amqsputc
Colocando uma única mensagem usando a chamada MQPUT1	amqsecha amqsinqa	amqminq2 amqmech2 amqiinq2 amqiech2	amqsech amqsinq	amqsechc amqsinqc

Tabela 16. IBM WebSphere MQ para Windows programas de amostra demonstrando o uso do MQI (C e COBOL) (continuação)

Técnica	**C (origem)**	**COBOL (origem)**	**Servidor (executável C)**	**Cliente (executável C)**
Colocando mensagens em uma lista de distribuição	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Respondendo a uma mensagem de solicitação	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Obtendo mensagens (nenhuma espera)	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Obtendo mensagens (aguardar com um limite de tempo)	amqsget0	amq0get0	amqsget	amqsgetc
Obtendo mensagens (espera ilimitada)	amqstrg0	sem amostra	amqstrg	amqstrgc
Obtendo mensagens (com conversão de dados)	amqsecha	sem amostra	amqsech	amqsechc
Colocando mensagens de referência em uma fila	amqsprma	sem amostra	amqsprm	amqsprmc
Obtendo Mensagens de Referência de uma fila	amqsgrma	sem amostra	amqsgrm	amqsgrmc
Saída do canal de mensagem de referência	amqsxrma amqsqrma	sem amostra	amqsxrm	sem amostra
Procurando 20 primeiros caracteres de uma mensagem	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Procurando mensagens completas	amqsbcg0	sem amostra	amqsbcg	amqsbcgc
Usando uma fila de entrada compartilhada	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Usando uma fila de entrada exclusiva	amqstrg0	amq0req0	amqstrg	amqstrgc
Usando a chamada MQINQ	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Usando a chamada MQSET	amqsseta	amqmset2 amqiset2	amqsset	amqssetc
Usando a chamada MQINQMP	amqsiqma	sem amostra	sem amostra	sem amostra
Usando uma fila de resposta	amqsreq0	amq0req0	amqsreq	amqsreqc
Solicitando exceções de mensagens	amqsreq0	amq0req0	amqsreq	amqsreqc
Aceitando uma mensagem truncada	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Usando um nome de fila resolvido	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Acionando um processo	amqstrg0	sem amostra	amqstrg	amqstrgc
Usando a conversão de dados	amqsvfc0	sem amostra	sem amostra	sem amostra
WebSphere MQ (coordenando gerenciadores de banco de dados compatíveis com XA) acessando um único banco de dados usando SQL	amqsxas0.sqc DB2 amqsxas0.ec Informix	amq0xas0.sqb	sem amostra	sem amostra

Tabela 16. IBM WebSphere MQ para Windows programas de amostra demonstrando o uso do MQI (C e COBOL) (continuação)

Técnica	**C (origem)**	**COBOL (origem)**	**Servidor (executável C)**	**Cliente (executável C)**
WebSphere MQ (coordenando gerenciadores de banco de dados compatíveis com XA) acessando dois bancos de dados usando SQL	amqsxag0.c amqsxab0.sqc DB2 amqsxaf0.sqc DB2	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	sem amostra	sem amostra
TUXEDO transação para colocar mensagens	amqstxpx	sem amostra	sem amostra	sem amostra
TUXEDO para obter mensagens da transação	amqstxgx	sem amostra	sem amostra	sem amostra
o Servidor para TUXEDO	amqstxsx	sem amostra	sem amostra	sem amostra
Manipulador da fila de devoluções	Diretório `./tools/c/Samples/dlq` (“1” na página 110)	sem amostra	amqsdlq	sem amostra
Em um cliente MQI do WebSphere MQ , colocando uma mensagem	sem amostra	sem amostra	sem amostra	amqsputc
Em um cliente MQI do WebSphere MQ , obtendo uma mensagem	sem amostra	sem amostra	sem amostra	amqsgetc
Conectando-se ao gerenciador de filas usando MQCONNX	amqscnxc	sem amostra	sem amostra	amqscnxc
Usando saídas de API	amqsaxe0	sem amostra	amqsaxe	sem amostra
Balanceamento de carga de trabalho do cluster	amqswlm0	sem amostra	amqswlm	sem amostra
rotinas de segurança SSPI	amqsspin	sem amostra	amqrspin.dll	amqrspin.dll
Colocando mensagens de maneira assíncrona e obtendo o status usando a chamada MQSTAT	amqsapt0	sem amostra	amqsapt	amqsaptc
Clientes reconectáveis	amqsphac amqsghac amqsmhac	sem amostra	Não-aplicável	amqsphac amqsghac amqsmhac
Utilizando os consumidores de mensagens para consumir mensagens de várias filas de maneira assíncrona	amqscbf0	sem amostra	amqscbf	amqscbfc
Especificando informações de conexão SSL/TLS no MQCONNX	amqssslc	sem amostra	não aplicável	amqssslc

Notes:

1. A origem para o manipulador da fila de mensagens não entregues consiste em vários arquivos e é fornecida em um diretório separado.

Amostras do Visual Basic para IBM WebSphere MQ para Windows

Este tópico mostra as técnicas demonstradas por programas de amostra do Visual Basic para IBM WebSphere MQ para Windows.

Tabela 17 na página 111 mostra as técnicas demonstradas pelos programas de amostra IBM WebSphere MQ para Windows .

Um projeto pode conter vários arquivos. Quando você abre um projeto no Visual Basic, os outros arquivos serão carregados automaticamente. Nenhum programa executável é fornecido.

Todos os projetos de amostra, exceto mqtrivc.vbp, são configurados para funcionarem com o servidor IBM WebSphere MQ. Para descobrir como mudar os projetos de amostra para funcionarem com os clientes IBM WebSphere MQ, consulte “Preparando programas do Visual Basic no Windows” na página 469.

Tabela 17. IBM WebSphere MQ para Windows programas de amostra que demonstram o uso do MQI (Visual Basic)

Técnica	**Nome do arquivo do projeto**
Colocando mensagens usando a chamada MQPUT	amqsputb.vbp
Obtendo mensagens usando a chamada MQGET	amqsgetb.vbp
Procurando em uma fila usando a chamada MQGET	amqsbcgb.vbp
Amostras de MQGET e MQPUT simples (cliente)	mqtrivc.vbp
Amostras de MQGET e MQPUT simples (servidor)	mqtrivs.vbp
Colocando e obtendo sequências e estruturas definidas pelo usuário usando MQPUT e MQGET	strings.vbp
Usando estruturas PCF para iniciar e parar um canal	pcfsamp.vbp
Criando uma fila usando MQAI	amqsaicq.vbp
Listando as filas de um gerenciador de filas usando MQAI	amqsailq.vbp
Monitorando eventos usando MQAI	amqsaiem.vbp

Preparando e executando os programas de amostra

Configure seu gerenciador de filas para aceitar com segurança as solicitações de conexão recebidas dos aplicativos em execução no modo cliente.

Antes de começar

Assegure que o gerenciador de filas já exista e tenha sido iniciado. Determine se os registros de autenticação de canal já estão ativados da seguinte forma:

```
DISPLAY QMGR CHLAUTH
```

Esta tarefa espera que os registros de autenticação de canal estejam ativados. Se esse for um gerenciador de filas usado por outros usuários e aplicativos, a mudança dessa configuração afetará todos os outros usuários e aplicativos. Se o gerenciador de filas não usar registros de autenticação de canal, a etapa “4” na página 112 poderá ser substituída por um método de autenticação alternativo (por exemplo, uma saída de segurança) que configure o MCAUSER para o *não privilegiado-ID do usuário* que você obterá na etapa “1” na página 112.

Deve-se saber qual nome de canal seu aplicativo espera usar para que o aplicativo possa ter permissão para usar o canal. Deve-se também saber quais objetos, por exemplo, filas ou tópicos, seu aplicativo espera usar para que seu aplicativo possa ter permissão para usá-los.

Sobre esta tarefa

Esta tarefa cria um ID do usuário não privilegiado para ser usado para um aplicativo cliente que se conecta ao gerenciador de filas. O acesso é concedido ao aplicativo cliente somente para ser capaz de usar o canal que necessita e a fila que precisa por uso desse ID do usuário.

Procedimento

1. Obtenha um ID do usuário no sistema no qual seu gerenciador de filas está em execução. Para essa tarefa, esse ID do usuário não deve ser um usuário administrativo privilegiado. Esse ID do usuário será a autoridade sob a qual a conexão do cliente será executada no gerenciador de filas.
2. Inicie um programa listener com os comandos a seguir, em que:

   qmgr é o nome do gerenciador de filas
   nnnn é o número da porta escolhido

   a) Para sistemas UNIX e Windows :

   ```
   runmqlsr -t tcp -m qmgr -p nnnn
   ```

3. Se seu aplicativo usar SYSTEM.DEF.SVRCONN, então, esse canal já está definido. Se seu aplicativo usar outro canal, crie-o emitindo o comando do MQSC:

   ```
   DEFINE CHANNEL('channel-name') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Channel for use by sample programs')
   ```

   channel-name é o nome de seu canal.
4. Crie uma regra de autenticação de canal permitindo que somente o endereço IP do seu sistema cliente use o canal emitindo o comando do MQSC:

   ```
   SET CHLAUTH('channel-name') TYPE(ADDRESSMAP) ADDRESS('client-machine-IP-address') +
   MCAUSER('non-privileged-user-id')
   ```

   channel-name é o nome de seu canal.
   client-machine-IP-address é o endereço IP de seu sistema cliente.
   Se o aplicativo cliente de amostra estiver em execução na mesma máquina que o gerenciador de filas, então, use o endereço IP '127.0.0.1' se seu aplicativo for se conectar usando 'localhost'. Se várias máquinas clientes diferentes forem se conectar, é possível usar um padrão ou um intervalo em vez de um único endereço IP. Consulte Endereços IP genéricos para obter detalhes.
   non-privileged-user-id é o ID do usuário obtido na etapa “1” na página 112
5. Se seu aplicativo usar SYSTEM.DEFAULT.LOCAL.QUEUE, então, essa fila já está definida. Se seu aplicativo usar outra fila, crie-a emitindo o comando do MQSC:

   ```
   DEFINE QLOCAL('queue-name') DESCR('Queue for use by sample programs')
   ```

   queue-name é o nome de sua fila.
6. Conceda acesso para se conectar e consultar o gerenciador de filas:

   a) Para os sistemas UNIX eWindows emitem os comandos MQSC:

   ```
   SET AUTHREC OBJTYPE(QMGR) PRINCIPAL('non-privileged-user-id') +
   AUTHADD(CONNECT, INQ)
   ```

   non-privileged-user-id é o ID do usuário obtido na etapa “1” na página 112
7. Se seu aplicativo for de ponto a ponto, ou seja, usa filas, conceda acesso para permitir a consulta e a colocação e obtenção de mensagens usando sua fila pelo ID do usuário a ser usado, emitindo os comandos do MQSC:

   a) Para os sistemas UNIX e Windows emitem os comandos MQSC:

   ```
   SET AUTHREC PROFILE('queue-name') OBJTYPE(QUEUE) +
   PRINCIPAL('non-privileged-user-id') AUTHADD(PUT, GET, INQ, BROWSE)
   ```

queue-name é o nome de sua fila.
non-privileged-user-id é o ID do usuário obtido na etapa "1" na página 112

8. Se seu aplicativo for de publicar/assinar, ou seja, usa tópicos, conceda acesso para permitir publicação e assinatura usando seu tópico pelo ID do usuário a ser usado, emitindo os comandos do MQSC:
   a) Para os sistemas UNIX e Windows emitem os comandos MQSC:

```
SET AUTHREC PROFILE('SYSTEM.BASE.TOPIC') OBJTYPE(TOPIC) +
PRINCIPAL('non-privileged-user-id') AUTHADD(PUB, SUB)
```

non-privileged-user-id é o ID do usuário obtido na etapa "1" na página 112
Isso fornecerá acesso *non-privileged-user-id* a qualquer tópico na árvore de tópicos; como alternativa, é possível definir um objeto do tópico usando **DEFINE TOPIC** e conceder acesso somente à parte da árvore de tópicos referida por esse objeto do tópico. Consulte Controlando o acesso do usuário aos tópicos para obter detalhes.

Como proceder a seguir

Agora, seu aplicativo cliente pode se conectar ao gerenciador de filas e colocar ou obter mensagens usando a fila.

Tarefas relacionadas

Fornecendo acesso a um objeto WebSphere MQ em sistemas UNIX ou Linux e Windows

Referências relacionadas

SET CHLAUTH
DEFINE CHANNEL
DEFINE QLOCAL
SET AUTHREC

Preparando e executando programas de amostra em sistemas UNIX

Tabela 18. Onde localizar as amostras para sistemas WebSphere MQ em UNIX and Linux

Conteúdo	**Diretório**
arquivos de origem	*MQ_INSTALLATION_PATH*/samp
arquivos de origem do manipulador da fila de mensagens não entregues	*MQ_INSTALLATION_PATH*/samp/dlq
arquivos executáveis	*MQ_INSTALLATION_PATH*/samp/bin
O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado.	

Os arquivos de amostra dos sistemas WebSphere MQ on UNIX and Linux estão nos diretórios listados em Tabela 18 na página 113 se os padrões foram usados no momento da instalação Para executar as amostras, use as versões executáveis fornecidas ou compile as versões de origem como você faria com qualquer outro aplicativo usando um compilador ANSI. Para obter informações sobre como fazer isso, consulte "Executando os programas de amostra" na página 114.

Preparando e executando programas de amostra em sistemas Windows

Tabela 19. Onde localizar as amostras para o WebSphere MQ para Windows

Conteúdo	**Diretório**
Código de origem C	*MQ_INSTALLATION_PATH*\Tools\C\Samples

Tabela 19. Onde localizar as amostras para o WebSphere MQ para Windows (continuação)	
Conteúdo	**Diretório**
Código fonte para a amostra do manipulador de mensagens não entregues	\tools\c\samples\dlq do*MQ_INSTALLATION_PATH*
Código fonte COBOL	*MQ_INSTALLATION_PATH*\Tools\Cobol \ Amostras
Arquivos C executáveis [1]	*MQ_INSTALLATION_PATH*\ Tools\C\Samples\Bin (versões 32 bits) do *MQ_INSTALLATION_PATH*\ Tools\C\Samples\Bin64 (versões 64 bits)
Arquivos de amostra do MQSC	*MQ_INSTALLATION_PATH*\Tools\MQSC\Samples
Código fonte Visual Basic	*MQ_INSTALLATION_PATH*\Tools\VB\SampVB6
Amostras .NET	*MQ_INSTALLATION_PATH*\Tools\dotnet \ Amostras

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

Nota:

1. Versões de 64 bits estão disponíveis de algumas amostras de arquivos C executáveis.

Os arquivos de amostra do WebSphere MQ para Windows estão nos diretórios listados em Tabela 19 na página 113 se os padrões foram usados no momento da instalação; a unidade de instalação é padronizada para < c:>. Para executar as amostras, use as versões executáveis fornecidas ou compile as versões de origem como faria com qualquer outro WebSphere MQ para aplicativos Windows . Para obter informações sobre como fazer isso, consulte “Executando os programas de amostra” na página 114.

Executando os programas de amostra

Considere usar este tópico ao executar os programas de amostra entre diferentes plataformas.

Antes de ser possível executar quaisquer dos programas de amostra, crie um gerenciador de filas e configure as definições padrão. Isso é explicado em Administrando.

Nas plataformas Windows, UNIX e Linux

As amostras precisam de um conjunto de filas com o qual trabalhar. Use suas próprias filas ou execute o arquivo amqscos0.tst do MQSC de amostra para criar um conjunto.

Para fazer isso em sistemas UNIX and Linux, insira:

- `runmqsc QManagerName <amqscos0.tst >/tmp/sampobj.out`

Verifique o arquivo `sampobj.out` para assegurar que não haja nenhum erro.

Para fazer isso nos sistemas Windows , insira:

- `runmqsc QManagerName  <amqscos0.tst > sampobj.out`

Verifique o arquivo `sampobj.out` para assegurar que não haja nenhum erro. Esse arquivo encontra-se em seu diretório atual.

Agora, é possível executar os aplicativos de amostra. Insira o nome do aplicativo de amostra seguido por quaisquer parâmetros, por exemplo:

- `amqsput myqueue qmanagername`

em que `myqueue` é o nome da fila na qual as mensagens serão colocadas e `qmanagername` é o gerenciador de filas que possui `myqueue`.

Consulte a descrição das amostras individuais para obter informações sobre os parâmetros que cada uma delas espera.

Comprimento do nome da fila

Para os programas de amostra em COBOL, ao passar nomes de filas como parâmetros, deve-se fornecer 48 caracteres, preenchendo com caracteres em branco, se necessário. Qualquer coisa diferente de 48 caracteres faz com que o programa falhe com o código de razão 2085.

Exemplos de Inquire, Set e Echo

Para os exemplos de Inquire, Set e Echo, as definições de amostra acionam as versões de C dessas amostras.

Se desejar as versões de COBOL, deve-se mudar as definições de processo:

- SYSTEM.SAMPLE.INQPROCESS
- SYSTEM.SAMPLE.SETPROCESS
- SYSTEM.SAMPLE.ECHOPROCESS

Nos sistemas Windows, UNIX and Linux, faça isso editando o arquivo `amqscos0.tst` e mudando os nomes dos arquivos executáveis de C para os nomes dos arquivos executáveis de COBOL antes de usar o comando **runmqsc**, conforme mostrado anteriormente.

O programa de amostra de saída de API

A saída API de amostra gera um rastreio de MQI para um arquivo especificado pelo usuário com um prefixo definido na variável de ambiente MQAPI_TRACE_LOGFILE.

Para obter informações adicionais sobre as saídas de API, consulte “Escrevendo e compilando saídas de API” na página 393.

Fonte
amqsaxe0.c

Binário
amqsaxe

Configurando para a saída de amostra

1. Inclua o seguinte ao arquivo qm.ini.

   Plataformas diferentes de Windows

   ```
   ApiExitLocal:
      Sequence=100
      Function=EntryPoint
      Module=MQ_INSTALLATION_PATH/samp/bin/amqsaxe
      Name=SampleApiExit
   ```

   em que *MQ_INSTALLATION_PATH* representa o diretório no qual o IBM WebSphere MQ está instalado.

   Windows

   ```
   ApiExitLocal:
      Sequence=100
      Function=EntryPoint
      Module=MQ_INSTALLATION_PATH\Tools\c\Samples\bin\amqsaxe
      Name=SampleApiExit
   ```

   em que *MQ_INSTALLATION_PATH* representa o diretório no qual o IBM WebSphere MQ está instalado.

2. Configure a variável de ambiente

```
MQAPI_TRACE_LOGFILE=/tmp/MqiTrace
```

3. Execute seu aplicativo.

   Arquivos de saída são criados no diretório /tmp com nomes como: `MqiTrace.<pid>.<tid>.log`

O programa de amostra Asynchronous Consumption

O programa de amostra amqscbf demonstra o uso de MQCB e MQCTL para consumir mensagens de diversas filas de forma assíncrona.

amqscbf é fornecido como código-fonte em C e um executável binário de cliente e servidor nas plataformas Windows, UNIX and Linux.

O programa é iniciado a partir da linha de comandos e aceita os seguintes parâmetros opcionais:

```
Usage: [Options] <Queue Name> { <Queue Name> }
  where Options are:
    -m <Queue Manager Name>
    -o <Open options>
    -r <Reconnect Type>
       d Reconnect Disabled
       r Reconnect
       m Reconnect Queue Manager
```

Forneça mais de um nome de fila para ler mensagens de diversas filas (no máximo dez filas são suportadas pela amostra).

Nota: *Reconnect type* é válido somente para programas clientes.

exemplo

O exemplo mostra amqscbf executado como um programa do servidor lendo uma mensagem de QL1 e, em seguida, sendo interrompido.

Use o WebSphere MQ Explorer para colocar uma mensagem de teste no QL1.. Pare o programa pressionando Enter.

```
C:\>amqscbf QL1
Sample AMQSCBF0 start

Press enter to end
Message Call (9 Bytes) :
Message 1

Sample AMQSCBF0 end
```

O que amqscbf demonstra

A amostra mostra como ler mensagens de diversas filas na ordem de chegada. Isso exigiria muito mais código usando MQGET síncrono. No caso de consumo assíncrono, nenhuma pesquisa é necessária e o gerenciamento de encadeamento e de armazenamento é executado pelo WebSphere MQ Um exemplo do "mundo real" precisaria lidar com erros; na amostra, os erros são gravados no console.

O código de amostra tem as etapas a seguir,

1. Definir a função de retorno de chamada de consumo de mensagem única,

```
void MessageConsumer(MQHCONN      hConn,
                     MQMD    * pMsgDesc,
                     MQGMO   * pGetMsgOpts,
                     MQBYTE  * Buffer,
                     MQCBC   * pContext)
{ ... }
```

2. Conectar-se ao gerenciador de filas,

```
MQCONNX(QMName,&cno,&Hcon,&CompCode,&CReason);
```

3. Abrir as filas de entrada e associar cada uma à função de retorno de chamada `MessageConsumer`,

```
MQOPEN(Hcon,&od,O_options,&Hobj,&OpenCode,&Reason);
cbd.CallbackFunction = MessageConsumer;
MQCB(Hcon,MQOP_REGISTER,&cbd,Hobj,&md,&gmo,&CompCode,&Reason);
```

   `cbd.CallbackFunction` não precisa ser configurado para cada fila; é um campo somente de entrada. Mas você poderia associar uma função de retorno de chamada diferente a cada fila.

4. Iniciar o consumo das mensagens,

```
MQCTL(Hcon,MQOP_START,&ctlo,&CompCode,&Reason);
```

5. Esperar até que o usuário tenha pressionado Enter e, em seguida, parar o consumo de mensagens,

```
MQCTL(Hcon,MQOP_STOP,&ctlo,&CompCode,&Reason);
```

6. Por fim, desconectar-se do gerenciador de filas,

```
MQDISC(&Hcon,&CompCode,&Reason);
```

O programa de amostra Asynchronous Put

Aprenda sobre execução da amostra amqsapt e o design do programa de amostra Asynchronous Put.

O programa de amostra Asynchronous Put coloca as mensagens em uma fila, usando a chamada MQPUT assíncrona e, em seguida, recupera as informações de status usando a chamada MQSTAT. Consulte “Recursos demonstrados nos programas de amostra” na página 99 para obter o nome deste programa em diferentes plataformas.

Executando a Amostra amqsapt

Este programa usa até 6 parâmetros:

1. O nome da fila de destino (obrigatório)
2. O nome do gerenciador de filas (opcional)
3. Opções de abertura (opcional)
4. Opções de fechamento (opcional)
5. O nome do gerenciador de filas de destino (opcional)
6. O nome da fila dinâmica (opcional)

Se um gerenciador de filas não for especificado, amqsapt se conecta ao gerenciador de filas padrão.

Design do programa de amostra Asynchronous Put

O programa usa a chamada MQOPEN com as opções de saída fornecidas ou com as opções MQOO_OUTPUT e MQOO_FAIL_IF_QUIESCING para abrir a fila de destino para colocar mensagens.

Se ele não puder abrir a fila, o programa envia uma mensagem de erro contendo o código de razão retornado pela chamada MQOPEN. Para manter o programa simples, nesta e em chamadas MQI subsequentes, o programa usa valores padrão para muitas das opções.

Para cada linha de entrada, o programa lê o texto em um buffer e usa a chamada MQPUT com MQPMO_ASYNC_RESPONSE para criar uma mensagem de datagrama que contém o texto dessa linha e o coloca de forma assíncrona na fila de destino. O programa continua até chegar ao fim da entrada ou a chamada MQPUT falha. Se o programa atingir o final da entrada, ele fechará a fila usando a chamada MQCLOSE.

O programa, em seguida, emite a chamada MQSTAT, retornando uma estrutura MQSTS, e exibe mensagens que contêm o número de mensagens colocadas com êxito, o número de mensagens colocadas com um aviso e o número de falhas.

Os programas de amostra Browse

As mensagens de Procurar por programas de amostra em uma fila usando a chamada MQGET.

Consulte “Recursos demonstrados nos programas de amostra” na página 99 para obter os nomes desses programas.

Design do programa de amostra Browse

O programa abre a fila de destino usando a chamada MQOPEN com a opção MQOO_BROWSE. Se ele não puder abrir a fila, o programa envia uma mensagem de erro contendo o código de razão retornado pela chamada MQOPEN.

Para cada mensagem na fila, o programa usará a chamada MQGET para copiar a mensagem da fila, em seguida, exibirá os dados contidos na mensagem. A chamada MQGET usa estas opções:

MQGMO_BROWSE_NEXT
Depois da chamada MQOPEN, o cursor de pesquisa é posicionado logicamente antes da primeira mensagem na fila, portanto, essa opção faz com que a ***primeira*** mensagem seja retornada quando a chamada é feita pela primeira vez.

MQGMO_NO_WAIT
O programa não espera se não houver mensagens na fila.

MQGMO_ACCEPT_TRUNCATED_MSG
A chamada MQGET especifica um buffer de tamanho fixo. Se uma mensagem for maior do que esse buffer, o programa exibe a mensagem truncada, juntamente com um aviso de que a mensagem foi truncada.

O programa demonstra como os campos *MsgId* e *CorrelId* devem ser desmarcados da estrutura MQMD após cada chamada MQGET, porque a chamada define esses campos para os valores contidos na mensagem que ela recupera. Desmarcar esses campos significa que sucessivas chamadas MQGET recuperam as mensagens na ordem em que elas são retidas na fila.

O programa continua até o final da fila; a chamada MQGET retorna o código de razão MQRC_NO_MSG_AVAILABLE e o programa exibe uma mensagem de aviso. Se a chamada MQGET falhar, o programa exibirá uma mensagem de erro que contém o código de razão.

O programa, então, fecha a fila usando a chamada MQCLOSE.

Sistemas UNIX, Linux e Windows

Considere usar este tópico ao aprender sobre Procurar programas de amostra nos sistemas UNIX, Linux e Windows .

A versão de C do programa aceita dois parâmetros

1. O nome da fila de origem (necessário)
2. O nome do gerenciador de filas (opcional)

Se um gerenciador de filas não for especificado, ele se conectará ao padrão. Por exemplo, insira um dos seguintes:

- `amqsgbr myqueue qmanagername`
- `amqsgbrc myqueue qmanagername`
- `amq0gbr0 myqueue`

em que `myqueue` é o nome da fila a partir da qual as mensagens serão visualizadas e `qmanagername` é o gerenciador de filas que possui `myqueue`.

Se você omitir o qmanagername, quando estiver executando a amostra em C, supõe-se que o gerenciador de filas padrão possui a fila.

A versão em COBOL não tem nenhum parâmetro. Ela se conecta ao gerenciador de filas padrão e quando executá-la, será solicitado o seguinte:

```
Please enter the name of the target queue
```

Somente os primeiros 50 caracteres de cada mensagem são exibidos, seguidos por `- - - truncated` quando este é o caso.

O programa de amostra Browser

O programa de amostra Browser lê e grava o descritor de mensagens e os campos de conteúdo de mensagens de todas as mensagens em uma fila.

O programa de amostra é gravado como um utilitário, não apenas para demonstrar uma técnica. Consulte "Recursos demonstrados nos programas de amostra" na página 99 para obter os nomes desses programas.

Este programa utiliza estes parâmetros:

1. O nome da fila de origem
2. O nome do gerenciador de filas
3. Um parâmetro opcional para as propriedades

Os dois primeiros parâmetros de entrada para este programa são obrigatórios Por exemplo, inicie o programa de uma das seguintes maneiras:

- `amqsbcg myqueue qmanagername`
- `amqsbcgc myqueue qmanagername`

em que `myqueue` é o nome da fila na qual as mensagens serão procuradas e `qmanagername` é o gerenciador de filas que possui `myqueue`.

Ele lê cada mensagem da fila e grava o seguinte para stdout:

- Campos do descritor de mensagens formatados
- Dados da mensagem (com dump em hex e, quando possível, formato de caractere)

Os valores permitidos para o parâmetro da propriedade são:

Value	Comportamento
0	Comportamento padrão, como era para a V6 As propriedades que são entregues ao aplicativo dependem do atributo de fila *PropertyControl* do qual a mensagem é recuperada.

1	Um identificador de mensagens é criado e usado com o MQGET. As propriedades da mensagem, exceto aquelas contidas no descritor de mensagens (ou extensão), são exibidas de forma semelhante para o descritor de mensagens. Por exemplo:
	`****Message properties****`
	`  <property name> : <property value>`
	Ou se nenhuma propriedade estiver disponível:
	`****Message properties****`
	`  None`
	Valores numéricos são exibidos usando o `printf`, os valores de sequência são colocados entre aspas simples e as sequências de bytes são marcadas com X e colocadas entre aspas simples, como para o descritor de mensagens.

Value	Comportamento
2	MQGMO_NO_PROPERTIES é especificado, de forma que apenas as propriedades do descritor de mensagens sejam retornadas.
3	MQGMO_PROPERTIES_FORCE_MQRFH2 é especificado, de forma que todas as propriedades sejam retornadas nos dados da mensagem.
4	MQGMO_PROPERTIES_COMPATIBILITY é especificado, de modo que todas as propriedades possam ser retornadas dependendo se uma propriedade da versão 6 for incluída, caso contrário, as propriedades serão descartados

O programa é restrito ao imprimir os primeiros 65535 caracteres da mensagem e falhará com a razão *msg truncada* se uma mensagem mais longa for lida.

Consulte o Administrando para obter um exemplo da saída desse utilitário.

A amostra de transação CICS

Um programa de transação CICS de amostra é fornecido, denominado amqscic0.ccs para o código fonte e amqscic0 para a versão executável. É possível construir transações usando os recursos padrão do CICS

Consulte “Construindo um aplicativo IBM WebSphere MQ” na página 435 para obter detalhes sobre os comandos necessários para sua plataforma.

A transação lê mensagens da fila de transmissão SYSTEM.SAMPLE.CICS.WORKQUEUE no gerenciador de filas padrão e coloca-os na fila local, cujo nome está contido no cabeçalho de transmissão da mensagem Quaisquer falhas são enviadas à fila SYSTEM.SAMPLE.CICS.DLQ.

Nota: É possível usar um script de amostra MQSC amqscic0.tst para criar estas filas e filas de entrada de amostra.

O programa de amostra Connect

O programa de amostra Connect permite explorar a chamada MQCONNX e suas opções a partir de um cliente. A amostra se conecta ao gerenciador de filas usando a chamada MQCONNX, consulta sobre o nome do gerenciador de filas usando a chamada MQINQ e exibe o mesmo. Além disso, aprenda sobre como executar a amostra amqscnxc.

Nota: O programa de amostra Connect é uma amostra do cliente. É possível compilar e executar a mesma em um servidor, mas a função é significativa somente em um cliente e somente arquivos executáveis de cliente são fornecidos.

Executando a amostra amqscnxc

A sintaxe da linha de comandos do programa de amostra Connect é:

```
amqscnxc [-x ConnName [-c SvrconnChannelName]] [QMgrName]
```

Os parâmetros são opcionais e sua ordem não é importante, exceto para QMgrName, que, se especificado, deve vir por último. Os parâmetros são:

ConnName
O nome da conexão TCP/IP do gerenciador de filas do servidor

SvrconnChannelName
O nome do canal de conexão do servidor

QMgrName
O nome do gerenciador de filas de destino

Se você não especificar o nome da conexão TCP/IP, MQCONNX será emitido com *ClientConnPtr* configurado para NULL. Se você especificar o nome da conexão TCP/IP, mas não o canal de conexão do servidor (o inverso não é permitido), a amostra usará o nome SYSTEM.DEF.SVRCONN. Se não especificar

o gerenciador de filas de destino, a amostra se conecta a qualquer gerenciador de filas que esteja atendendo no nome da conexão TCP/IP especificada.

Nota: Se inserir um ponto de interrogação como o único parâmetro ou se inserir parâmetros incorretos, receberá uma mensagem explicando como usar o programa.

Se executar a amostra sem opções da linha de comandos, os conteúdos da variável de ambiente MQSERVER serão usados para determinar as informações de conexão. (Neste exemplo, MQSERVER é configurado para `SYSTEM.DEF.SVRCONN/TCP/machine.site.company.com`.) Você verá uma saída semelhante a esta:

```
Sample AMQSCNXC start
Connecting to the default queue manager
with no client connection information specified.
Connection established to queue manager machine

Sample AMQSCNXC end
```

Se executar a amostra e fornecer um nome de conexão TCP/IP e um nome de canal de conexão de servidor, mas nenhum nome do gerenciador de filas de destino, desta forma:

```
amqscnxc -x machine.site.company.com -c SYSTEM.ADMIN.SVRCONN
```

o nome do gerenciador de filas padrão será usado e você verá uma saída semelhante a esta:

```
Sample AMQSCNXC start
Connecting to the default queue manager
using the server connection channel SYSTEM.ADMIN.SVRCONN
on connection name machine.site.company.com.
Connection established to queue manager MACHINE

Sample AMQSCNXC end
```

Se executar a amostra e fornecer um nome de conexão TCP/IP e um nome do gerenciador de filas de destino, desta forma:

```
amqscnxc -x machine.site.company.com MACHINE
```

você verá uma saída semelhante a esta:

```
Sample AMQSCNXC start
Connecting to queue manager MACHINE
using the server connection channel SYSTEM.DEF.SVRCONN
on connection name machine.site.company.com.
Connection established to queue manager MACHINE

Sample AMQSCNXC end
```

O programa de amostra Data-Conversion

O programa de amostra de conversão de dados é uma estrutura básica de uma rotina de saída de conversão de dados. Saiba mais sobre o design da amostra de conversão de dados.

Consulte “Recursos demonstrados nos programas de amostra” na página 99 para obter os nomes desses programas.

Design da amostra de conversão de dados

Cada rotina de saída de conversão de dados converte um único formato da mensagem nomeada. Essa estrutura básica é destinada como um wrapper para fragmentos de código gerados pelo programa utilitário da geração de saída de conversão de dados.

O utilitário produz um fragmento de código para cada estrutura de dados. Diversas estruturas compõem um formato, portanto, vários fragmentos de código são incluídos nesta estrutura básica para produzir uma rotina para fazer a conversão de dados do formato inteiro.

O programa, então, verifica se a conversão foi bem-sucedida ou se houve falha e retorna os valores necessários para o responsável pela chamada.

Amostras de coordenação de banco de dados

São fornecidas duas amostras que demonstram como o WebSphere MQ pode coordenar atualizações do WebSphere MQ e do banco de dados dentro da mesma unidade de trabalho.

Essas amostras são:

1. AMQSXAS0 (em C) ou AMQ0XAS0 (em COBOL), que atualiza um único banco de dados em uma unidade de trabalho WebSphere MQ .
2. AMQSXAG0 (em C) ou AMQ0XAG0 (em COBOL), AMQSXAB0 (em C) ou AMQ0XAB0 (em COBOL) e AMQSXAF0 (em C) ou AMQ0XAF0 (em COBOL), que juntos atualizam dois bancos de dados em uma unidade de trabalho do WebSphere MQ , mostrando como diversos bancos de dados podem ser acessados. Essas amostras são fornecidas para mostrar o uso da chamada MQBEGIN, chamadas SQL combinadas e WebSphere MQ e onde e quando conectar a um banco de dados.

Figura 18 na página 122 mostra como as amostras fornecidas são usadas para atualizar os bancos de dados:

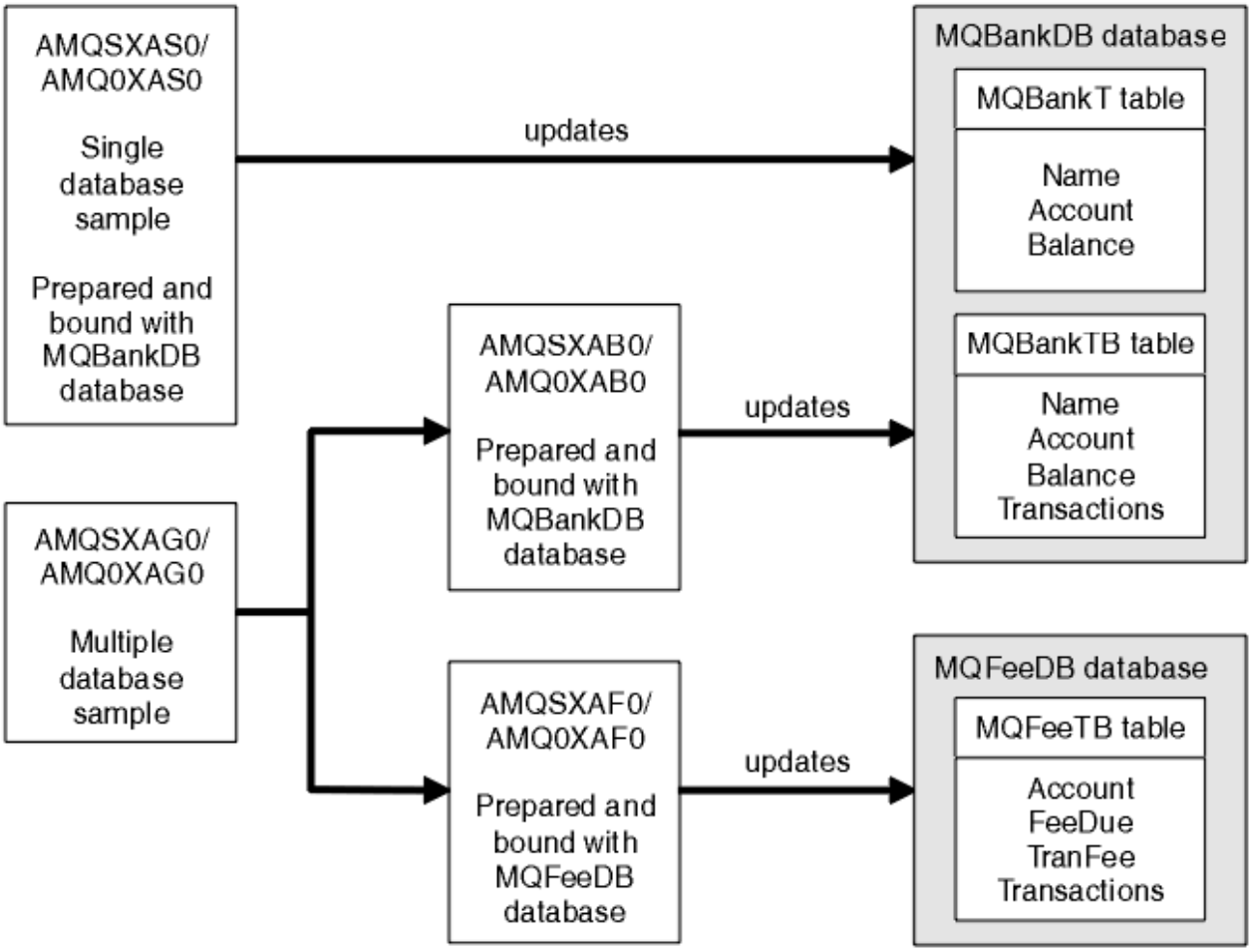


Figura 18. As amostras de coordenação de banco de dados

Os programas leem uma mensagem de uma fila (sob o ponto de sincronização), em seguida, usando as informações na mensagem, obtêm as informações relevantes do banco de dados e atualizam o mesmo. O novo status do banco de dados será, então, impresso.

A lógica do programa é a seguinte:

1. Use o nome da fila de entrada a partir do argumento de programa
2. Conecte-se ao gerenciador de filas padrão (ou, como opção, ao nome fornecido em C) usando MQCONN
3. Abra uma fila (usando MQOPEN) para a entrada enquanto não houver falhas
4. Inicie uma unidade de trabalho usando MQBEGIN
5. Obtenha a próxima mensagem (usando MQGET) da fila sob o ponto de sincronização
6. Obtenha informações de bancos de dados
7. Atualize as informações de bancos de dados
8. Consolide mudanças usando MQCMIT
9. Imprima informações atualizadas (nenhuma mensagem disponível conta como uma falha e o loop é finalizado)
10. Feche a fila usando MQCLOSE
11. Desconecte-se da fila usando MQDISC

Os cursores de SQL são usados nas amostras, de modo que leituras dos bancos de dados (ou seja, diversas instâncias) são bloqueadas enquanto uma mensagem estiver sendo processada, permitindo que várias instâncias desses programas sejam executadas simultaneamente. Os cursores são explicitamente abertos, mas implicitamente fechados pela chamada MQCMIT.

A amostra de banco de dados única (AMQSXAS0 ou AMQ0XAS0) não tem instruções SQL CONNECT e a conexão com o banco de dados é feita implicitamente pelo WebSphere MQ com a chamada MQBEGIN. A amostra de diversos banco de dados (AMQSXAG0 ou AMQ0XAG0, AMQSXAB0 ou AMQ0XAB0 e AMQSXAF0 ou AMQ0XAF0) tem instruções SQL CONNECT, pois alguns produtos de banco de dados permitem somente uma conexão ativa. Se esse não for o caso para seu produto de banco de dados ou se você estiver acessando um único banco de dados em produtos de vários bancos de dados, as instruções SQL CONNECT podem ser removidas.

As amostras são preparadas com o produto de banco de dados IBM DB2 , portanto, pode ser necessário modificá-las para trabalhar com outros produtos de banco de dados

A verificação de erro de SQL usa rotinas em UTIL.C e CHECKERR.CBL fornecido por DB2. Elas devem ser compiladas ou substituídas antes da compilação e ligação.

Nota: Se estiver usando o código-fonte CHECKERR.MFC do Micro Focus COBOL para verificação de erro de SQL, deve-se mudar o ID do programa para maiúsculas, ou seja, CHECKERR, para que AMQ0XAS0 faça a ligação corretamente.

Criando os bancos de dados e as tabelas

Crie os bancos de dados e as tabelas antes de compilar as amostras.

Para criar os bancos de dados, use o método usual para seu produto de banco de dados, por exemplo:

```
DB2 CREATE DB MQBankDB
DB2 CREATE DB MQFeeDB
```

Crie as tabelas usando instruções SQL da seguinte forma:

Em C:

```
EXEC SQL CREATE TABLE MQBankT(Name         VARCHAR(40) NOT NULL,
                              Account      INTEGER     NOT NULL,
                              Balance      INTEGER     NOT NULL,
                              PRIMARY KEY (Account));
```

```
EXEC SQL CREATE TABLE MQBankTB(Name         VARCHAR(40) NOT NULL,
                               Account      INTEGER     NOT NULL,
                               Balance      INTEGER     NOT NULL,
                               Transactions INTEGER,
                               PRIMARY KEY (Account));

EXEC SQL CREATE TABLE MQFeeTB(Account      INTEGER     NOT NULL,
                              FeeDue       INTEGER     NOT NULL,
                              TranFee      INTEGER     NOT NULL,
                              Transactions INTEGER,
                              PRIMARY KEY (Account));
```

Em COBOL:

```
 EXEC SQL CREATE TABLE
   MQBankT(Name      VARCHAR(40) NOT NULL,
           Account   INTEGER     NOT NULL,
           Balance   INTEGER     NOT NULL,
           PRIMARY KEY (Account))
   END-EXEC.

EXEC SQL CREATE TABLE
  MQBankTB(Name      VARCHAR(40) NOT NULL,
           Account   INTEGER     NOT NULL,
           Balance   INTEGER     NOT NULL,
           Transactions INTEGER,
           PRIMARY KEY (Account))
  END-EXEC.

EXEC SQL CREATE TABLE
  MQFeeTB(Account      INTEGER     NOT NULL,
          FeeDue       INTEGER     NOT NULL,
          TranFee      INTEGER     NOT NULL,
          Transactions INTEGER,
          PRIMARY KEY (Account))
  END-EXEC.
```

Insira os dados nas tabelas usando instruções SQL, da seguinte forma:

```
EXEC SQL INSERT INTO MQBankT VALUES ('Mr Fred Bloggs',1,0);
EXEC SQL INSERT INTO MQBankT VALUES ('Mrs S Smith',2,0);
EXEC SQL INSERT INTO MQBankT VALUES ('Ms Mary Brown',3,0);
.
EXEC SQL INSERT INTO MQBankTB VALUES ('Mr Fred Bloggs',1,0,0);
EXEC SQL INSERT INTO MQBankTB VALUES ('Mrs S Smith',2,0,0);
EXEC SQL INSERT INTO MQBankTB VALUES ('Ms Mary Brown',3,0,0);
.
EXEC SQL INSERT INTO MQFeeTB VALUES (1,0,50,0);
EXEC SQL INSERT INTO MQFeeTB VALUES (2,0,50,0);
EXEC SQL INSERT INTO MQFeeTB VALUES (3,0,50,0);
.
```

Nota: Para COBOL, use as mesmas instruções SQL, mas inclua END_EXEC no final de cada linha.

Pré-compilação, compilação e vinculação de amostras

Aprenda sobre a pré-compilação, compilação e vinculação de amostras em C e COBOL.

Pré-compile os arquivos .SQC (em C) e arquivos .SQB (em COBOL) e ligue-os em relação ao banco de dados apropriado para produzir os arquivos .C ou .CBL. Para fazer isto, use o método típico para seu produto de banco de dados.

Pré-compilando em C

```
db2 connect to MQBankDB
db2 prep AMQSXAS0.SQC
db2 connect reset

db2 connect to MQBankDB
db2 prep AMQSXAB0.SQC
db2 connect reset
```

```
db2 connect to MQFeeDB
db2 prep AMQSXAF0.SQC
db2 connect reset
```

Pré-compilando em COBOL

```
db2 connect to MQBankDB
db2 prep AMQ0XAS0.SQB bindfile target ibmcob
db2 bind AMQ0XAS0.BND
db2 connect reset

db2 connect to MQBankDB
db2 prep AMQ0XAB0.SQB bindfile target ibmcob
db2 bind AMQ0XAB0.BND
db2 connect reset

db2 connect to MQFeeDB
db2 prep AMQ0XAF0.SQB bindfile target ibmcob
db2 bind AMQ0XAF0.BND
db2 connect reset
```

Compilando e vinculando-se

Os comandos de exemplo a seguir usam os símbolos *<DB2TOP>* e *MQ_INSTALLATION_PATH*. *<DB2TOP>* representa o diretório de instalação para o produto DB2 . *MQ_INSTALLATION_PATH* Representa o diretório de alto nível no qual o WebSphere MQ está instalado

- No AIX, o caminho do diretório é:.

  ```
  /usr/lpp/db2_05_00
  ```

- No HP-UX e Solaris, o caminho do diretório é:

  ```
  /opt/IBMdb2/V5.0
  ```

- Nos sistemas Windows , o caminho do diretório depende do caminho escolhido durante a instalação do produto Se você escolher as configurações padrão, o caminho será:

  ```
  c:\sqllib
  ```

Nota: Antes de emitir o comando de link em sistemas Windows , certifique-se de que a variável de ambiente LIB contenha caminhos para as bibliotecas DB2 e WebSphere MQ

Copie os arquivos a seguir em um diretório temporário:

- O arquivo `amqsxag0.c` de sua instalação do WebSphere MQ

  Nota: Esse arquivo pode ser encontrado nos diretórios a seguir:

  – Nos sistemas UNIX and Linux:

  ```
  MQ_INSTALLATION_PATH/samp/xatm
  ```

  – Nos sistemas Windows:

  ```
  MQ_INSTALLATION_PATH\tools\c\samples\xatm
  ```

- Os arquivos `.c` que você obteve pré-compilando os arquivos de origem `.sqc` , `amqsxas0.sqc` , `amqsxaf0.sqc` e `amqsxab0.sqc`
- Os arquivos `util.c` e `util.h` de sua instalação do DB2 .

Nota: Esses arquivos podem ser localizados no diretório:

```
<DB2TOP>/samples/c
```

Crie os arquivos de objeto para cada arquivo `.c` usando o comando do compilador a seguir para a plataforma que você está usando:

- AIX

```
xlc_r -IMQ_INSTALLATION_PATH/inc -I
<DB2TOP>/include -c -o
 <FILENAME>.o <FILENAME>.c
```

- HP-UX

```
cc -Aa +z -IMQ_INSTALLATION_PATH/inc -I
<DB2TOP>/include -c -o
 <FILENAME>.o <FILENAME>.c
```

- Solaris

```
cc -Aa -KPIC -mt -IMQ_INSTALLATION_PATH
/inc -I<DB2TOP>/include -c -o
 <FILENAME>.o <FILENAME>.c
```

- Sistemas Windows

```
cl /c /IMQ_INSTALLATION_PATH\tools\c\include /I
<DB2TOP>\include
 <FILENAME>.c
```

Construa o arquivo executável `amqsxag0` usando o comando de link a seguir para a plataforma que você está usando:

- AIX

```
xlc_r -H512 -T512 -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib
 -lmqm util.o amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- HP-UX Revisão 11i

```
ld -E -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib -lmqm -lc -lpthread -lcl
 /lib/crt0.o util.o amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- Solaris

```
cc -mt -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib
 -lmqm -lthread -lsocket -lc -lnsl -ldl util.o
  amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- Sistemas Windows

```
link util.obj amqsxaf0.obj amqsxab0.obj amqsxag0.obj mqm.lib db2api.lib
 /out:amqsxag0.exe
```

Construa o arquivo executável `amqsxas0` usando os comandos compilar e vincular a seguir para a plataforma que você está usando:

- AIX

```
xlc_r -H512 -T512 -L<DB2TOP>/lib -ldb2
 -LMQ_INSTALLATION_PATH/lib -lmqm util.o amqsxas0.o -o amqsxas0
```

- HP-UX Revisão 11i

```
ld -E -L<DB2TOP>/lib -ldb2 -LMQ_INSTALLATION_PATH/lib -lmqm -lc -lpthread
 -lcl /lib/crt0.o util.o amqsxas0.o -o amqsxas0
```

- Solaris

```
cc -mt -L<DB2TOP>/lib -ldb2-LMQ_INSTALLATION_PATH/lib
 -lqm -lthread -lsocket -lc -lnsl -ldl util.o
  amqsxas0.o -o amqsxas0
```

- Sistemas Windows

```
link util.obj amqsxas0.obj mqm.lib db2api.lib /out:amqsxas0.exe
```

Informações adicionais

Se você estiver trabalhando no AIX ou HP-UX e desejar acessar o Oracle, use o compilador xlc_r e o link para libmqm_r.a.

Executando as amostras

Use estas informações para aprender como configurar o gerenciador de filas antes de executar amostras de coordenação de banco de dados em C e COBOL.

Antes de executar as amostras, configure o gerenciador de filas com o produto de banco de dados que está usando. Para obter informações sobre como fazer isso, consulte “Cenário 1: Gerenciador de Filas Executa a Coordenação” na página 43.

O títulos a seguir fornecem informações sobre como executar amostras em C e COBOL:

- “Amostras em C” na página 127
- “Amostras em COBOL” na página 128

Amostras em C

As mensagens devem estar no formato a seguir para serem lidas a partir de uma fila:

```
UPDATE Balance change=nnn WHERE Account=nnn
```

AMQSPUT pode ser usado para colocar a mensagem na fila.

As amostras de coordenação de banco de dados aceitam dois parâmetros:

1. Nome da fila (obrigatório)
2. Nome do gerenciador de filas (opcional)

Supondo que tenha criado e configurado um gerenciador de filas para a amostra de banco de dados único chamada singDBQM, com uma fila chamada singDBQ, você incrementa a conta do Sr. Fred Bloggs em 50 da seguinte forma:

```
  AMQSPUT singDBQ singDBQM
```

Em seguida, digite a mensagem a seguir:

```
UPDATE Balance change=50 WHERE Account=1
```

É possível colocar várias mensagens na fila.

```
AMQSXAS0 singDBQ singDBQM
```

O status atualizado da conta do Sr. Fred Bloggs será, então, impresso.

Supondo que tenha criado e configurado um gerenciador de filas para a amostra de diversos bancos de dados chamada multDBQM, com uma fila chamada multDBQ, você decrementa a conta da Sra. Mary Brown em 75 da seguinte forma:

```
AMQSPUT multDBQ multDBQM
```

Em seguida, digite a mensagem a seguir:

```
UPDATE Balance change=-75 WHERE Account=3
```

É possível colocar várias mensagens na fila.

```
AMQSXAG0 multDBQ multDBQM
```

O status atualizado da conta da Sra. Mary Brown será, então, impresso.

Amostras em COBOL

As mensagens devem estar no formato a seguir para serem lidas a partir de uma fila:

```
UPDATE Balance change=snnnnnnnn WHERE Account=nnnnnnnn
```

Para simplificar, `Balance change` deve ser um número com oito caracteres com sinal e `Account` deve ser um número de oito caracteres.

A amostra AMQSPUT pode ser usada para colocar as mensagens na fila.

As amostras não aceitam parâmetros e usam o gerenciador de filas padrão. É possível configurar para que somente uma das amostras seja executada por vez. Supondo que tenha configurado o gerenciador de filas padrão para a amostra de banco de dados único, com uma fila chamada singDBQ, você incrementa a conta do Sr. Fred Bloggs em 50 da seguinte forma:

```
AMQSPUT singDBQ
```

Em seguida, digite a mensagem a seguir:

```
UPDATE Balance change=+00000050 WHERE Account=00000001
```

É possível colocar várias mensagens na fila:

```
AMQ0XAS0
```

Digite o nome da fila:

```
singDBQ
```

O status atualizado da conta do Sr. Fred Bloggs será, então, impresso.

Supondo que tenha configurado o gerenciador de filas padrão para a amostra de diversos bancos de dados, com uma fila chamada multDBQ, você decrementa a conta da Sra. Mary Brown em 75 da seguinte forma:

```
AMQSPUT multDBQ
```

Em seguida, digite a mensagem a seguir:

```
UPDATE Balance change=-00000075 WHERE Account=00000003
```

É possível colocar várias mensagens na fila:

```
AMQ0XAG0
```

Digite o nome da fila:

```
multDBQ
```

O status atualizado da conta da Sra. Mary Brown será, então, impresso.

Amostra do manipulador de filas de mensagens não entregues

Uma amostra de manipulador da fila de mensagens não entregues é fornecida, o nome da versão executável é amqsdlq. Se você deseja um manipulador da fila de devoluções diferente de RUNMQDLQ, a origem da amostra está disponível para você usar como sua base.

A amostra é semelhante ao manipulador de mensagens não entregues fornecido no produto, mas o rastreio e relatório de erros são diferentes. Há duas variáveis de ambiente disponíveis para você:

ODQ_TRACE
Configure como YES ou sim para ativar o rastreio

ODQ_MSG
Configure para o nome do arquivo que contém mensagens de erro e de informações. O arquivo fornecido é chamado amqsdlq.msg.

É necessário fazer essas variáveis conhecidas para seu ambiente usando os comandos **export** ou **set** os comandos, dependendo de sua plataforma; o rastreio é desativado usando o comando **unset**.

É possível modificar o arquivo de mensagens de erro, amqsdlq.msg, para adequar a seus próprios requisitos. A amostra coloca as mensagens em stdout, ***não*** no arquivo do log de erros do WebSphere MQ

O Administrando ou o *Guia de gerenciamento de sistemas* para sua plataforma explica como o manipulador de mensagens não entregues funciona e como executá-lo.

O programa de amostra Distribution List

A amostra Distribution List amqsptl0 fornece um exemplo de como colocar uma mensagem em várias filas de mensagens. Ela é baseada na amostra MQPUT, amqsput0.

Executando a amostra Distribution List, amqsptl0

A amostra Distribution List é executada de maneira semelhante às amostras Put.

Ela aceita os parâmetros a seguir:

- Os nomes das filas
- Os nomes dos gerenciadores de filas

Esses valores são inseridos como pares. Por exemplo:

```
amqsptl0 queue1 qmanagername1 queue2 qmanagername2
```

As filas são abertas usando MQOPEN e as mensagens são colocadas nas filas usando MQPUT. Códigos de razão são retornados se qualquer um dos nomes de filas ou de gerenciadores de filas não forem reconhecidos.

Lembre-se de definir canais entre os gerenciadores de filas para que as mensagens possam fluir entre eles. O programa de amostra não faz isso para você.

Design da amostra Distribution List

Put Message Records (MQPMRs) especificam atributos de mensagens para cada destino. A amostra fornece valores para *MsgId* e *CorrelId* e eles substituem os valores especificados na estrutura MQMD.

O campo *PutMsgRecFields* na estrutura MQPMO indica quais campos estão presentes em MQPMRs:

```
MQLONG PutMsgRecFields=MQPMRF_MSG_ID + MQPMRF_CORREL_ID;
```

Em seguida, a amostra aloca os registros de resposta e os registros de objeto. Os registros de objeto (MQORs) requerem pelo menos um par de nomes e um número par de nomes, ou seja, *ObjectName* e *ObjectQMgrName*.

A próxima etapa envolve a conexão com os gerenciadores de filas usando MQCONN. A amostra tenta se conectar ao gerenciador de filas associado à primeira fila no MQOR; se isso falhar, ele passa pelos registros de objeto da vez. Você será informado se não for possível se conectar a nenhum gerenciador de filas e o programa sairá.

As filas de destino são abertas usando MQOPEN e a mensagem é colocada nessas filas usando MQPUT. Quaisquer problemas e falhas são relatados nos registros de resposta (MQRRs).

Por fim, as filas de destino são fechadas usando MQCLOSE e o programa se desconecta do gerenciador de filas usando MQDISC. Os mesmos registros de resposta são usados para cada chamada informando *CompCode* e *Reason*.

Os programas de amostra Echo

Os programas de amostra Echo repetem uma mensagem de uma fila de mensagens para a fila de resposta.

Consulte “Recursos demonstrados nos programas de amostra” na página 99 para obter os nomes desses programas.

Os programas são destinados a execução como programas acionados.

Nos sistemas UNIX, Linux e Windows , sua única entrada é uma estrutura MQTMC2 (mensagem do acionador) que contém o nome de uma fila de destino e o gerenciador de filas. A versão COBOL usa o gerenciador de filas padrão.

Quando você tiver configurado a definição corretamente, primeiro inicie AMQSERV4 em uma tarefa e, em seguida, inicie AMQSREQ4 em outra. Seria possível usar AMQSTRG4 em vez de AMQSERV4, mas os potenciais atrasos de envio de tarefa poderiam tornar mais difícil seguir o que está acontecendo.

Use os programas de amostra Request para enviar mensagens para a fila SYSTEM.SAMPLE.ECHO. Os programas de amostra Echo enviam uma mensagem de resposta contendo os dados na mensagem de solicitação para a fila de resposta especificada na mensagem de solicitação.

Design dos programas de amostra Echo

O programa abre a fila denominada na estrutura da mensagem do acionador que foi passada quando ele foi iniciado. (Por questão de clareza, chamaremos isso de *fila de solicitações*.) O programa usa a chamada MQOPEN para abrir essa fila para entrada compartilhada.

O programa usa a chamada MQGET para remover as mensagens dessa fila. Essa chamada usa as opções MQGMO_ACCEPT_TRUNCATED_MSG, MQGMO_CONVERT e MQGMO_WAIT, com um intervalo de espera de 5 segundos. O programa testa o descritor de cada mensagem para ver se é uma mensagem de solicitação; se não for, o programa descartará a mensagem e exibirá uma mensagem de aviso.

Para cada linha de entrada, o programa lê o texto em um buffer e usa a chamada MQPUT1 para colocar uma mensagem de solicitação, contendo o texto dessa linha, na fila de resposta.

Se a chamada MQGET falhar, o programa colocará uma mensagem de relatório na fila de resposta, configurando o campo *Feedback* do descritor de mensagens como o código de razão retornado pelo MQGET.

Quando não houver nenhuma mensagem restante na fila de solicitações, o programa fecha essa fila e desconecta do gerenciador de filas.

Os programas de amostra Get

Os programas de amostra Get recebem mensagens de uma fila usando a chamada MQGET.

Consulte "Recursos demonstrados nos programas de amostra" na página 99 para obter os nomes desses programas.

Design do programa de amostra Get

O programa abre a fila de destino usando a chamada MQOPEN com a opção MQOO_INPUT_AS_Q_DEF. Se não puder abrir a fila, o programa exibirá uma mensagem de erro contendo o código de razão retornado pela chamada MQOPEN.

Para cada mensagem na fila, o programa usa a chamada MQGET para remover a mensagem da fila e, em seguida, exibe os dados contidos na mensagem. A chamada MQGET usa a opção MQGMO_WAIT, especificando um *WaitInterval* de 15 segundos, para que o programa aguarde esse período se não houver nenhuma mensagem na fila. Se nenhuma mensagem chegar antes desse intervalo expirar, a chamada falhará e retornará o código de razão MQRC_NO_MSG_AVAILABLE.

O programa demonstra como os campos *MsgId* e *CorrelId* devem ser desmarcados da estrutura MQMD após cada chamada MQGET porque a chamada configura esses campos como os valores contidos na mensagem que ela recupera. Desmarcar esses campos significa que sucessivas chamadas MQGET recuperam as mensagens na ordem em que elas são retidas na fila.

A chamada MQGET especifica um buffer de tamanho fixo. Se uma mensagem for maior do que esse buffer, a chamada falhará e o programa irá parar.

O programa continua até que a chamada MQGET retorne o código de razão MQRC_NO_MSG_AVAILABLE ou a chamada MQGET falhe. Se a chamada falhar, o programa exibirá uma mensagem de erro que contém o código de razão.

O programa, então, fecha a fila usando a chamada MQCLOSE.

Executando as amostras amqsget e amqsgetc

Cada um desses programas tem dois parâmetros:

1. O nome da fila de origem (obrigatório)
2. O nome do gerenciador de filas (opcional)

Se um gerenciador de filas não for especificado, amqsget se conecta ao gerenciador de filas padrão e amqsgetc se conecta ao gerenciador de filas identificado por uma variável de ambiente ou o arquivo de definição de canal do cliente.

Para executar esses programas, insira uma das opções a seguir:

- `amqsget myqueue qmanagername`
- `amqsgetc myqueue qmanagername`

em que myqueue é o nome da fila a partir da qual o programa receberá mensagens e qmanagername é o gerenciador de filas que tem myqueue.

Se você omitir o qmanagername, os programas assumirão o padrão ou, no caso do cliente MQI, o gerenciador de filas identificado por uma variável de ambiente ou o arquivo de definição de canal do cliente.

Programas de amostra de alta disponibilidade

Os programas de amostra de alta disponibilidade **amqsghac**, **amqsphac** e **amqsmhac** usam reconexão do cliente automatizada para demonstrar recuperação após a falha de um gerenciador de filas. **amqsfhac** verifica se um gerenciador de filas usando armazenamento em rede mantém a integridade de dados após uma falha.

Os programas **amqsghac**, **amqsphac** e **amqsmhac** são iniciados a partir da linha de comandos e podem ser usados em combinação para demonstrar a reconexão após a falha de uma instância de um gerenciador de filas de várias instâncias.

Como alternativa, também é possível usar as amostras **amqsghac**, **amqsphac** e **amqsmhac** para demonstrar reconexão do cliente a gerenciadores de filas de instância única, geralmente configurados em um grupo de gerenciadores de filas.

Para manter o exemplo simples, para facilitar a configuração, serão mostrados os programas de amostra reconectando a um gerenciador de filas de instância única iniciado, interrompido e, em seguida, reiniciado novamente; consulte "Configurar e controlar o gerenciador de filas" na página 134.

Use **amqsfhac** em paralelo com **amqmfsck** para verificar a integridade do sistema de arquivos. Consulte **amqmfsck** (verificação do sistema de arquivos) e Verificando o comportamento do sistema de arquivo compartilhado para obter mais informações...

amqsphac *queueName* [*qMgrName*]

- **amqsphac** é um aplicativo IBM WebSphere MQ MQI client . Ele coloca uma sequência de mensagens em uma fila com um atraso de dois segundos entre cada mensagem e exibe eventos enviados ao seu manipulador de eventos.
- Nenhum ponto de sincronização é usado para colocar mensagens na fila.
- A reconexão pode ser feita para qualquer gerenciador de filas no mesmo grupo de gerenciadores de filas.

amqsghac *queueName* [*qMgrName*]

- **amqsghac** é um aplicativo IBM WebSphere MQ MQI client . Ele obtém mensagens de uma fila e exibe eventos enviados ao seu manipulador de eventos.
- Nenhum ponto de sincronização é usado para obter mensagens da fila.
- A reconexão pode ser feita para qualquer gerenciador de filas no mesmo grupo de gerenciadores de filas.

amqsmhac -s *sourceQueueNome* -t *targetQueueNome* [-m *qMgrNome*] [-w *waitInterval*]

- **amqsmhac** é um aplicativo IBM WebSphere MQ MQI client . Ele copia mensagens de uma fila para outra com um intervalo de espera padrão de 15 minutos após a última mensagem recebida antes da conclusão do programa.
- As mensagens são copiadas dentro do ponto de sincronização.
- A reconexão pode ser feita somente no mesmo gerenciador de filas.

amqsfhac *QueueManagerName QueueName SideQueueName InTransactionCount RepeatCount* (0|1|2)

- **amqsfhac** é um aplicativo IBM WebSphere MQ MQI client . Ele verifica se um gerenciador de filas de várias instâncias do IBM WebSphere MQ que está usando armazenamento em rede, como um NAS ou um sistema de arquivos de cluster, mantém a integridade dos dados. Siga as etapas para executar **amqsfhac** em Verificando o comportamento do sistema de arquivo compartilhado

- Ele usa a opção MQCNO_RECONNECT_Q_MGR ao se conectar ao *QueueManagerName*. Reconecta automaticamente quando o gerenciador de filas efetua failover.
- Ele coloca *InTransactionCount***RepeatCount* mensagens persistentes em *QueueName* durante o qual você faz com que o gerenciador de filas falhe em qualquer número de vezes. **amqsfhac** reconecta-se ao gerenciador de filas toda vez e continua. O teste é para assegurar que nenhuma mensagem seja perdida.
- Mensagens *InTransactionCount* são colocadas dentro de cada transação. A transação é repetida *RepeatCount* número de vezes. Se ocorrer uma falha em uma transação, **amqsfhac** recupera e reenvia a transação quando **amqsfhac** reconectar ao gerenciador de filas.
- Ele também coloca mensagens em *SideQueueName*. Usa *SideQueueName* para verificar se o todas as mensagens estão confirmadas ou retrocedidas do *QueueName* com sucesso. Se detectar uma inconsistência, grava uma mensagem de erro.
- Varie a quantia de rastreio de saída de **amqsfhac** configurando o último parâmetro como `(0|1|2)`.

  0
  Menos saída.

  1
  Saída moderada.

  2
  Mais saída.

Configurando uma conexão do cliente

Você precisa configurar um canal de conexão do cliente e do servidor para executar as amostras. O procedimento de verificação do cliente explica como configurar um ambiente de teste do cliente. Consulte Verificando uma instalação do cliente.

Como alternativa, use a configuração fornecida no exemplo a seguir.

Exemplo de uso de amqsghac, amqsphac e amqsmhac

O exemplo demonstra clientes reconectáveis usando um gerenciador de filas de instância única.

As mensagens são colocadas na fila SOURCE por **amqsphac**, transferidas para TARGET por **amqsmhac**e recuperadas de TARGET por **amqsghac**; consulte Figura 19 na página 133

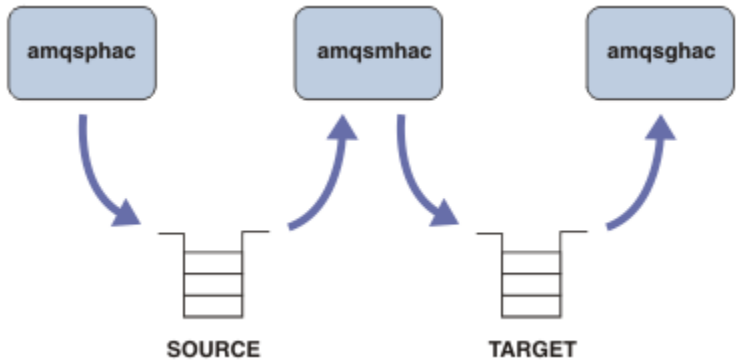


Figura 19. Amostras de clientes reconectáveis

Siga estas etapas para executar as amostras.

1. Crie um arquivo `hasamples.tst` contendo os comandos:

```
DEFINE QLOCAL(SOURCE) REPLACE
DEFINE QLOCAL(TARGET) REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)   TRPTYPE(TCP) +
       MCAUSER(MUSR_MQADMIN) REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP) +
       CONNAME('LOCALHOST(2345)') QMNAME(QM1) REPLACE
```

```
ALTER  LISTENER(SYSTEM.DEFAULT.LISTENER.TCP) TRPTYPE(TCP) +
       PORT(2345)
START  LISTENER(SYSTEM.DEFAULT.LISTENER.TCP)
START  CHANNEL(CHANNEL1)
```

2. Digite os comandos a seguir em um prompt de comandos:
   a. `crtmqm QM1`
   b. `strmqm QM1`
   c. `runmqsc QM1 < hasamples.tst`
3. Configure a variável de ambiente **MQCHLLIB** para o caminho para o arquivo de definição de canal do cliente AMQCLCHL.TAB; por exemplo, `SET MQCHLLIB=C:\IBM\MQ\MQ7\Data\qmgrs\QM1\@ipcc`.
4. Abra três novas janelas com **MQCHLLIB** configurado; por exemplo, no Windows, digite **start** três vezes no prompt de comandos anterior, iniciando cada programa em uma das portas. Consulte a etapa “5” na página 135 em “Configurar e controlar o gerenciador de filas” na página 134.)
5. Digite o comando `endmqm -r -p QM1` para parar o gerenciador de filas e, em seguida, permita que os clientes se reconectem.
6. Digite o comando `strmqm QM1` para reiniciar o gerenciador de filas.

Os resultados da execução das amostras **amqsghac**, **amqsphac**e **amqsmhac** no Windows são mostrados nos exemplos a seguir:

Configurar e controlar o gerenciador de filas

1. Crie o gerenciador de filas.

```
C:\>crtmqm QM1
WebSphere MQ queue manager created.
Directory 'C:\IBM\MQ\MQ7\Data\qmgrs\QM1' created.
Creating or replacing default objects for QM1.
Default objects statistics : 67 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

   Lembre-se do diretório de dados para configurar a variável **MQCHLLIB** posteriormente.
2. Inicie o gerenciador de filas.

```
C:\>strmqm QM1
WebSphere MQ queue manager 'QM1' starting.
5 log records accessed on queue manager 'QM1' during the log replay phase.
Log replay for queue manager 'QM1' complete.
Transaction manager state recovered for queue manager 'QM1'.
WebSphere MQ queue manager 'QM1' started.
```

3. Crie as filas e os canais, modifique a porta do listener e inicie o listener e o canal.

```
C:\>runmqsc QM1 < hasamples.tst
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM1.

     1 : DEFINE QLOCAL(SOURCE) REPLACE
AMQ8006: WebSphere MQ queue created.
     2 : DEFINE QLOCAL(TARGET) REPLACE
AMQ8006: WebSphere MQ queue created.
     3 : DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)   TRPTYPE(TCP) MCAUSER(MUSR_MQADMIN)
REPLACE
AMQ8014: WebSphere MQ channel created.
     4 : DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP) CONNAME('LOCALHOST(2345)')
QMNAME(QM1) REPLACE
AMQ8014: WebSphere MQ channel created.
     5 : ALTER LISTENER(SYSTEM.DEFAULT.LISTENER.TCP) TRPTYPE(TCP) PORT(2345)
AMQ8623: WebSphere MQ listener changed.
     6 : START LISTENER(SYSTEM.DEFAULT.LISTENER.TCP)
AMQ8021: Request to start WebSphere MQ Listener accepted.
     7 : START CHANNEL(CHANNEL1)
AMQ8018: Start WebSphere MQ channel accepted.
7 MQSC commands read.
```

```
No commands have a syntax error.
All valid MQSC commands were processed.
```

4. Torne a tabela do canal do cliente conhecida para os clientes.

   Use o diretório de dados retornado do comando **crtmqm** na etapa “1” na página 134 e inclua o diretório `@ipcc` nele para configurar a variável **MQCHLLIB**.

```
C:\>SET MQCHLLIB=C:\IBM\MQ\MQ7\Data\qmgrs\QM1\@ipcc
```

5. Iniciar os programas de amostra nas outras janelas

```
C:\>start amqsphac SOURCE QM1
C:\>start amqsmhac -s SOURCE -t TARGET -m QM1
C:\>start amqsghac TARGET QM1
```

6. Finalize o gerenciador de filas e reinicie-o novamente.

```
C:\>endmqm -r -p QM1
Waiting for queue manager 'QM1' to end.
WebSphere MQ queue manager 'QM1' ending.
WebSphere MQ queue manager 'QM1' ended.

C:\>strmqm QM1
WebSphere MQ queue manager 'QM1' starting.
5 log records accessed on queue manager 'QM1' during the log replay phase.
Log replay for queue manager 'QM1' complete.
Transaction manager state recovered for queue manager 'QM1'.
WebSphere MQ queue manager 'QM1' started.
```

amqsphac

```
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
message <Message 2>
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnectedmessage
<Message 3>
message <Message 4>
message <Message 5>
```

amqsmhac

```
Sample AMQSMHA0 start
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
No more messages.
Sample AMQSMHA0 end
C:\>
```

amqsghac

```
Sample AMQSGHAC start
message <Message 1>
message <Message 2>
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
message <Message 3>
message <Message 4>
message <Message 5>
```

Tarefas relacionadas

Verificando o comportamento do sistema de arquivo compartilhado

Referências relacionadas

amqmfsck (verificação de sistema de arquivos)

Os programas de amostra Inquire

Programas de amostra Inquire consultam sobre alguns dos atributos de uma fila usando a chamada MQINQ.

Consulte “Recursos demonstrados nos programas de amostra” na página 99 para obter os nomes desses programas.

Esses programas são projetados para execução como programas acionados, portanto única entrada é um estrutura MQTMC2 (mensagem do acionador) para sistemas IBM i, Windows, UNIX and Linux. Essa estrutura contém o nome de uma fila de destino com atributos que devem ser consultados. A versão de C também usa o nome do gerenciador de filas. A versão COBOL usa o gerenciador de filas padrão.

Para o processo de acionamento trabalhar, assegure-se de que o programa de consulta de amostra que você deseja usar seja acionado pelas mensagens que chegam na fila de SYSTEM.SAMPLE.INQ. Para fazer isso, especifique o nome do programa de amostra Inquire que desejar usar no campo *ApplicId* da definição de processo SYSTEM.SAMPLE.INQPROCESS. A fila de amostra tem um tipo de acionador FIRST; se já houver mensagens na fila antes de executar a amostra de solicitação, a amostra de consulta não será acionada pelas mensagens enviadas por você.

Quando a definição tiver sido configurada corretamente:

- Para sistemas UNIX, Linux e Windows, inicie o programa **runmqtrm** em uma sessão, em seguida, inicie o programa `amqsreq` em outra.

Use os programas de amostra Request para enviar mensagens de pedido, cada uma contendo apenas um nome de fila para a fila SYSTEM.SAMPLE.INQ. Para cada mensagem de solicitação, os programas de amostra de consulta enviarão uma mensagem de resposta contendo informações sobre a fila especificada na mensagem de pedido. As respostas são enviadas à fila de resposta especificada na mensagem de solicitação.

Design do programa de amostra de consulta

O programa abre a fila denominada na estrutura da mensagem do acionador que foi passada quando ele foi iniciado. (Por questão de clareza, chamaremos isso de *fila de solicitações*.) O programa usa a chamada MQOPEN para abrir essa fila para entrada compartilhada.

O programa usa a chamada MQGET para remover as mensagens dessa fila. Essa chamada usa as opções MQGMO_ACCEPT_TRUNCATED_MSG e MQGMO_WAIT com um intervalo de espera de 5 segundos. O programa testa o descritor de cada mensagem para ver se é uma mensagem de solicitação; se não for, o programa descartará a mensagem e exibirá uma mensagem de aviso.

Para cada mensagem de solicitação removida da fila de solicitações, o programa lê o nome da fila (que chamaremos de *fila de destino*) contida nos dados e abre essa fila usando a chamada MQOPEN com a opção MQOO_INQ. O programa, então, usa a chamada MQINQ para consultar sobre os valores dos atributos *InhibitGet*, *CurrentQDepth* e *OpenInputCount* da fila de destino.

Se a chamada MQINQ for bem-sucedida, o programa usará a chamada MQPUT1 para colocar uma mensagem de resposta na fila de resposta. Essa mensagem contém os valores dos três atributos.

Se a chamada MQOPEN ou MQINQ for mal sucedida, o programa usará a chamada MQPUT1 para colocar uma mensagem de relatório na fila de resposta. No campo *Feedback* do descritor de mensagens dessa mensagem de relatório está o código de razão retornado pela chamada MQOPEN ou MQINQ, dependendo de qual delas falhou.

Após a chamada MQINQ, o programa fecha a fila de destino usando a chamada MQCLOSE.

Quando não houver nenhuma mensagem restante na fila de solicitações, o programa fecha essa fila e desconecta do gerenciador de filas.

O programa de amostra Inquire Properties of a Message Handle

AMQSIQMA é um programa C de amostra para consultar propriedades de um identificador de mensagens a partir de uma fila de mensagens e é um exemplo de uso da chamada de API MQINQMP.

Essa amostra cria uma manipulação de mensagem e a coloca no campo MsgHandle da estrutura MQGMO. A amostra, então, obtém uma mensagem e consulta e imprime todas as propriedades com as quais a manipulação de mensagem foi preenchida.

```
C:\Program Files\IBM\WebSphere MQ\tools\c\Samples\Bin >amqsiqm Q QM1
Sample AMQSIQMA start
property name <MyProp> value <MyValue>
message text <Hello world!>
Sample AMQSIQMA end
```

Os programas de amostra Publish/Subscribe

Os programas de amostra de publicação / assinatura demonstram o uso dos recursos de publicação e assinatura no WebSphere MQ

Há três programas de amostra de linguagem C ilustrando como programar para a interface de publicação / assinatura do WebSphere MQ . Há algumas amostras C que usam interfaces antigas e há amostras Java. As amostras Java usam a interface de publicação / assinatura do WebSphere MQ em com.ibm.mq.jar e a interface de publicação / assinatura JMS em com.ibm.mqjms. As amostras JMS não são cobertas neste tópico..

C

Localize a amostra do publicador `amqspub` na pasta de amostras C. Execute-a com qualquer nome de tópico que quiser como o primeiro parâmetro, seguido por um nome do gerenciador de filas opcional. Por exemplo, `amqspub mytopic QM3`. Há também uma versão de cliente chamada `amqspubc`. Se optar por executar a versão de cliente, primeiro consulte “Preparando e executando os programas de amostra” na página 111 para obter detalhes.

O publicador se conecta ao gerenciador de filas padrão e responde com a saída, `target topic is mytopic`. Cada linha que você inserir nessa janela a partir de agora será publicada em `mytopic`

Abra outra janela de comando no mesmo diretório e execute o programa do assinante, `amqssub`, fornecendo-o com o mesmo nome de tópico e um nome de gerenciador de filas opcional. Por exemplo, `amqssub mytopic QM3`.

O assinante responde com a saída `Calling MQGET : 30 seconds wait time`. De agora em diante, as linhas que digitar no publicador aparecem na saída do assinante.

Inicie outro assinante em outra janela de comandos e observe ambos os assinantes receberem publicações.

Para a documentação completa dos parâmetros, incluindo opções de configuração, consulte o código-fonte de amostra. Os valores para o campo de opções do assinante estão descritos no tópico a seguir: Opções (MQLONG).

Há outra amostra de assinante `amqssbx`, que oferece opções de assinatura adicionais como comutadores da linha de comandos.

Digite `amqssbx -d mysub -t mytopic -k` para chamar o assinante que está usando assinaturas duráveis que são retidas após o assinante ter finalizado.

Teste a assinatura publicando outro item usando o publicador. Espere 30 segundos até o assinante finalizar. Publique mais alguns itens sob o mesmo tópico. Reinicie o assinante. O último item publicado enquanto o assinante não estava em execução será exibido pelo assinante imediatamente ao ser reiniciado.

Legado de C

Há um conjunto adicional de amostras C que demonstram os comandos enfileirados. Algumas dessas amostras foram enviadas originalmente como parte do MQ0C Supportpac. O recursos que as amostras demonstram são totalmente suportados por razões de compatibilidade.

Desaconselhamos o uso da interface de comando enfileirada. É muito mais complexa do que a API de publicar/assinar e não há razão funcional convincente para programar comandos enfileirados complexos. No entanto, você pode achar a abordagem enfileirada mais adequada, talvez porque já esteja usando a interface ou porque seu ambiente de programação facilita a construção de uma mensagem complexa e chamar uma chamada MQPUT genérica, em vez de construir diferentes chamadas para MQSUB.

As amostras adicionais estão localizadas no subdiretório `pubsub` na pasta `samples`.

Há seis tipos de amostras listados em Tabela 20 na página 138.

Tabela 20. Categorias de programas C de amostra Publish/Subscribe de legado

Categoria	Programas	Comentários
RFH1	`amqssr1a.c` `amqspr1a.c`	Exemplo simples de publicar/assinar construído usando mensagens de formato RFH1.
RFH2	`amqssr2a.c` `amqspr2a.c`	Exemplo simples de publicar/assinar construído usando mensagens de formato RFH2.
Amostras MQAI	`amqsppca.c` `amqsspca.c`	Exemplo simples de publicar/assinar construído usando os comandos PCF e a interface de comandos MQAI.
Serviço MA0C Results usando RFH1	`amqsgama.c` `amqsresa.c`	Serviço Results construído usando cabeçalhos RFH1 1. Requer as filas definidas em `amqsgama.tst` e `amqsresa.tst` 2. `amqsresa` deve ser iniciado antes de `amqsgama`
Serviço MA0C Results usando RFH2	`amqsgr2a.c` `amqsrr2a.c`	Serviço Results construído usando cabeçalhos RFH2 1. Requer as filas definidas em `amqsgama.tst` e `amqsresa.tst` 2. `amqsresa` deve ser iniciado antes de `amqsgama`
Amostra Publish/ Subscribe da saída de roteamento	`amqspsra.c`	Demonstra como mudar o destino da fila ou do gerenciador de filas para uma mensagem de publicar/assinar em uma saída de roteamento.

Java

A amostra Java `MQPubSubApiSample.java` combina publicador e assinantes em um único programa. Seus arquivos de origem e de classe compilada estão localizados na pasta de amostras `wmqjava`.

Se optar por executar no modo de cliente, primeiro consulte “Preparando e executando os programas de amostra” na página 111 para obter detalhes.

Execute a amostra da linha de comandos usando o comando Java, se você tiver um ambiente Java configurado. Também é possível executar a amostra a partir da área de trabalho do WebSphere MQ Explorer Eclipse que possui um ambiente de trabalho de programação Java já configurado..

Talvez seja necessário mudar algumas das propriedades do programa de amostra para executá-lo. Você faz isso fornecendo parâmetros para a JVM ou editando a origem.

As instruções em "Executando a amostra Java MQPubSubApiSample" na página 139 mostram como executar a amostra a partir da área de trabalho do Eclipse.

Executando a amostra Java MQPubSubApiSample

Como executar o MQPubSubApiSample usando o Java Development Tools da plataforma Eclipse .

Antes de começar

Abra o ambiente de trabalho do Eclipse. Crie um novo diretório de área de trabalho e selecione-o. Feche a janela de boas-vindas.

Siga as etapas em "Preparando e executando os programas de amostra" na página 111 antes de executar como um cliente.

Sobre esta tarefa

O programa de amostra de publicação / assinatura Java é um programa Java do cliente MQI do WebSphere MQ . A amostra é executada sem modificação usando um gerenciador de filas padrão que está atendendo na porta 1414. A tarefa descreve esse caso simples e indica em termos gerais como fornecer parâmetros e modificar a amostra para se adequar a diferentes configurações do WebSphere MQ . O exemplo é ilustrado em execução no Windows Os caminhos de arquivo serão diferentes em outras plataformas.

Procedimento

1. Importar programas de amostra Java
   a) No ambiente de trabalho, clique em **Janela** > **Abrir perspectiva** > **Outros** > **Java** e clique em **OK**
   b) Alterne para a visualização **Package Explorer**.
   c) Clique com o botão direito no espaço em branco na visualização **Package Explorer**. Clique em **Novo** > **Projeto Java**..
   d) No **Project name** tipo de campo `MQ Java Samples`. Clique em **Avançar**.
   e) No painel **Java Settings** , alterne para a guia **Bibliotecas** .
   f) Clique em **Incluir JARs externos**.
   g) Navegue para *MQ_INSTALLATION_PATH*`\java\lib` em que *MQ_INSTALLATION_PATH* é a pasta de instalação do WebSphere MQ e selecione `com.ibm.mq.jar` e `com.ibm.mq.jmqi.jar`
   h) Clique em **Abrir** > **Concluir.**
   i) Clique com o botão direito em `src` na visualização **Package Explorer**.
   j) Selecionar **Importar ...** > **Geral** > **Sistema de Arquivos** > **Avançar** > **Procurar**... e navegue até o caminho *MQ_INSTALLATION_PATH*`\tools\wmqjava\samples` , em que *MQ_INSTALLATION_PATH* é o diretório de instalação do WebSphere MQ
   k) No painel **Importar**, Figura 20 na página 140, clique em `samples` (não selecione a caixa de seleção).
   l) Selecione `MQPubSubApiSample.java`. O campo **Into folder** deve conter `MQ Java Samples/src`.. Clique em **Concluir**.

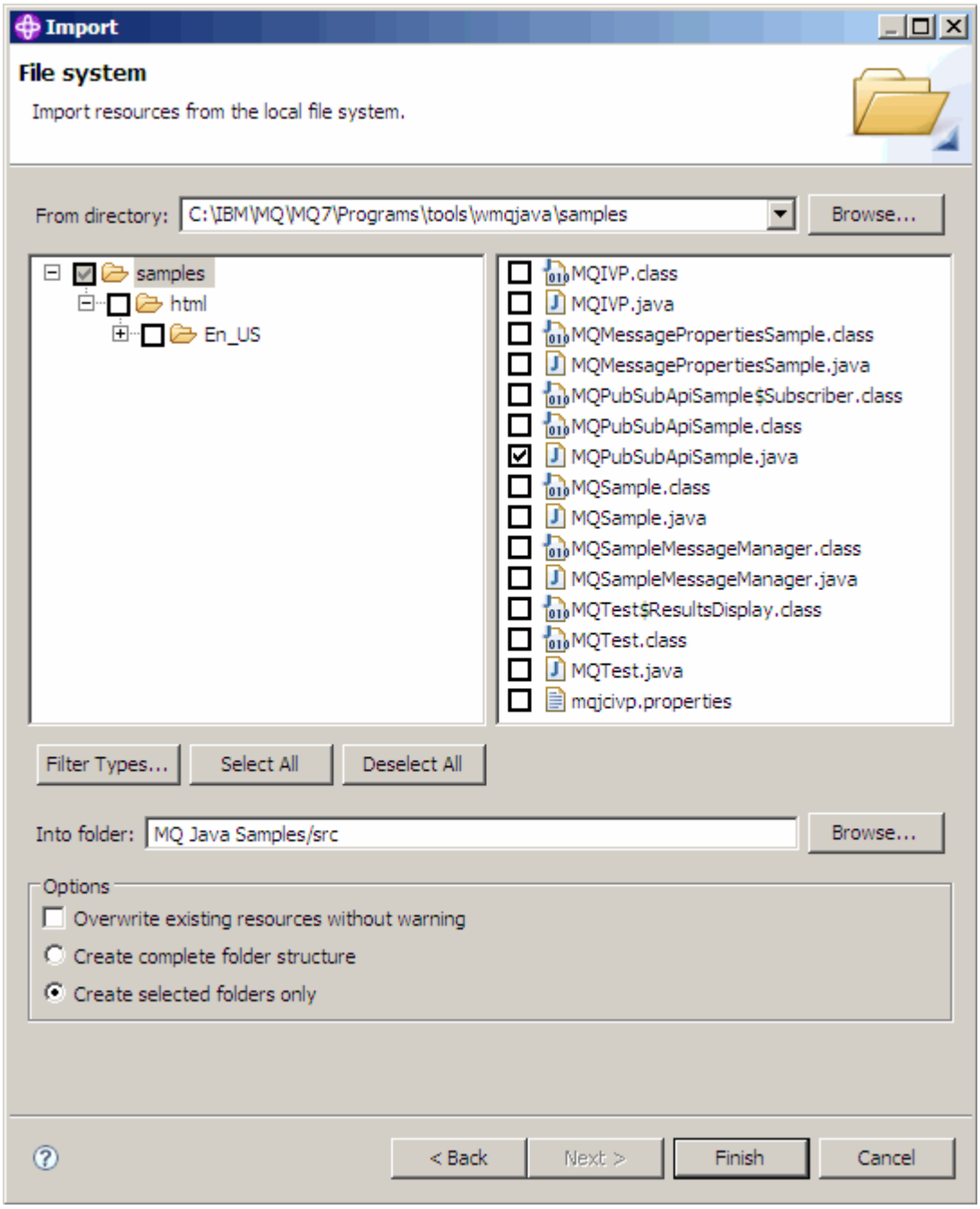


Figura 20. Importação do sistema de arquivos

2. Execute o programa de amostra de publicar/assinar.

   Há duas maneiras de executar o programa, dependendo de se você precisa mudar os parâmetros padrão.

   - A primeira opção executa o programa sem fazer nenhuma mudança:
     - No menu principal da área de trabalho, expanda a pasta `src` . Clique com o botão direito em **MQPubSubApiSample.java Executar como** > **1. Aplicativo Java**
   - A segunda opção executa o programa com parâmetros ou com código-fonte modificado para seu ambiente:
     - Abra `MQPubSubApiSample.java` e estude o construtor `MQPubSubApiSample`.

– Modifique os atributos do programa.

  Esses atributos são modificáveis usando o comutador JVM -D ou fornecendo um valor padrão para a propriedade Sistema editando o código-fonte.

  - topicObject
  - queueManagerName
  - subscriberCount

  Esses atributos podem ser mudados somente editando o código-fonte no construtor.

  - hostname
  - port
  - channel

  Para configurar as propriedades do sistema, codifique um valor padrão no acessador, por exemplo:

```
queueManagerName = System.getProperty("com.ibm.mq.pubSubSample.queueManagerName",
"QM3");
```

  Ou forneça o parâmetro para a JVM usando a opção `-D`, conforme mostrado nas etapas a seguir:

a. Copie o nome completo do System.Property que deseja configurar, por exemplo: `com.ibm.mq.pubSubSample.queueManagerName`.
b. Na área de trabalho, clique com o botão direito em **Executar** > **Abrir Diálogo de Execução**. Clique duas vezes em Aplicativo Java em **Criar, Gerenciar e Executar Aplicativos** e clique na guia **(x) = Argumentos** .
c. Na área de janela **Argumentos da VM:** , digite `-D` e cole o nome System.property , `com.ibm.mq.pubSubSample.queueManagerName`, seguido por `=QM3`. Clique em **Aplicar** > **Executar**.
d. Inclua argumentos adicionais como uma lista separada por vírgulas ou como linhas adicionais na área de janela, sem separadores de vírgula.

   Por exemplo:`-Dcom.ibm.mq.pubSubSample.queueManagerName=QM3, -Dcom.ibm.mq.pubSubSample.subscriberCount=6`.

O programa de amostra Publish Exit

AMQSPSE0 é um programa C de amostra de uma saída para interceptar uma publicação antes de ser entregue a um assinante. A saída pode então, por exemplo, alterar os cabeçalhos, a carga útil ou o destino da da mensagem ou impedir que a mensagem seja publicada a um assinante.

Para executar a amostra, execute as tarefas a seguir:

1. Configure o gerenciador de filas:
   - Nos sistemas UNIX and Linux, inclua uma sub-rotina como esta no arquivo `qm.ini`:

```
PublishSubscribe:
          PublishExitPath=<Module>
          PublishExitFunction=EntryPoint
```

   em que o módulo é *MQ_INSTALLATION_PATH*`/samp/bin/amqspse` .. *MQ_INSTALLATION_PATH* Representa o diretório de alto nível no qual o WebSphere MQ está instalado No Windows, configure os atributos equivalentes no registro
2. Certifique-se de que o Módulo esteja acessível para o WebSphere MQ
3. Reinicie o Gerenciador de filas para captar a configuração.
4. No processo do aplicativo a ser rastreado, descreva onde os arquivos de rastreio devem ser gravados. Por exemplo:

- Em sistemas UNIX and Linux , assegure-se de que o diretório `/var/mqm/trace` exista e exporte a variável de ambiente a seguir:

```
export MQPSE_TRACE_LOGFILE=/var/mqm/trace/PubTrace
```

- No Windows, assegure-se de que o diretório `C:\temp` exista e configure a variável de ambiente a seguir:

```
set MQPSE_TRACE_LOGFILE=C:\temp\PubTrace
```

Os programas de amostra Put

Os programas de amostra Put colocam mensagens em uma fila usando a chamada MQPUT.

Consulte "Recursos demonstrados nos programas de amostra" na página 99 para obter os nomes desses programas.

Design do programa de amostra Put

O programa usa a chamada MQOPEN com a opção MQOO_OUTPUT para abrir a fila de destino para colocar mensagens.

Se ele não puder abrir a fila, o programa envia uma mensagem de erro contendo o código de razão retornado pela chamada MQOPEN. Para manter o programa simples, nesta e em chamadas MQI subsequentes, o programa usa valores padrão para muitas das opções.

Para cada linha de entrada, o programa lê o texto em um buffer e usa a chamada MQPUT para criar uma mensagem de datagrama que contém o texto dessa linha. O programa continua até que ele atinja o final da entrada ou a chamada MQPUT falhará. Se o programa atingir o final da entrada, ele fechará a fila usando a chamada MQCLOSE.

Executando os programas de amostra Put

Executando as amostras amqsput e amqsputc

Cada um desses programas possui 2 parâmetros:

1. O nome da fila de destino (obrigatório)
2. O nome do gerenciador de filas (opcional)

Se um gerenciador de filas não for especificado, amqsput se conectará ao gerenciador de filas padrão e amqsputc se conectará ao gerenciador de filas identificado por uma variável de ambiente ou o arquivo de definição de canal do cliente. Para executar esses programas, insira uma das opções a seguir:

- `amqsput myqueue qmanagername`
- `amqsputc myqueue qmanagername`

em que `myqueue` é o nome da fila na qual as mensagens serão colocadas e `qmanagername` é o gerenciador de filas que possui `myqueue`.

Executando a amostra amq0put

A versão em COBOL não tem nenhum parâmetro. Ela se conecta ao gerenciador de filas padrão e quando executá-la, será solicitado o seguinte:

```
Please enter the name of the target queue
```

Ela obtém entrada de StdIn e inclui cada linha de entrada na fila de destino. Uma linha em branco indica que não há mais dados.

Os programas de amostra Reference Message

As amostras de Reference Message permitem que um objeto grande seja transferido de um nó para outro (geralmente em sistemas diferentes) sem a necessidade de o objeto ser armazenado nas filas do WebSphere MQ nos nós de origem ou de destino.

Um conjunto de programas de amostra é fornecido para demonstrar como Reference Messages pode ser colocado em uma fila, recebido por saídas de mensagens e obtido de uma fila. Os programas de amostra usam Reference Messages para mover arquivos. Se desejar mover outros objetos como bancos de dados ou se desejar executar verificações de segurança, defina sua própria saída, com base na nossa amostra, amqsxrm. As seções a seguir descrevem os programas de amostra Reference Message.

A versão do programa de amostra de saída Reference Message a ser usada depende da plataforma na qual o canal está em execução. Em todas as plataformas, use amqsxrma na extremidade de envio. Use amqsxrma na extremidade de recebimento se o receptor estiver em execução em qualquer produto WebSphere MQ , exceto WebSphere MQ para IBM i

Executando as amostras Reference Message

Use estas informações para aprender como executar os programas de amostra Reference Message.

As amostras Reference Message são executadas da seguinte forma:

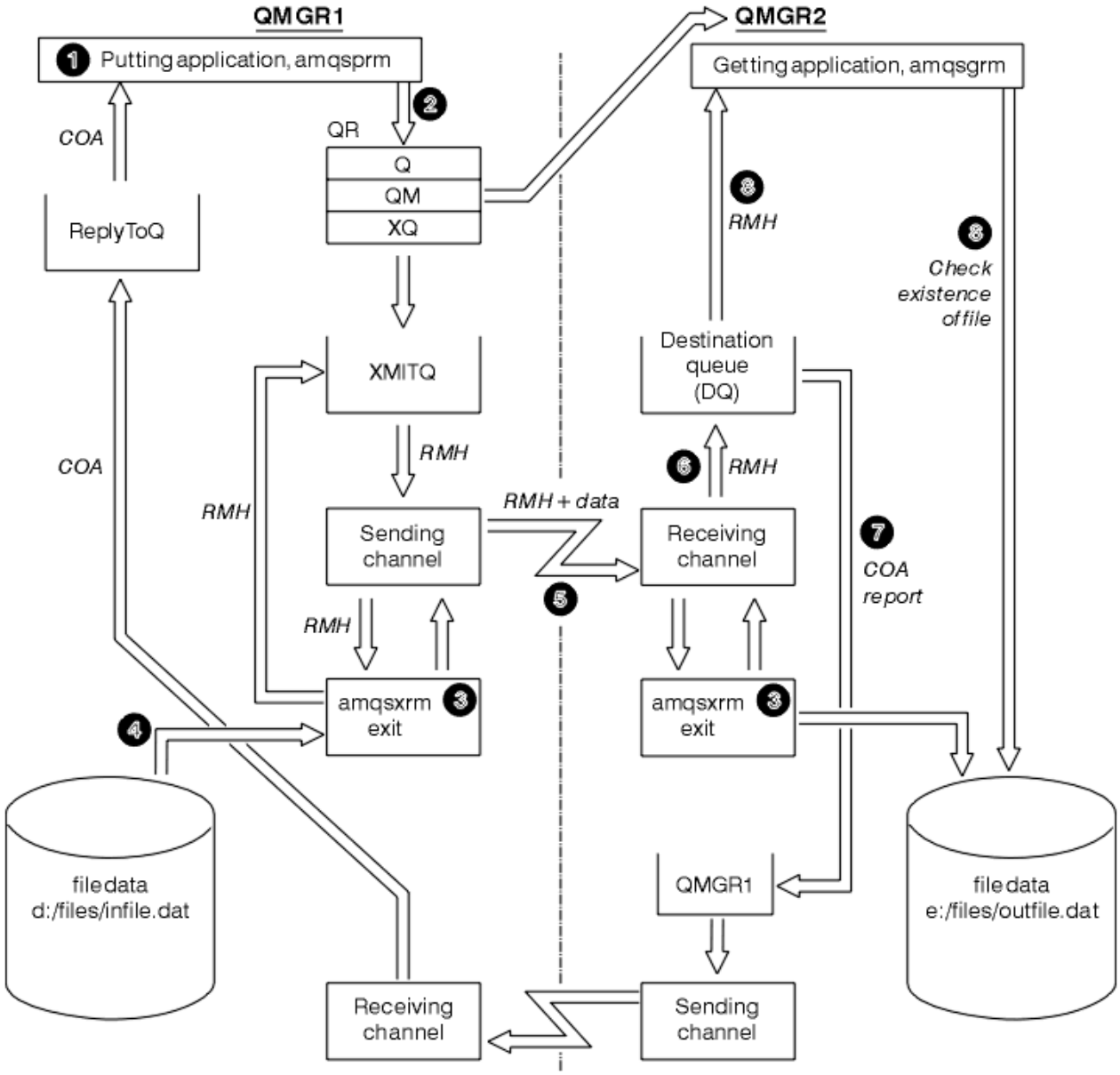


Figura 21. Executando as amostras Reference Message

1. Configure o ambiente para iniciar os listeners, canais e monitores acionadores e definir os canais e filas.

   Para os propósitos de descrever como configurar o exemplo de Reference Message, o presente refere-se à máquina de envio como MACHINE1 com um gerenciador de filas chamado QMGR1 e a máquina de destino como MACHINE2 com um gerenciador de filas chamado QMGR2.

   Nota: As definições a seguir permitem que uma Reference Message seja construída para enviar um arquivo com um tipo de objeto FLATFILE do gerenciador de filas QMGR1 para o QMGR2 e recriar o arquivo, conforme definido na chamada para AMQSPRM (ou AMQSPRMA no IBM i). A Reference Message (incluindo o arquivo de dados) é enviada usando canal CHL1 e a fila de transmissão XMITQ e é colocada na fila DQ. Relatórios de exceções e COA são enviados de volta a QMGR1 usando o canal REPORT e a fila de transmissão QMGR1.

   O aplicativo que recebe o Reference Message (AMQSGRM) é acionado usando a fila de inicialização INITQ e o processo PROC. Assegure-se de que os campos CONNAME estejam configurados corretamente e o campo MSGEXIT reflita sua estrutura de diretório, dependendo do tipo de máquina e de onde o produto WebSphere MQ está instalado.

As definições MQSC usaram um estilo AIX para definir as saídas. É importante observar que os dados da mensagem FLATFILE fazem distinção entre maiúsculas e minúsculas e a amostra não funcionará a menos que esteja em maiúsculas.

Na máquina MACHINE1, gerenciador de filas QMGR1

Sintaxe do MQSC

```
define chl(chl1) chltype(sdr) trptype(tcp) conname('machine2') xmitq(xmitq)
msgdata(FLATFILE) msgexit('/usr/lpp/mqm/samp/bin/amqsxrm(MsgExit)
')

define ql(xmitq) usage(xmitq)

define chl(report) chltype(rcvr) trptype(tcp) replace

define qr(qr) rname(dq) rqmname(qmgr2) xmitq(xmitq) replace
```

Nota: Se você não especificar um nome de gerenciador de filas, o sistema usa o gerenciador de filas padrão.

```
CRTMQMCHL  CHLNAME(CHL1) CHLTYPE(*SDR) MQMNAME(QMGR1) +
             REPLACE(*YES) TRPTYPE(*TCP) +
             CONNAME('MACHINE2(60501)') TMQNAME(XMITQ) +
             MSGEXIT(QMQM/AMQSXRM4) MSGUSRDATA(FLATFILE)

CRTMQMQ    QNAME(XMITQ) QTYPE(*LCL) MQMNAME(QMGR1) +
             REPLACE(*YES) USAGE(*TMQ)

CRTMQMCHL  CHLNAME(REPORT) CHLTYPE(*RCVR) +
               MQMNAME(QMGR1) REPLACE(*YES) TRPTYPE(*TCP)

CRTMQMQ    QNAME(QR) QTYPE(*RMT) MQMNAME(QMGR1) +
             REPLACE(*YES) RMTQNAME(DQ) +
             RMTMQMNAME(QMGR2) TMQNAME(XMITQ)
```

Na máquina MACHINE2, gerenciador de filas QMGR2

Sintaxe do MQSC

```
define chl(chl1) chltype(rcvr) trptype(tcp)
msgexit('/usr/lpp/mqm/samp/bin/amqsxrm(MsgExit)')
        msgdata(flatfile)

define chl(report) chltype(sdr) trptype(tcp) conname('MACHINE1')
        xmitq(qmgr1)

define ql(initq)

define ql(qmgr1) usage(xmitq)

define pro(proc) applicid('/usr/lpp/mqm/samp/bin/amqsgrm')

define ql(dq) initq(initq) process(proc) trigger trigtype(first)
```

2. Quando os objetos do WebSphere MQ tiverem sido criados:
   a. Quando aplicável para a plataforma, inicie o listener para os gerenciadores de filas de envio e de recebimento
   b. Inicie os canais CHL1 e REPORT
   c. No gerenciador de filas de recebimento, inicie o monitor acionador para a fila de inicialização INITQ
3. Chame o programa de amostra Put Reference Message AMQSPRM a partir da linha de comandos usando os parâmetros a seguir:

   -m Nome do gerenciador de filas local; o padrão usado é o gerenciador de filas padrão

   -i Nome e local do arquivo de origem

   -o Nome e local do arquivo de destino

-q Nome da fila

-g Nome do gerenciador de filas no qual a fila, definida no parâmetro -q existe. O padrão usado é o gerenciador de filas especificado no parâmetro -m

-t Tipo de Objeto

-w Intervalo de espera, ou seja, o tempo de espera para relatórios de exceções e COA do gerenciador de filas de recebimento

Por exemplo, para usar a amostra com os objetos definidos anteriormente, você usaria os parâmetros a seguir:

```
-mQMGR1 -iInput File -oOutput File -qQR -tFLATFILE -w120
```

O aumento do tempo de espera concede tempo para que um arquivo grande seja enviado por uma rede antes que o programa que está colocando as mensagens atinja o tempo limite.

```
amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
```

Nota: Para as plataformas UNIX and Linux, deve-se usar duas barras invertidas (\\) em vez de uma para indicar o diretório do arquivo de destino. Portanto, o comando **amqsprm** é semelhante a este:

```
amqsprm -i /files/infile.dat -o e:\\files\\outfile.dat -q QR
-m QMGR1 -w 30 -t FLATFILE
```

Executar o programa Put Reference Message faz o seguinte:

- A Reference Message é colocada na fila QR no gerenciador de filas QMGR1.
- O arquivo de origem e o caminho são `d:\files\infile.dat` e existem no sistema no qual o comando de exemplo é emitido.
- Se a fila QR for uma fila remota, a Reference Message será enviada para outro gerenciador de filas, em um sistema diferente, no qual um arquivo é criado com o nome e o caminho `e:\files\outfile.dat`. Os conteúdos desse arquivo são os mesmos que do arquivo de origem.
- amqsprm espera 30 segundos por um relatório de COA do gerenciador de filas de destino.
- O tipo de objeto é `flatfile`, portanto, o canal usado para mover as mensagens da fila QR deve especificar isso no campo *MsgData*.

4. Ao definir seus canais, selecione a saída de mensagem em ambas as extremidades de envio e de recebimento para ser amqsxrm. Isso é definido no WebSphere MQ para Windows como a seguir:

```
msgexit('pathname\amqsxrm.dll(MsgExit)')
```

Isso é definido no WebSphere MQ para AIX, WebSphere MQ para HP-UXe WebSphere MQ para Solaris conforme a seguir:

```
msgexit('pathname/amqsxrm(MsgExit)')
```

Se você especificar um nome de caminho, especifique o nome completo. Se você omitir o nome do caminho, será assumido que o programa está no caminho especificado no arquivo `qm.ini` (ou, no WebSphere MQ para Windows, o caminho especificado no registro).

5. A saída do canal lê o da Reference Message e localiza o arquivo ao qual ele se refere.
6. A saída do canal pode então segmentar o arquivo antes de enviá-lo para o canal juntamente com o cabeçalho. No WebSphere MQ para AIX, no WebSphere MQ para HP-UXe WebSphere MQ para Solaris, altere o proprietário do grupo do diretório de destino para 'mqm' para que a saída de mensagem de amostra possa criar o arquivo nesse diretório. Além disso, mude as permissões do diretório de destino

para permitir que membros do grupo mqm gravem nele. Os dados do arquivo não são armazenados no WebSphere MQ filas.

7. Quando o último segmento do arquivo for processado pela saída de mensagem de recebimento, a Reference Message será colocada na fila de destino especificada por amqsprm. Se essa fila for acionada (ou seja, a definição especificar os atributos de fila *Trigger*, *InitQ* e *Process*), o programa especificado pelo parâmetro PROC da fila de destino será acionado. O programa a ser acionado deve ser definido no campo *ApplId* do atributo *Process*.
8. Quando a Reference Message atingir a fila de destino (DQ), um relatório de COA será enviado de volta ao aplicativo de put (amqsprm).
9. A amostra Get Reference Message, amqsgrm, obtém mensagens da fila especificada na mensagem do acionador de entrada e verifica a existência do arquivo.

Design da amostra Put Reference Message (amqsprma.c, AMQSPRM4)

Este tópico fornece uma descrição detalhada de uma amostra Put Reference Message.

Essa amostra cria uma Reference Message que refere-se a um arquivo e a coloca em uma fila especificada:

1. A amostra se conecta a um gerenciador de filas locais usando MQCONN.
2. Em seguida, abre (MQOPEN) uma fila modelo que é usada para receber mensagens de relatório.
3. A amostra constrói uma Reference Message que contém os valores necessários para mover o arquivo, por exemplo, os nomes dos arquivos de origem e de destino e o tipo de objeto. Como um exemplo, a amostra enviada com o WebSphere MQ constrói uma Reference Message para enviar o arquivo `d:\x\file.in` de `QMGR1` para `QMGR2` e para recriar o arquivo como `d:\y\file.out` usando os parâmetros a seguir:

   ```
   amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
   ```

   Em que `QR` é uma definição de fila remota que refere-se a uma fila de destino em `QMGR2`.

   Nota: Para as plataformas UNIX and Linux, use duas barras invertidas (\\) em vez de uma para denotar o diretório do arquivo de destino. Portanto, o comando **amqsprm** é semelhante a este:

   ```
   amqsprm -q QR -m QMGR1 -i /x/file.in -o d:\\y\\file.out -t FLATFILE
   ```

4. A Reference Message é colocada (sem nenhum dado de arquivo) na fila especificada pelo parâmetro /q. Se essa for uma fila remota, a mensagem será colocada na fila de transmissão correspondente.
5. A amostra espera o período de tempo especificado no parâmetro /w (que usa como padrão 15 segundos), para relatórios COA, que, juntamente com os relatórios de exceções, são enviados de volta à fila dinâmica criada no gerenciador de filas locais (QMGR1).

Design da amostra Reference Message Exit (amqsxrma.c, AMQSXRM4)

Esta amostra reconhece Mensagens de Referência com um tipo de objeto que corresponde ao tipo de objeto no campo de dados do usuário de saída de mensagem da definição do canal.

Para essas mensagens, ocorre o seguinte:

- No canal emissor ou do servidor, o comprimento especificado de dados é copiado do deslocamento especificado do arquivo especificado para o espaço restante no buffer do agente após a Reference Message. Se o fim do arquivo não for atingido, a Reference Message é colocada de volta na fila de transmissão após atualizar o campo *DataLogicalOffset*.
- No canal solicitante ou receptor, se o campo *DataLogicalOffset* for zero e o arquivo especificado não existir, ele será criado. Os dados que seguem a Reference Message são incluídos no final do arquivo especificado. Se a Reference Message não for a última do arquivo especificado, será descartada. Caso contrário, será retornada à saída de canal, sem os dados anexados, para ser colocada na fila de destino.

Para canais emissor e do servidor, se o campo *DataLogicalLength* na Message Reference de entrada for zero, a parte restante do arquivo, de *DataLogicalOffset* até o fim do arquivo, deve ser enviada ao longo do canal. Se não for zero, somente o comprimento especificado será enviado.

Se ocorrer um erro (por exemplo, se a amostra não puder abrir um arquivo), MQCXP.*ExitResponse* será configurado para MQXCC_SUPPRESS_FUNCTION de forma que a mensagem sendo processada será colocada na fila de mensagens não entregues em vez de continuar para a fila de destino. Um código de feedback é retornado em MQCXP.*Feedback* e retornado ao aplicativo, que coloca a mensagem no campo *Feedback* do descritor de mensagens de uma mensagem de relatório. Isso ocorre porque o aplicativo de put solicitou relatórios de exceção configurando MQRO_EXCEPTION no campo *Report* do MQMD.

Se a codificação ou *CodedCharacterSetId* (CCSID) de Reference Message for diferente daquela do gerenciador de filas, Reference Message será convertida para a codificação local e o CCSID. Em nossa amostra, amqsprm, o formato do objeto é MQFMT_STRING, portanto, amqsxrm converte os dados do objeto para o CCSID local na extremidade de recebimento antes que os dados sejam gravados no arquivo.

Não especifique o formato do arquivo que está sendo transferido como MQFMT_STRING se o arquivo contiver caracteres de multibyte (por exemplo, DBCS ou Unicode). Isso ocorre porque um caractere de multibyte pode ser dividido quando o arquivo for segmentado na extremidade de envio. Para transferir e converter esse arquivo, especifique o formato como algo diferente de MQFMT_STRING de forma que a saída de Reference Message não o converta e converta o arquivo na extremidade de recebimento quando a transferência for concluída.

Compilando a amostra Reference Message Exit

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

Para compilar amqsxrma, use os seguintes comandos:

No AIX

```
xlc_r -q64 -e MsgExit -bE:amqsxrm.exp -bM:SRE -o amqsxrm_64_r
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -lmqm_r amqsqrma.c
```

no HP-UX

```
$ c89 +DD64 +z -c -D_HPUX_SOURCE -o amqsxrma.o amqsqrma.c -IMQ_INSTALLATION_PATH/inc
$ ld -b amqsxrma.o -o /var/mqm/exits64/amqsxrma -LMQ_INSTALLATION_PATH/lib64
-L/usr/lib/pa20_64 -lmqm_r -lpthread
```

EmLinux

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsxrma amqsqrma.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm_r
```

No Solaris

```
$ cc -xarch=v9 -mt -G -o /var/mqm/exits64/amqsxrma amqsqrma.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm

 -lsocket
-lnsl -ldl
```

No Windows

WebSphere MQ agora fornece à biblioteca mqm pacotes do cliente, bem como pacotes do servidor, portanto, o exemplo a seguir usa `mqm.lib` em vez de `mqmvx.lib`:

```
cl amqsqrma.c /link /out:amqsxrm.dll /dll mqm.lib mqm.lib /def:amqsxrm.def
```

Para obter informações gerais sobre como gravar e compilar saídas de canal, consulte “Gravando programas de saída do canal” na página 405

Design da amostra Get Reference Message (amqsgrma.c, AMQSGRM4)

Este tópico explica o design da amostra Get Reference Message.

A lógica do programa é a seguinte:

1. A amostra é acionada e extrai os nomes da fila e do gerenciador de filas da mensagem do acionador de entrada.
2. Em seguida, conecta-se ao gerenciador de filas especificado usando MQCONN e abre a fila especificada usando MQOPEN.
3. A amostra emite MQGET com um intervalo de espera de 15 segundos em um loop para obter mensagens da fila.
4. Se uma mensagem for uma Reference Message, a amostra verifica a existência do arquivo que foi transferido.
5. Em seguida, fecha a fila e desconecta do gerenciador de filas.

Os programas de amostra Request

Os programas de amostra Request demonstram o processamento do cliente/servidor. As amostras são os clientes que colocam mensagens de solicitação em uma fila do servidor de destino que é processado por um programa do servidor. Eles aguardam o programa do servidor para colocar uma mensagem de resposta em uma fila de responder para.

As amostras Request colocam uma série de mensagens de solicitação na fila do servidor de destino usando a chamada MQPUT. Essas mensagens especificam a fila local, SYSTEM.SAMPLE.REPLY, como a fila de resposta, que pode ser uma fila local ou remota. Os programas aguardam mensagens de resposta e, em seguida, exibem-nas. As respostas serão enviadas apenas se a fila do servidor de destino estiver sendo processada por um aplicativo do servidor ou se um aplicativo for acionado para esse propósito (os programas de amostra Inquire, Set e Echo são projetados para serem acionados). A amostra C aguarda 1 minuto (a amostra COBOL aguarda 5 minutos) pela chegada da primeira resposta (para permitir que o tempo para um aplicativo do servidor seja acionado) e 15 segundos por respostas subsequentes, mas ambas as amostras podem ser encerradas sem obter nenhuma resposta. Consulte “Recursos demonstrados nos programas de amostra” na página 99 para obter os nomes dos programas de amostra de Request.

Executando os programas de amostra Request

Executando as amostras amqsreq0.c, amqsreq e amqsreqc

A versão de C do programa aceita três parâmetros:

1. O nome da fila do servidor de destino (necessário)
2. O nome do gerenciador de filas (opcional)
3. A fila de resposta (opcional)

Por exemplo, insira um dos seguintes:

- `amqsreq myqueue qmanagername replyqueue`
- `amqsreqc myqueue qmanagername`
- `amq0req0 myqueue`

em que `myqueue` é o nome da fila do servidor de destino, `qmanagername` é o nome do gerenciador de filas que possui `myqueue` e `replyqueue` é o nome da fila de resposta.

Se você omitir o nome do gerenciador de filas, supõe-se que o gerenciador de filas padrão possui a fila. Se omitir o nome da fila de resposta, a fila de resposta padrão será fornecida.

Executando a amostra amq0req0.cbl

A versão em COBOL não tem nenhum parâmetro. Ela se conecta ao gerenciador de filas padrão e quando executá-la, será solicitado o seguinte:

```
Please enter the name of the target server queue
```

O programa usa sua entrada de StdIn e inclui cada linha na fila de servidor de destino, tendo cada linha de texto como o conteúdo de uma mensagem de solicitação. O programa termina quando uma linha nula é lida.

Executando a amostra AMQSREQ4

O programa C cria mensagens obtendo dados de stdin (o teclado) com uma entrada de finalização de tempo em branco. O programa aceita até três parâmetros: o nome da fila de destino (obrigatório), o nome do gerenciador de filas (opcional) e o nome da fila de resposta (opcional). Se nenhum nome de gerenciador de filas for especificado, o gerenciador de filas padrão será usado. Se nenhuma fila de resposta for especificada, a fila SYSTEM.SAMPLE.REPLY será usada.

Aqui está um exemplo de como chamar o programa de amostra de C, especificando a fila de resposta, mas deixando o gerenciador de filas padrão:

```
CALL PGM(QMQM/AMQSREQ4) PARM('SYSTEM.SAMPLE.LOCAL' '' 'SYSTEM.SAMPLE.REPLY')
```

Nota: Lembre-se de que os nomes de filas fazem distinção entre maiúsculas e minúsculas. Todas as filas criadas pelo programa de criação de arquivo de amostra AMQSAMP4 têm nomes criados em caracteres maiúsculos.

Executando a amostra AMQ0REQ4

O programa em COBOL cria mensagens, aceitando dados do teclado. Para iniciar o programa, chame o programa e especifique o nome da fila de destino como um parâmetro. O programa aceita entrada do teclado em um buffer e cria uma mensagem de solicitação para cada linha de texto. O programa para quando você insere uma linha em branco no teclado.

Executando a amostra Request usando o acionamento

Se a amostra for usada com o acionamento e um dos programas de amostra Inquire, Set ou Echo, a linha de entrada deverá ser o nome da fila que você deseja que o programa acionado acesse.

Sistemas UNIX, Linux e Windows

Para executar as amostras usando acionamento:

1. Inicie o programa do monitor acionador em uma sessão RUNMQTRM (a fila de inicialização SYSTEM.SAMPLE.TRIGGER está disponível para uso).
2. Inicie o programa amqsreq em outra sessão.
3. Certifique-se de que tenha definido uma fila do servidor de destino.

   As filas de amostra disponíveis para uso como a fila do servidor de destino para a amostra de solicitação nas quais colocar mensagens são:

   - SYSTEM.SAMPLE.INQ - para o programa de amostra Inquire
   - SYSTEM.SAMPLE.SET - para o programa de amostra Set
   - SYSTEM.SAMPLE.ECHO - para o programa de amostra Echo

   Essas filas têm um tipo de acionador FIRST, portanto, se já houver mensagens nas filas antes de você executar a amostra Request, os aplicativos do servidor não serão acionados pelas mensagens que você enviar.

4. Certifique-se de que tenha definido uma fila para o programa de amostra Inquire, Set ou Echo para uso.

Isso significa que o monitor acionador está pronto quando a amostra de solicitação envia uma mensagem.

Nota: As definições de processo de amostra criadas usando RUNMQSC e o arquivo amqscos0.tst acionam as amostras de C. Mude as definições de processo em amqscos0.tst e use RUNMQSC com este arquivo atualizado para usar versões em COBOL.

Figura 22 na página 151 demonstra como usar as amostras Request e Inquire juntas.

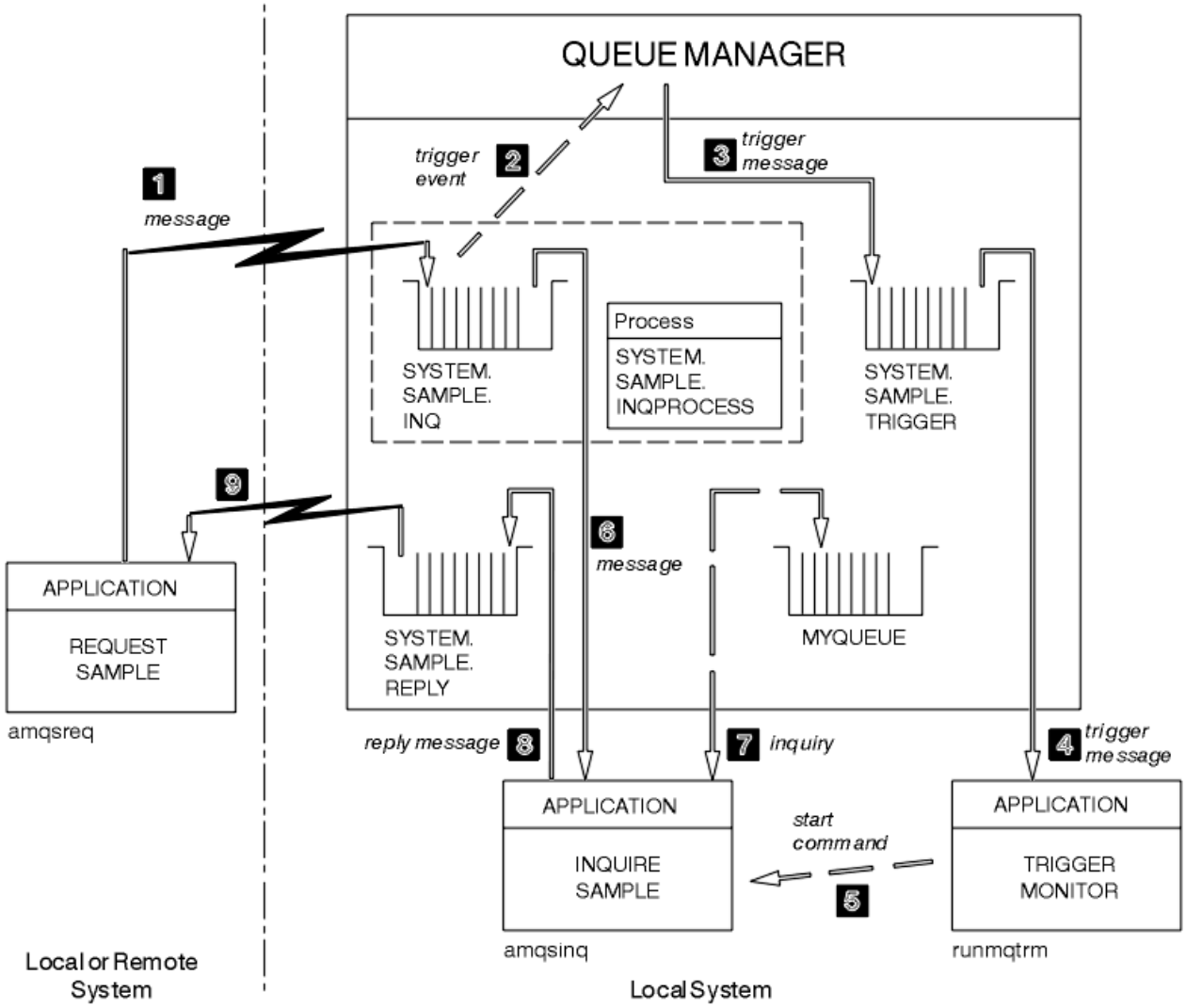


Figura 22. Amostras Request e Inquire usando acionamento

Em Figura 22 na página 151, a amostra Request coloca mensagens na fila do servidor de destino, SYSTEM.SAMPLE.INQ e a amostra Inquire consulta a fila, MYQUEUE. Como alternativa, é possível usar uma das filas de amostra definidas quando executou amqscos0.tst ou qualquer outra fila que tenha definido para a amostra Inquire.

Nota: Os números em Figura 22 na página 151 mostram a sequência de eventos.

Para executar as amostras Request e Inquire usando acionamento:

1. Verifique se as filas que deseja usar estão definidas. Execute amqscos0.tst para definir as filas de amostra e definir uma fila MYQUEUE.
2. Execute o comando do monitor acionador RUNMQTRM:

```
RUNMQTRM -m qmanagername -q SYSTEM.SAMPLE.TRIGGER
```

3. Execute a amostra Request

```
amqsreq SYSTEM.SAMPLE.INQ
```

Nota: O objeto do processo define o que deve ser acionado. Se o cliente e o servidor não estiverem em execução na mesma plataforma, qualquer processo iniciado pelo monitor acionador deverá definir *ApplType*, caso contrário, o servidor usará suas definições padrão (ou seja, o tipo de aplicativo que está normalmente associado à máquina servidor) e causará uma falha.

Para obter uma lista de tipos de aplicativos, consulte ApplType.

4. Insira o nome da fila que deseja que a amostra Inquire use:

```
MYQUEUE
```

5. Insira uma linha em branco (para finalizar o programa Request).
6. A amostra Request exibirá, então, exibir uma mensagem contendo os dados do programa Inquire obtidos a partir de MYQUEUE.

É possível usar mais de uma fila; neste caso, insira os nomes das outras filas na etapa “4” na página 152.

Para obter mais informações sobre acionamento, consulte “Iniciando aplicativos IBM WebSphere MQ usando acionadores” na página 333.

Design do programa de amostra Request

O programa abre a fila do servidor de destino de forma que possa colocar mensagens. Ele usa a chamada MQOPEN com a opção MQOO_OUTPUT. Se não for possível abrir a fila, o programa exibe uma mensagem de erro contendo o código de razão retornado pela chamada MQOPEN.

O programa abre, então, a fila de resposta chamada SYSTEM.SAMPLE.REPLY para que possa receber mensagens de resposta. Para isso, o programa usa a chamada MQOPEN com a opção MQOO_INPUT_EXCLUSIVE. Se não puder abrir a fila, o programa exibirá uma mensagem de erro contendo o código de razão retornado pela chamada MQOPEN.

Para cada linha de entrada, o programa então lê o texto em um buffer e usa a chamada MQPUT para criar uma mensagem de solicitação que contém o texto dessa linha. Nessa chamado, o programa usa a opção de relatório MQRO_EXCEPTION_WITH_DATA para solicitar que quaisquer mensagens de relatório enviadas sobre a mensagem de solicitação incluirão os primeiros 100 bytes dos dados da mensagem. O programa continua até que ele atinja o final da entrada ou a chamada MQPUT falhará.

O programa usa, então, a chamada MQGET para remover mensagens de resposta da fila e exibe os dados contidos nas respostas. A chamada MQGET usa as opções MQGMO_WAIT, MQGMO_CONVERT e MQGMO_ACCEPT_TRUNCATED. O *WaitInterval* é de 5 minutos na versão em COBOL e de 1 minuto na versão em C, para a primeira resposta (para conceder tempo para que um aplicativo do servidor seja acionado) e 15 segundos para as respostas subsequentes. O programa espera esses períodos se não houver nenhuma mensagem na fila. Se nenhuma mensagem chegar antes desse intervalo expirar, a chamada falhará e retornará o código de razão MQRC_NO_MSG_AVAILABLE. A chamada também usa a opção MQGMO_ACCEPT_TRUNCATED_MSG de forma que as mensagens mais longas do que o tamanho do buffer declarado sejam truncadas.

O programa demonstra como limpar os campos *MsgId* e *CorrelId* da estrutura MQMD após cada chamada MQGET porque a chamada configura esses campos para os valores contidos na mensagem que ela recupera. Desmarcar esses campos significa que sucessivas chamadas MQGET recuperam as mensagens na ordem em que elas são retidas na fila.

O programa continua até que a chamada MQGET retorne o código de razão MQRC_NO_MSG_AVAILABLE ou a chamada MQGET falhe. Se a chamada falhar, o programa exibirá uma mensagem de erro que contém o código de razão.

O programa fecha, então, a fila do servidor de destino e a fila de resposta usando a chamada MQCLOSE.

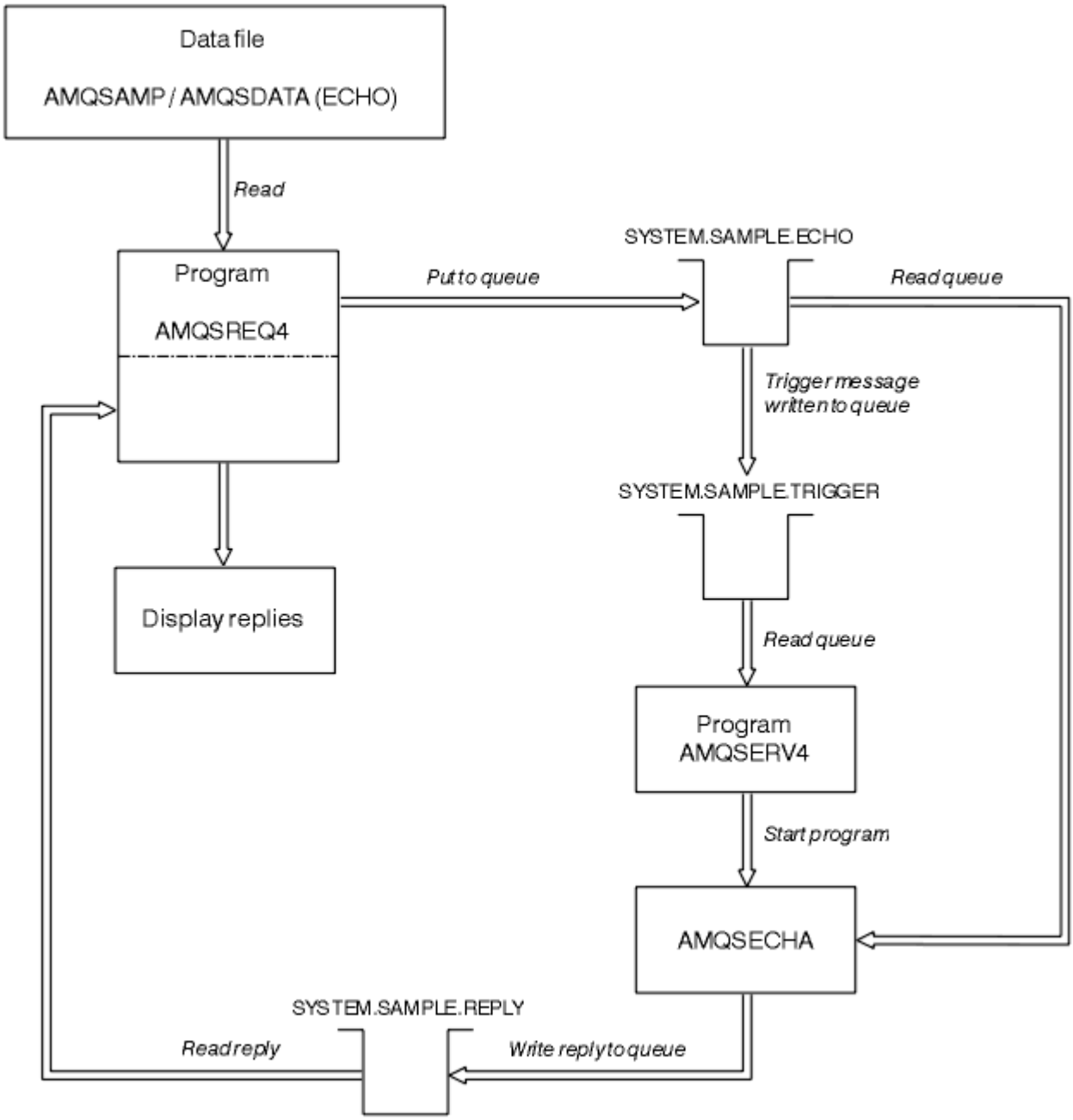


Figura 23. Fluxograma do programa IBM i Cliente/Server (Echo) de amostra

Os programas de amostra Set

Os programas de amostra Set inibem operações put em uma fila usando a chamada MQSET para mudar o atributo *InhibitPut* da fila. Além disso, aprenda sobre o design dos programas de amostra Set.

Consulte “Recursos demonstrados nos programas de amostra” na página 99 para obter os nomes desses programas.

Os programas são destinados a serem executados como programas acionados, portanto, sua única entrada é uma estrutura MQTMC2 (mensagem do acionador) que contém o nome de uma fila de destino com atributos que devem ser consultados. A versão de C também usa o nome do gerenciador de filas. A versão COBOL usa o gerenciador de filas padrão.

Para o processo de acionamento funcionar, assegure que o programa de amostra Set que deseja usar seja acionado por mensagens que chegam na fila SYSTEM.SAMPLE.SET. Para fazer isso, especifique o nome do programa de amostra Set que deseja usar no campo *ApplicId* da definição de processo SYSTEM.SAMPLE.SETPROCESS. A fila de amostra tem um tipo de acionador FIRST; se já houver mensagens na fila antes da execução da amostra Request, a amostra Set não será acionada pelas mensagens enviadas.

Quando a definição tiver sido configurada corretamente:

- Para sistemas UNIX, Linux e Windows, inicie o programa **runmqtrm** em uma sessão, em seguida, inicie o programa amqsreq em outra.
- Para o IBM i, inicie o programa AMQSERV4 em uma sessão, em seguida, inicie o programa AMQSREQ4 em outra. Seria possível usar AMQSTRG4 em vez de AMQSERV4, mas os potenciais atrasos de envio de tarefa poderiam tornar mais difícil seguir o que está acontecendo.

Use os programas de amostra Request para enviar mensagens de solicitação, cada uma contendo apenas um nome de fila, para a fila SYSTEM.SAMPLE.SET. Para cada mensagem de solicitação, os programas de amostra Set enviam uma mensagem de resposta que contém uma confirmação de que as operações put foram inibidas na fila especificada. As respostas são enviadas à fila de resposta especificada na mensagem de solicitação.

Design do programa de amostra Set

O programa abre a fila denominada na estrutura da mensagem do acionador que foi passada quando ele foi iniciado. (Por questão de clareza, chamaremos isso de *fila de solicitações*.) O programa usa a chamada MQOPEN para abrir essa fila para entrada compartilhada.

O programa usa a chamada MQGET para remover as mensagens dessa fila. Essa chamada usa as opções MQGMO_ACCEPT_TRUNCATED_MSG e MQGMO_WAIT com um intervalo de espera de 5 segundos. O programa testa o descritor de cada mensagem para ver se é uma mensagem de solicitação; se não for, o programa descarta a mensagem e exibe uma mensagem de aviso.

Para cada mensagem de solicitação removida da fila de solicitações, o programa lê o nome da fila (que chamaremos a *fila de destino*) contida nos dados e abre essa fila usando a chamada MQOPEN com a opção MQOO_SET. O programa usa, então, a chamada MQSET para configurar o valor do atributo *InhibitPut* da fila de destino para MQQA_PUT_INHIBITED.

Se a chamada MQSET for bem-sucedida, o programa usará a chamada MQPUT1 para colocar uma mensagem de resposta na fila de resposta. Esta mensagem contém a sequência `PUT inhibited`.

Se a chamada MQOPEN ou MQSET não for bem-sucedida, o programa usará a chamada MQPUT1 para colocar uma mensagem `report` na fila de resposta. No campo *Feedback* do descritor da mensagem de relatório está o código de razão retornado pela chamada MQOPEN ou pela chamada MQSET, dependendo de qual falhou.

Após a chamada MQSET, o programa fecha a fila de destino usando a chamada MQCLOSE.

Quando não houver nenhuma mensagem restante na fila de solicitações, o programa fecha essa fila e desconecta do gerenciador de filas.

O programa de amostra SSL/TLS

AMQSSLC é um programa C de amostra que demonstra como usar as estruturas MQCNO e MQSCO para fornecer informações de conexão do cliente SSL/TLS na chamadas MQCONNX. Isso permite que um aplicativo MQI cliente forneça a definição do canal de conexão do cliente e as configurações de SSL/TLS no tempo de execução sem uma tabela de definição de canal do cliente (CCDT).

Se um nome de conexão for fornecido, o programa constrói uma definição de canal de conexão do cliente em uma estrutura MQCD.

Se o nome de stem do arquivo de repositório de chaves for fornecido, o programa construirá uma estrutura MQSCO; se uma URL do replicador OCSP também for fornecida, o programa construirá uma estrutura MQAIR de registro de informações de autenticação.

O programa, então, conecta-se ao gerenciador de filas usando MQCONNX. Ele consulta e imprime o nome do gerenciador de filas ao qual ele está conectado.

Esse programa destina-se a ser vinculado como um aplicativo cliente de MQI. No entanto, ele pode ser vinculado como um aplicativo MQI regular. Em seguida, ele simplesmente se conecta a um gerenciador de filas local e ignora as informações de conexão do cliente

AMQSSLC aceita os seguintes parâmetros, todos os quais são opcionais:

-m QmgrName
Nome do gerenciador de filas ao qual se conectar

-c ChannelName
Nome do canal a ser usado

-x ConnName
Nome de conexão do servidor

Parâmetros SSL/TLS:

-k KeyReposStem
O nome do stem do arquivo de repositório de chaves. Este é o caminho completo para o arquivo sem o sufixo .kdb. Por exemplo:

```
/home/user/client
C:\User\client
```

-s CipherSpec
A sequência CipherSpec do canal SSL/TLS correspondente ao SSLCIPH na definição de canal SVRCONN no gerenciador de filas.

-f
Especifica que apenas algoritmos certificados por FIPS 140-2 devem ser usados.

-b VALUE1[,VALUE2...]
Especifica que apenas algoritmos em conformidade com o Conjunto B devem ser usados. Este parâmetro é uma lista separada por vírgula de um ou mais dos valores a seguir: NONE,128_BIT,192_BIT. Estes valores possuem o mesmo significado que aqueles para a variável de ambiente MQSUITEB e a configuração EncryptionPolicySuiteB equivalente na sub-rotina SSL do arquivo de configuração do cliente.

-p Policy
Especifica a política de validação de certificado a ser usada. Este pode ser um dos valores a seguir:

QUALQUER
Aplique cada uma das políticas de validação de certificado suportadas pela biblioteca de soquetes seguros e aceite a sequência de certificados se alguma das políticas considerar a sequência de certificados válida. Esta configuração pode ser usada para retrocompatibilidade máxima com certificados digitais mais antigos que não estão em conformidade com os padrões de certificados modernos.

RFC5280
Aplique apenas a política de validação de certificado em conformidade com RFC 5280. Esta configuração fornece validação mais estrita do que a configuração ANY, mas rejeita alguns certificados digitais mais antigos.

O valor padrão é ANY.

Parâmetro de revogação de certificado do OCSP:

-o URL
A URL do respondente de OCSP

Executando o programa de amostra SSL/TLS

Para executar o programa de amostra SSL/TLS, primeiro, deve-se configurar seu ambiente de SSL ou TLS. Em seguida, você executa a amostra a partir da linha de comandos, fornecendo diversos parâmetros.

Sobre esta tarefa

As instruções a seguir executam o programa de amostra usando certificados pessoais. Ao variar o comando, é possível, por exemplo, usar certificados de CA e verificar seus status usando um respondente OCSP. Consulte as instruções dentro da amostra.

Procedimento

1. Crie um gerenciador de filas com o nome QM1. Para obter mais informações, consulte crtmqm.
2. Crie um repositório de chaves para o gerenciador de filas. Para obter mais informações, consulte Configurando um repositório de chaves em sistemas UNIX, Linux, and Windows.
3. Crie um repositório de chaves para o cliente. Chame-o de *clientkey.kdb*.
4. Crie um certificado pessoal para o gerenciador de filas. Para obter mais informações, consulte Criando um certificado pessoal autoassinado em sistemas UNIX, Linux, and Windows.
5. Crie um certificado pessoal para o cliente.
6. Extraia o certificado pessoal do repositório de chaves do servidor e inclua o mesmo no repositório do cliente. Para obter mais informações, consulte Extraindo a parte pública de um certificado autoassinado de um repositório de chaves nos sistemas UNIX, Linux e Windows e Incluindo um certificado de autoridade de certificação (ou a parte pública de um certificado autoassinado) em um repositório de chaves nos sistemas UNIX, Linux ou Windows.
7. Extraia o certificado pessoal do repositório de chaves do cliente e inclua o mesmo no repositório de chaves do servidor.
8. Crie um canal de conexão do servidor usando o comando MQSC:

   ```
   DEFINE CHANNEL(QM1SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCIPH(NULL_SHA)
   ```

   Para obter mais informações, consulte Canal de conexão do servidor
9. Defina e inicie um listener do canal no gerenciador de filas. Para obter mais informações, consulte DEFINE LISTENER e START LISTENER.
10. Execute o programa de amostra usando o comando a seguir:

    ```
    AMQSSSLC -m QM1 -c QM1SVRCONN -x localhost
    -k "c:\Program Files\IBM\WebSphere MQ\clientkey" -s NULL_SHA
    -o http://dummy.OCSP.responder
    ```

Resultados

O programa de amostra executa as ações a seguir:

1. Conecta-se a qualquer gerenciador de filas especificado ou ao gerenciador de filas padrão, usando quaisquer opções especificadas.
2. Abre o gerenciador de filas e consulta sobre seu nome.
3. Fecha o gerenciador de filas.
4. Desconecta do gerenciador de filas.

Se o programa de amostra for executado com sucesso, ele exibe uma saída semelhante ao exemplo a seguir:

```
Sample AMQSSSLC start
Connecting to queue manager QM1
Using the server connection channel QM1SVRCONN
on connection name localhost.
Using SSL CipherSpec NULL_SHA
Using SSL key repository stem c:\Program Files\IBM\WebSphere MQ\clientkey
Using OCSP responder URL http://dummy.OCSP.responder
Connection established to queue manager QM1

Sample AMQSSSLC end
```

Se o programa de amostra encontrar um problema, ele exibirá uma mensagem de erro apropriada, por exemplo, se você especificar uma URL respondente OCSP inválida, receberá a mensagem a seguir:

```
MQCONNX ended with reason code 2553
```

Para obter uma lista de códigos de razão, consulte Códigos de razão de API.

Os programas de amostra Triggering

A função fornecida na amostra de acionamento é um subconjunto daquele fornecido no monitor acionador no programa **runmqtrm**.

Consulte “Recursos demonstrados nos programas de amostra” na página 99 para obter os nomes desses programas.

Design da amostra de acionamento

O programa de amostra de acionamento abre a fila de inicialização usando a chamada MQOPEN com a opção MQOO_INPUT_AS_Q_DEF. Ele recebe mensagens da fila de inicialização usando a chamada MQGET com as opções MQGMO_ACCEPT_TRUNCATED_MSG e MQGMO_WAIT, especificando um intervalo de espera ilimitado. O programa limpa os campos *MsgId* e *CorrelId* antes de cada chamada MQGET para obter mensagens em sequência.

Quando recupera uma mensagem da fila de inicialização, o programa testa a mensagem verificando o seu tamanho para certificar-se de que ela seja do mesmo tamanho que uma estrutura MQTM. Se esse teste falhar, o programa exibe um aviso.

Para mensagens do acionador válidas, a amostra do acionador copia dados desses campos: *ApplicId*, *EnvrData*, *Versione ApplType*. Os dois últimos campos são numéricos, portanto, o programa cria substituições de caracteres para usar em uma estrutura MQTMC2 para UNIX, Linuxe sistemas Windows .

A amostra de acionamento emite um comando inicial para o aplicativo especificado no campo *ApplicId* da mensagem do acionador e passa uma estrutura MQTMC2 ou MQTMC (uma versão em caractere da mensagem do acionador). Nos sistemas UNIX, Linux e Windows , o campo *EnvData* é usado como uma extensão para a sequência de comandos de chamada.

Por último, o programa fecha a fila de iniciação.

Executando os programas de amostra Triggering

Este tópico contém informações sobre a execução de programas de amostra Triggering.

Executando as amostras amqstrg0.c, amqstrg e amqstrgc

O programa aceita dois parâmetros:

1. O nome da fila de inicialização (necessário)
2. O nome do gerenciador de filas (opcional)

Se um gerenciador de filas não for especificado, ele se conectará ao padrão. Uma fila de inicialização de amostra terá sido definida quando amqscos0.tst foi executado; o nome dessa fila é SYSTEM.SAMPLE.TRIGGER e ela pode ser usada quando esse programa for executado.

Nota: A função nesta amostra é um subconjunto da função acionamento completo que é fornecida no programa **runmqtrm**.

Design do servidor acionador

O design do servidor acionador é semelhante ao do monitor acionador, exceto que o servidor acionador:

- Permite MQAT_CICS, bem como aplicativos MQAT_OS400
- Para aplicativos CICS , substitui o *EnvData*, por exemplo, para especificar a região CICS , a partir da mensagem do acionador no comando STRCICSUSR

- Abre a fila de inicialização para entrada compartilhada, de forma que muitos servidores acionadores possam ser executados ao mesmo tempo

Nota: Programas iniciados por AMQSERV4 não devem usar a chamada MQDISC, pois isso para o servidor acionador. Se programas iniciados por AMQSERV4 usarem a chamada MQCONN, eles obtêm o código de razão MQRC_ALREADY_CONNECTED.

Amostras do TUXEDO

Aprenda sobre os programas de amostra Put e Get para o TUXEDO e como construir o ambiente do servidor no TUXEDO.

Antes de executar essas amostras, deve-se construir o ambiente do servidor.

Nota: Em todo este tópico, o caractere de barra invertida (\) é usado para dividir os comandos longos em mais de uma linha. Não insira esse caractere. Insira cada comando como uma única linha.

Construindo o ambiente do servidor

Informações sobre como construir o ambiente do servidor para WebSphere MQ para diferentes plataformas.

Supõe-se que você tenha um ambiente funcional do TUXEDO.

Construindo o ambiente do servidor para WebSphere MQ para AIX (32 bits)

1. Crie um diretório (por exemplo, <APPDIR>) no qual o ambiente do servidor seja construído e execute todos os comandos neste diretório.
2. Exporte as seguintes variáveis de ambiente, em que TUXDIR é o diretório-raiz para TUXEDO e *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado:

```
$ export CFLAGS="-I MQ_INSTALLATION_PATH□inc -I /<APPDIR> -L MQ_INSTALLATION_PATH□lib"
$ export LDOPTS="-lmqm"
$ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=/<APPDIR>/amqstxvx.V
$ export LIBPATH=$TUXDIR□lib:MQ_INSTALLATION_PATH□lib:□lib
```

3. Inclua o seguinte no arquivo TUXEDO udataobj/RM

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa -lmqm
```

4. Execute os comandos:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT2:MPUT
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a
$ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a
```

5. Edite ubbstxcx.cfg e inclua detalhes do nome da máquina, diretórios ativos e o gerenciador de filas conforme necessário:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Crie o TLOGDEVICE:

```
$tmadmin -c
```

   Um prompt é exibido. Nesse prompt, insira:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Inicie o gerenciador de filas:

```
$ strmqm
```

8. Inicie o Tuxedo:

```
$ tmboot -y
```

Agora é possível usar os programas doputs e dogets para colocar mensagens em uma fila e recuperá-las de uma fila.

Construindo o ambiente do servidor para o WebSphere MQ para AIX (64 bits)

1. Crie um diretório (por exemplo, <APPDIR>) no qual o ambiente do servidor seja construído e execute todos os comandos neste diretório.
2. Exporte as variáveis de ambiente a seguir, em que TUXDIR representa o diretório-raiz para TUXEDO e *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está installed.:

```
$ export CFLAGS="-I MQ_INSTALLATION_PATH□inc -I /<APPDIR> -L
MQ_INSTALLATION_PATH□lib64"
$ export LDOPTS="-lmqm"
$ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=/<APPDIR>/amqstxvx.V
$ export LIBPATH=$TUXDIR□lib64:MQ_INSTALLATION_PATH□lib64:□lib64
```

3. Inclua o seguinte no arquivo TUXEDO udataobj/RM

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa64 -lmqm
```

4. Execute os comandos:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT2:MPUT
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a
$ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a
```

5. Edite ubbstxcx.cfg e inclua detalhes do nome da máquina, diretórios ativos e o gerenciador de filas conforme necessário:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Crie o TLOGDEVICE:

```
$tmadmin -c
```

   Um prompt é exibido. Nesse prompt, insira:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Inicie o gerenciador de filas:

```
 $ strmqm
```

8. Inicie o Tuxedo:

```
 $ tmboot -y
```

Agora é possível usar os programas doputs e dogets para colocar mensagens em uma fila e recuperá-las de uma fila.

Construindo o ambiente do servidor para WebSphere MQ para Solaris (32 bits)

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

1. Crie um diretório (por exemplo, *APPDIR*) no qual o ambiente do servidor é construído e execute todos os comandos nesse diretório.
2. Exporte as variáveis de ambiente a seguir, em que TUXDIR é o diretório raiz para TUXEDO:

```
$ export CFLAGS="-I /APPDIR"
$ export FIELDTBLS=amqstxvx.flds
$ export VIEWFILES=amqstxvx.V
$ export SHLIB_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
$ export LD_LIBRARY_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
```

3. Inclua o seguinte no arquivo udataobj/RM do TUXEDO

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib/libmqmxa.a MQ_INSTALLATION_PATH/lib/libmqm.so \
/opt/tuxedo/lib/libtux.a
```

4. Execute os comandos:

```
$ mkfldhdr    amqstxvx.flds
$ viewc       amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
       -l -ldl
$ buildserver -o MQSERV2 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
       -l -ldl
$ buildclient -o doputs  -f amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so  \

$ buildclient -o dogets  -f amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so
```

5. Edite ubbstxcx.cfg e inclua detalhes do nome da máquina, diretórios ativos e o gerenciador de filas conforme necessário:

```
$ tmloadcf    -y ubbstxcx.cfg
```

6. Crie o TLOGDEVICE:

```
$tmadmin -c
```

   Um prompt é exibido. Nesse prompt, insira:

```
> crdl -z /APPDIR/TLOG1
```

7. Inicie o gerenciador de filas:

```
$ strmqm
```

8. Inicie o Tuxedo:

```
$ tmboot -y
```

Agora é possível usar os programas doputs e dogets para colocar mensagens em uma fila e recuperá-las de uma fila.

Construindo o ambiente do servidor para o WebSphere MQ para Solaris (64 bits)

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

1. Crie um diretório (por exemplo, <APPDIR>) no qual o ambiente do servidor seja construído e execute todos os comandos neste diretório.
2. Exporte as variáveis de ambiente a seguir, em que TUXDIR é o diretório raiz para TUXEDO:

```
$ export CFLAGS="-I /<APPDIR>"
$ export FIELDTBLS=amqstxvx.flds
$ export VIEWFILES=amqstxvx.V
$ export SHLIB_PATH=$TUXDIR□lib:MQ_INSTALLATION_PATH□lib:□lib64
$ export LD_LIBRARY_PATH=$TUXDIR□lib64:MQ_INSTALLATION_PATH□lib64:□lib64
```

3. Inclua o seguinte no arquivo udataobj/RM do TUXEDO

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib64/libmqmxa64.a MQ_INSTALLATION_PATH/lib64/libmqm.so \
/opt/tuxedo/lib64/libtux.a
```

4. Execute os comandos:

```
$ mkfldhdr    amqstxvx.flds
$ viewc       amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
       -l -ldl
$ buildserver -o MQSERV2 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
       -l -ldl
$ buildclient -o doputs  -f amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so  \

$ buildclient -o dogets  -f amqstxgx.c \
```

```
              -f MQ_INSTALLATION_PATH/lib64/libmqm.so
```

5. Edite ubbstxcx.cfg e inclua detalhes do nome da máquina, diretórios ativos e o gerenciador de filas conforme necessário:

```
       $ tmloadcf    -y ubbstxcx.cfg
```

6. Crie o TLOGDEVICE:

```
      $tmadmin -c
```

   Um prompt é exibido. Nesse prompt, insira:

```
      > crdl -z /<APPDIR>/TLOG1
```

7. Inicie o gerenciador de filas:

```
       $ strmqm
```

8. Inicie o Tuxedo:

```
       $ tmboot -y
```

Agora é possível usar os programas doputs e dogets para colocar mensagens em uma fila e recuperá-las de uma fila.

Construindo o ambiente do servidor para WebSphere MQ para HP-UX (32 bits)

Nota: O ambiente do servidor TUXEDO de 32 bits só pode ser criado na plataforma Itanium.

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

1. Crie um diretório (por exemplo, <APPDIR>) no qual o ambiente do servidor seja construído e execute todos os comandos neste diretório.
2. Exporte as variáveis de ambiente a seguir, em que TUXDIR é o diretório raiz para TUXEDO:

```
       $ export CFLAGS="-Aa -D_HPUX_SOURCE"
       $ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
       $ export VIEWFILES=$APPDIR/amqstxvx.V
       $ export TUXCONFIG=$APPDIR/tuxconfig
       $ export PATH=$TUXDIR/bin:/usr/bin:/sbin:MQ_INSTALLATION_PATH/bin:$PATH
       $ export SHLIB_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
       $ export FLDTBLDIR=$APPDIR:$TUXDIR/udataobj
```

3. Inclua o seguinte no arquivo TUXEDO udataobj/RM

```
       MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
       MQ_INSTALLATION_PATH/lib/libmqmxa.so MQ_INSTALLATION_PATH/lib/libmqm.so \
       /opt/tuxedo/lib/libtux.sl
```

4. Execute os comandos:

```
       $ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
       $ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
```

Após executar os comandos mkfldhdr e viewc, o arquivo de cabeçalho amqstxvx.h será criado no diretório do aplicativo TUXEDO. Copie esse arquivo do diretório application do TUXEDO para o diretório include do TUXEDO e, em seguida, execute os comandos a seguir.

```
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so
$ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so
```

5. Edite ubbstxcx.cfg e inclua detalhes do nome da máquina, diretórios ativos e o gerenciador de filas conforme necessário:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Crie o TLOGDEVICE:

```
$tmadmin -c
```

Um prompt é exibido. Nesse prompt, insira:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Inicie o gerenciador de filas:

```
$ strmqm
```

8. Inicie o TUXEDO:

```
$ tmboot -y
```

Agora é possível usar os programas doputs e dogets para colocar mensagens em uma fila e recuperá-las de uma fila.

Construindo o ambiente do servidor para WebSphere MQ para HP-UX (64 bits)

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

1. Crie um diretório (por exemplo, <APPDIR>) no qual o ambiente do servidor seja construído e execute todos os comandos neste diretório.
2. Exporte as variáveis de ambiente a seguir, em que TUXDIR é o diretório raiz para TUXEDO:

```
$ export CFLAGS="-Aa -D_HPUX_SOURCE"
$ export FIELDTBLS=MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=$APPDIR/amqstxvx.V
$ export TUXCONFIG=$APPDIR/tuxconfig
$ export PATH=$TUXDIR/bin:/usr/bin:/sbin:MQ_INSTALLATION_PATH/bin:$PATH
$ export SHLIB_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib64:/lib64
$ export FLDTBLDIR=$APPDIR:$TUXDIR/udataobj
```

3. Inclua o seguinte no arquivo TUXEDO udataobj/RM

Na plataforma HP-UX IA64 (IPF):

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib64/libmqmxa64.so MQ_INSTALLATION_PATH/lib64/libmqm.so \
/opt/tuxedo/lib/libtux.sl
```

Nota: As bibliotecas do WebSphere MQ fornecidas na plataforma HP-UX IA64 (IPF) possuem uma extensão de nome de arquivo .so.

4. Execute os comandos:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
```

Após executar os comandos mkfldhdr e viewc, o arquivo de cabeçalho amqstxvx.h será criado no diretório do aplicativo TUXEDO. Copie esse arquivo do diretório application do TUXEDO para o diretório include do TUXEDO e, em seguida, execute os comandos a seguir.

```
$ buildtms    -o MQXA -r MQSeries_XA_RMI
```

Na plataforma HP-UX IA64 (IPF):

```
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSeries_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs  -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so
$ buildclient -o dogets  -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so
```

5. Edite ubbstxcx.cfg e inclua detalhes do nome da máquina, diretórios ativos e o gerenciador de filas conforme necessário:

```
$ tmloadcf    -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Crie o TLOGDEVICE:

```
$tmadmin -c
```

Um prompt é exibido. Nesse prompt, insira:

```
> crdl -z /<APPDIR>/TLOG1
```

7. Inicie o gerenciador de filas:

```
$ strmqm
```

8. Inicie o TUXEDO:

```
$ tmboot -y
```

Agora é possível usar os programas doputs e dogets para colocar mensagens em uma fila e recuperá-las de uma fila.

Construindo o ambiente do servidor para o WebSphere MQ para Windows (32 bits)

Nota: Mude os campos identificados por <> no seguinte, para os caminhos de diretório:

<MQMDIR>	o caminho do diretório especificado quando o WebSphere MQ foi instalado, por exemplo, `g:\Program Files\IBM\WebSphere MQ`
<TUXDIR>	o caminho do diretório especificado quando TUXEDO foi instalado, por exemplo `f:\tuxedo`
<APPDIR>	o caminho do diretório a ser usado para o aplicativo de amostra, por exemplo `f:\tuxedo\apps\mqapp`

Para construir o ambiente do servidor e amostras:

1. Crie um diretório de aplicativo no qual construir o aplicativo de amostra, por exemplo:

   ```
   f:\tuxedo\apps\mqapp
   ```

2. Copie os seguintes arquivos de amostra do diretório de amostra do WebSphere MQ para o diretório do aplicativo:

   ```
   amqstxmn.mak
   amqstxen.env
   ubbstxcn.cfg
   ```

3. Edite cada um desses arquivos para configurar os nomes e caminhos de diretórios usados em sua instalação.
4. Edite ubbstxcn.cfg (consulte Figura 24 na página 166) para incluir detalhes sobre o nome da máquina e o gerenciador de filas ao qual deseja se conectar.
5. Inclua a linha a seguir no arquivo TUXEDO <TUXDIR>udataobj\rm

   ```
   MQSeries_XA_RMI;MQRMIXASwitchDynamic;
      <MQMDIR>\tools\lib\mqmxa.lib <MQMDIR>\tools\lib\mqm.lib
   ```

   em que < MQMDIR> é substituído, conforme mostrado no exemplo anterior Embora mostrada aqui como duas linhas, a nova entrada deve ser uma linha no arquivo.
6. Configure as variáveis de ambiente a seguir:

   ```
   TUXDIR=<TUXDIR>
   TUXCONFIG=<APPDIR>\tuxconfig
   FIELDTBLS=<MQMDIR>\tools\c\samples\amqstxvx.fld
   LANG=C
   ```

7. Crie um dispositivo TLOG para TUXEDO. Para fazer isso, chame `tmadmin -c`e insira o comando:

   ```
   crdl -z <APPDIR>\TLOG
   ```

   em que <*APPDIR*> é substituído
8. Configure o diretório atual como < APPDIR> e chame o makefile de amostra (amqstxmn.mak) como um makefile de projeto externo. Por exemplo, com o Microsoft Visual C++, emita o comando:

   ```
   msvc amqstxmn.mak
   ```

   Selecione **build** para construir todos os programas de amostra.

```
*RESOURCES
IPCKEY        99999
UID           0
GID           0
MAXACCESSERS  20
MAXSERVERS    20
MAXSERVICES   50
MASTER        SITE1
MODEL         SHM
LDBAL         N

*MACHINES
<MachineName> LMID=SITE1
              TUXDIR="f:\tuxedo"
              APPDIR="f:\tuxedo\apps\mqapp;g:\Program Files\IBM\WebSphere MQ\bin"
              ENVFILE="f:\tuxedo\apps\mqapp\amqstxen.env"
              TUXCONFIG="f:\tuxedo\apps\mqapp\tuxconfig"
              ULOGPFX="f:\tuxedo\apps\mqapp\ULOG"
              TLOGDEVICE="f:\tuxedo\apps\mqapp\TLOG"
              TLOGNAME=TLOG
              TYPE="i386NT"
              UID=0
              GID=0

*GROUPS
GROUP1
        LMID=SITE1  GRPNO=1
        TMSNAME=MQXA
        OPENINFO="MQSeries_XA_RMI:MYQUEUEMANAGER"

*SERVERS
DEFAULT: CLOPT="-A -- -m MYQUEUEMANAGER"

MQSERV1     SRVGRP=GROUP1 SRVID=1
MQSERV2     SRVGRP=GROUP1 SRVID=2

*SERVICES
MPUT1
MGET1
MPUT2
MGET2
```

Figura 24. Exemplo de arquivo ubbstxcn.cfg para WebSphere MQ para Windows

Nota: Mude os nomes e caminhos de diretórios para corresponderem à sua instalação. Mude também o nome do gerenciador de filas MYQUEUEMANAGER para o nome do gerenciador de filas ao qual deseja se conectar. Outras informações que você precisa incluir estão identificadas pelos caracteres <>.

O arquivo ubbconfig de amostra para WebSphere MQ para Windows é listado em Figura 24 na página 166. Ele é fornecido como ubbstxcn.cfg no diretório de amostras WebSphere MQ .

O makefile de amostra (consulte Figura 25 na página 167) fornecido para o WebSphere MQ para Windows é chamado ubbstxmn.make é mantido no diretório de amostras do WebSphere MQ .

```
TUXDIR  = f:\tuxedo
MQMDIR  = g:\Program Files\IBM\WebSphere MQ
APPDIR  = f:\tuxedo\apps\mqapp
MQMLIB  = $(MQMDIR)\tools\lib
MQMINC  = $(MQMDIR)\tools\c\include
MQMSAMP = $(MQMDIR)\tools\c\samples
INC = -f "-I$(MQMINC) -I$(APPDIR)"
DBG = -f "/Zi"

amqstx.exe:
 $(TUXDIR)\bin\mkfldhdr    -d$(APPDIR) $(MQMSAMP)\amqstxvx.fld
 $(TUXDIR)\bin\viewc       -d$(APPDIR) $(MQMSAMP)\amqstxvx.v
 $(TUXDIR)\bin\buildtms    -o MQXA -r MQSeries_XA_RMI
 $(TUXDIR)\bin\buildserver -o MQSERV1 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT1:MPUT -s MGET1:MGET
 $(TUXDIR)\bin\buildserver -o MQSERV2 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT2:MPUT -s MGET2:MGET
 $(TUXDIR)\bin\buildclient -o doputs -f $(MQMSAMP)\amqstxpx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG)
 $(TUXDIR)\bin\buildclient -o dogets -f $(MQMSAMP)\amqstxgx.c \
                           -f $(MQMLIB)\mqm.lib $(INC) -v $(DBG)
 $(TUXDIR)\bin\tmloadcf    -y $(APPDIR)\ubbstxcn.cfg
```

Figura 25. Makefile TUXEDO de amostra para WebSphere MQ para Windows

Construindo o ambiente do servidor para WebSphere MQ para Windows (64 bits)

Nota: Mude os campos identificados por <> no seguinte, para os caminhos de diretório:

<MQMDIR>	o caminho do diretório especificado quando o WebSphere MQ foi instalado, por exemplo, `g:\Program Files\IBM\WebSphere MQ`
<TUXDIR>	o caminho do diretório especificado quando TUXEDO foi instalado, por exemplo `f:\tuxedo`
<APPDIR>	o caminho do diretório a ser usado para o aplicativo de amostra, por exemplo `f:\tuxedo\apps\mqapp`

Para construir o ambiente do servidor e amostras:

1. Crie um diretório de aplicativo no qual construir o aplicativo de amostra, por exemplo:

   ```
   f:\tuxedo\apps\mqapp
   ```

2. Copie os seguintes arquivos de amostra do diretório de amostra do WebSphere MQ para o diretório do aplicativo:

   ```
   amqstxmn.mak
   amqstxen.env
   ubbstxcn.cfg
   ```

3. Edite cada um desses arquivos para configurar os nomes e caminhos de diretórios usados em sua instalação.
4. Edite `ubbstxcn.cfg` (consulte Figura 26 na página 168) para incluir detalhes sobre o nome da máquina e o gerenciador de filas ao qual deseja se conectar.
5. Inclua a linha a seguir no arquivo <TUXDIR>`udataobj\rm` do TUXEDO

   ```
   MQSeries_XA_RMI;MQRMIXASwitchDynamic;
      <MQMDIR>\tools\lib64\mqmxa64.lib <MQMDIR>\tools\lib64\mqm.lib
   ```

em que <MQMDIR> foi substituído. Embora mostrada aqui como duas linhas, a nova entrada deve ser uma linha no arquivo.

6. Configure as variáveis de ambiente a seguir:

```
TUXDIR=<TUXDIR>
TUXCONFIG=<APPDIR>\tuxconfig
FIELDTBLS=<MQMDIR>\tools\c\samples\amqstxvx.fld
LANG=C
```

7. Crie um dispositivo TLOG para TUXEDO. Para fazer isso, chame `tmadmin -c` e insira o comando:

```
crdl -z <APPDIR>\TLOG
```

em que <*APPDIR*> é substituído, conforme mostrado no exemplo anterior

8. Configure o diretório atual como < APPDIR> e chame o makefile de amostra (amqstxmn.mak) como um makefile de projeto externo. Por exemplo, com o Microsoft Visual C++, emita o comando:

```
msvc amqstxmn.mak
```

Selecione **build** para construir todos os programas de amostra.

```
*RESOURCES
IPCKEY        99999
UID           0
GID           0
MAXACCESSERS  20
MAXSERVERS    20
MAXSERVICES   50
MASTER        SITE1
MODEL         SHM
LDBAL         N

*MACHINES
<MachineName> LMID=SITE1
              TUXDIR="f:\tuxedo"
              APPDIR="f:\tuxedo\apps\mqapp;g:\Program Files\IBM\WebSphere MQ\bin"
              ENVFILE="f:\tuxedo\apps\mqapp\amqstxen.env"
              TUXCONFIG="f:\tuxedo\apps\mqapp\tuxconfig"
              ULOGPFX="f:\tuxedo\apps\mqapp\ULOG"
              TLOGDEVICE="f:\tuxedo\apps\mqapp\TLOG"
              TLOGNAME=TLOG
              TYPE="i386NT"
              UID=0
              GID=0

*GROUPS
GROUP1
        LMID=SITE1  GRPNO=1
        TMSNAME=MQXA
        OPENINFO="MQSeries_XA_RMI:MYQUEUEMANAGER"

*SERVERS
DEFAULT: CLOPT="-A -- -m MYQUEUEMANAGER"

MQSERV1     SRVGRP=GROUP1 SRVID=1
MQSERV2     SRVGRP=GROUP1 SRVID=2

*SERVICES
MPUT1
MGET1
MPUT2
MGET2
```

Figura 26. Exemplo de arquivo ubbstxcn.cfg para WebSphere MQ para Windows

Nota: Mude os nomes e caminhos de diretórios para corresponderem à sua instalação. Mude também o nome do gerenciador de filas MYQUEUEMANAGER para o nome do gerenciador de filas ao qual deseja se conectar. Outras informações que você precisa incluir estão identificadas pelos caracteres <>.

O arquivo ubbconfig de amostra para WebSphere MQ para Windows é listado em Figura 26 na página 168. Ele é fornecido como ubbstxcn.cfg no diretório de amostras WebSphere MQ .

O makefile de amostra (consulte Figura 27 na página 169) fornecido para o WebSphere MQ para Windows é chamado ubbstxmn.make é mantido no diretório de amostras do WebSphere MQ .

```
TUXDIR  = f:\tuxedo
MQMDIR  = g:\Program Files\IBM\WebSphere MQ
APPDIR  = f:\tuxedo\apps\mqapp
MQMLIB  = $(MQMDIR)\tools\lib64
MQMINC  = $(MQMDIR)\tools\c\include
MQMSAMP = $(MQMDIR)\tools\c\samples
INC = -f "-I$(MQMINC) -I$(APPDIR)"
DBG = -f "/Zi"

amqstx.exe:
 $(TUXDIR)\bin\mkfldhdr    -d$(APPDIR) $(MQMSAMP)\amqstxvx.fld
 $(TUXDIR)\bin\viewc       -d$(APPDIR) $(MQMSAMP)\amqstxvx.v
 $(TUXDIR)\bin\buildtms    -o MQXA -r MQSeries_XA_RMI
 $(TUXDIR)\bin\buildserver -o MQSERV1 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT1:MPUT -s MGET1:MGET
 $(TUXDIR)\bin\buildserver -o MQSERV2 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSeries_XA_RMI \
                           -s MPUT2:MPUT -s MGET2:MGET
 $(TUXDIR)\bin\buildclient -o doputs -f $(MQMSAMP)\amqstxpx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG)
 $(TUXDIR)\bin\buildclient -o dogets -f $(MQMSAMP)\amqstxgx.c \
                           -f $(MQMLIB)\mqm.lib $(INC) -v $(DBG)
 $(TUXDIR)\bin\tmloadcf    -y $(APPDIR)\ubbstxcn.cfg
```

Figura 27. Makefile TUXEDO de amostra para WebSphere MQ para Windows

Programa do servidor de amostra para TUXEDO

O programa do servidor de amostra (amqstxsx) é projetado para ser executado com os programas de amostra Put (amqstxpx.c) e Get (amqstxgx.c). O programa do servidor de amostra é executado automaticamente quando o TUXEDO é iniciado.

Nota: Deve-se iniciar o gerenciador de filas ***antes*** de iniciar o TUXEDO.

O servidor de amostra fornece dois serviços do TUXEDO, MPUT1 e MGET1:

- O serviço MPUT1 é conduzido pela amostra PUT e usa MQPUT1 no ponto de sincronização para colocar uma mensagem em uma unidade de trabalho controlada pelo TUXEDO. Aceita os parâmetros QName e Message Text, que são fornecidos pela amostra PUT.
- O serviço MGET1 abre e fecha a fila toda vez que obtém uma mensagem. Aceita os parâmetros QName e Message Text, que são fornecidos pela amostra GET.

Quaisquer mensagens de erro, códigos de razão e mensagens de status são gravados no arquivo de log do TUXEDO.

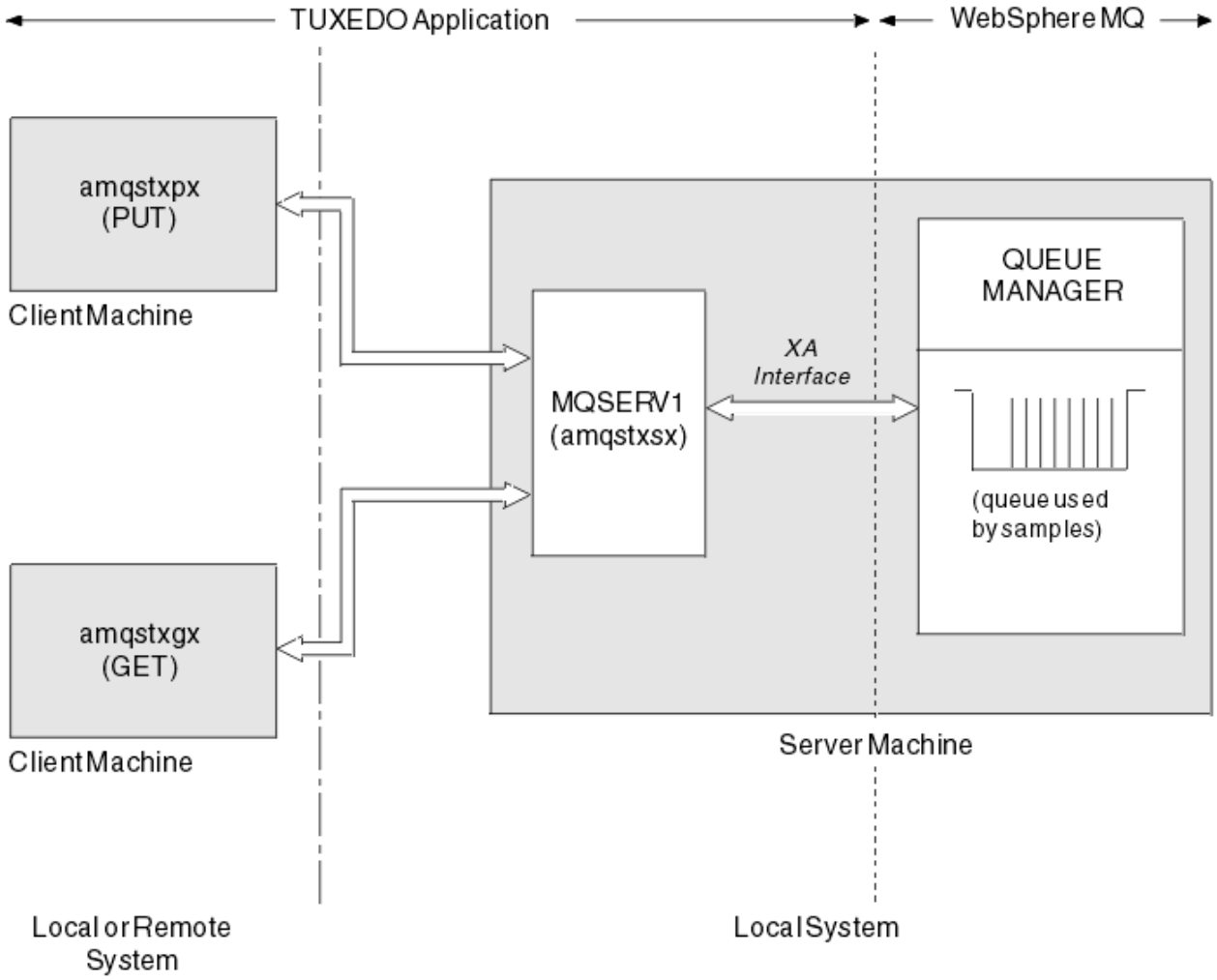


Figura 28. Como as amostras do TUXEDO funcionam juntas

Programa de amostra Put para TUXEDO

Esta amostra permite colocar uma mensagem em uma fila várias vezes, em lotes, demonstrando a indicação de sincronização usando o TUXEDO como o gerenciador de recursos.

O programa do servidor de amostra `amqstxsx` deve estar em execução para a amostra put ter sucesso; o programa de amostra do servidor se conecta ao gerenciador de filas e usa a interface XA. Para executar a amostra, insira:

- `doputs -n queuename -b batchsize -c trancount -t message`

Por exemplo:

- `doputs -n myqueue -b 5 -c 6 -t "Hello World"`

Isso coloca 30 mensagens na fila denominada `myqueue`, em seis lotes, cada um com cinco mensagens. Se houver algum problema, ele faz um lote de mensagens de saída, caso contrário, ele os confirma.

Quaisquer mensagens de erro serão gravadas no arquivo de log TUXEDO e no stderr. Quaisquer códigos de razão serão gravados no stderr.

Amostra Get para TUXEDO

Esta amostra permite obter mensagens de uma fila em lotes.

O programa do servidor de amostra `amqstxsx` deve estar em execução para a amostra put ter sucesso; o programa de amostra do servidor se conecta ao gerenciador de filas e usa a interface XA. Para executar a amostra, insira:

• `dogets -n queuename -b batchsize -c trancount`

Por exemplo:

• `dogets -n myqueue -b 6 -c 4`

Isso tira 24 mensagens da fila denominada `myqueue` em seis lotes, cada um com quatro mensagens. Se isso for executado após o exemplo de colocação, que coloca 30 mensagens na `myqueue`, haverá apenas seis mensagens em `myqueue`. O número de lotes e o tamanho do lote podem variar entre a colocação das mensagens e a obtenção delas.

Quaisquer mensagens de erro serão gravadas no arquivo de log TUXEDO e no stderr. Quaisquer códigos de razão serão gravados no stderr.

Usando a saída de segurança SSPI em sistemas Windows

Este tópico descreve como usar os programas de saída do canal SSPI nos sistemas Windows .. O código de saída fornecido está em dois formatos: objeto e origem.

Código de objeto

O arquivo de código de objeto é chamado amqrspin.dll. Para cliente e servidor, ele é instalado como uma parte padrão do WebSphere MQ para Windows na pasta *MQ_INSTALLATION_PATH*`/exits/`*INSTALLATION_NAME* . Por exemplo, `C:\Program Files\IBM\WebSphere MQ\exits\installation2`. É carregado como uma saída de usuário padrão. É possível executar a saída do canal de segurança fornecida e usar os serviços de autenticação em sua definição do canal.

Para fazer isso, especifique um dos seguintes:

```
SCYEXIT('amqrspin(SCY_KERBEROS)')

SCYEXIT('amqrspin(SCY_NTLM)')
```

Para fornecer suporte a um canal restrito, especifique o seguinte no canal SVRCONN:

```
SCYDATA('remote_principal_name')
```

em que *remote_principal_name* está no formato DOMAIN\user. O canal seguro é estabelecido somente se o nome do principal remoto corresponder a *remote_principal_name*.

Para usar os programas de saída do canal fornecidos entre sistemas que operam dentro de um domínio de segurança Kerberos, crie um servicePrincipalName para o gerenciador de filas.

Código-fonte

O arquivo de código-fonte de saída é chamado amqsspin.c. Ele está em `C:\Program Files\IBM\WebSphere MQ\Tools\c\Samples`

Se você modificar o código-fonte, deverá recompilar a origem modificada.

É possível compilar e vincular a ele da mesma maneira que qualquer outro canal de saída para a plataforma relevante, exceto que os cabeçalhos SSPI precisam ser acessados no tempo de compilação e as bibliotecas de segurança SSPI, juntamente com quaisquer bibliotecas associadas recomendadas, precisam ser acessadas no tempo do link.

Antes de executar o comando a seguir, certifique-se de que cl.exe e a biblioteca Visual C++ e a pasta `include` estejam disponíveis em seu caminho. Por exemplo:

```
cl /VERBOSE /LD /MT /I<path_to_Microsoft_platform_SDK\include>
/I<path_to_WebSphere MQ\tools\c\include> amqsspin.c /DSECURITY_WIN32
-link  /DLL /EXPORT:SCY_KERBEROS /EXPORT:SCY_NTLM STACK:8192
```

Nota: O código-fonte não inclui nenhuma provisão para rastreio ou manipulação de erros. Se você modificar e usar o código-fonte, inclua suas próprias rotinas de rastreio e de manipulação de erro.

Executando as amostras usando filas remotas

É possível demonstrar o enfileiramento remoto ao executar as amostras em gerenciadores de filas conectadas.

O programa amqscos0.tst fornece uma definição local de uma fila remota (SYSTEM.SAMPLE.REMOTE) que usa um gerenciador de filas remotas denominado OTHER. Para usar essa definição de amostra, mude OTHER para o nome do segundo gerenciador de filas que você deseja usar. Deve-se também configurar um canal de mensagens entre os dois gerenciadores de filas; para obter informações sobre como fazer isso, consulte Definindo os canais.

Os programas de amostra Request colocam seu próprio nome do gerenciador de filas locais no campo *ReplyToQMgr* de mensagens que eles enviam. As amostras Inquire e Set enviam mensagens de resposta para a fila e para o gerenciador de filas de mensagens nomeado nos campos *ReplyToQ* e *ReplyToQMgr* das mensagens de solicitação que processam.

O programa de amostra de Cluster Queue Monitoring (AMQSCLM)

Essa amostra usa os recursos de balanceamento de carga de trabalho do cluster IBM WebSphere MQ integrados para direcionar mensagens para instâncias de filas que têm aplicativos consumidores conectados. Esta direcionamento automático evita o acúmulo de mensagens em uma instância de uma fila de clusters para os quais nenhum aplicativo de consumo está conectado.

Visão Geral

É possível configurar um cluster que possui mais de uma definição para a mesma fila em gerenciadores de filas diferentes. Esta configuração proporciona o benefício de maior disponibilidade e balanceamento de carga de trabalho. No entanto, não há recurso integrado em IBM WebSphere MQ para modificar dinamicamente a distribuição de mensagens em um cluster com base no estado de aplicativos anexados. Por essa razão, um aplicativo consumidor deve estar sempre conectado a cada instância de uma fila para assegurar que as mensagens são processadas.

O programa de amostra de monitoramento da fila de clusters monitora o estado de aplicativos anexados. O programa ajusta dinamicamente a configuração de balanceamento de carga de trabalho integrada para direcionar mensagens para instâncias de uma fila em cluster com aplicativos consumidores anexados. Em determinadas situações este programa pode ser usado para diminuir a necessidade de um aplicativo consumidor estar sempre conectado a todas as ocorrências de uma fila. Também reenvia mensagens que se tornam enfileiradas em uma instância de uma fila sem aplicativos consumidores anexados. re-envio de mensagens permite que as mensagens sejam roteadas ao redor de um aplicativo de consumo que está temporariamente encerrado.

O programa é projetado para ser usado quando os aplicativos consumidores são de longa execução, em vez de aplicativos que conectam e desconectam frequentemente.

O programa de amostra de monitoramento da fila de clusters é o programa executável compilado do arquivo de amostra C `amqsclma.c`.

Informações adicionais sobre clusters e carga de trabalho podem ser localizadas em Usando clusters para gerenciamento de carga de trabalho

AMQSCLM: projetando e planejando o uso da amostra

Informações sobre como o programa de amostra de monitoramento de fila de clusters funciona, pontos a serem considerados ao configurar um sistema para execução do programa de amostra e modificações que podem ser feitas no código-fonte de amostra.

Projetar

O programa de amostra de monitoramento da fila de clusters monitora filas locais em cluster que têm aplicativos de consumo conectados. O programa monitora as filas especificadas pelo usuário. O nome da fila pode ser específico, por exemplo APP.TEST01 ou genérico. Nomes genéricos devem estar em um formato que esteja de acordo com o PCF (Formato de comando programável). Exemplos de nomes genéricos são APP.TEST* ou APP*.

Cada gerenciador de filas em um cluster que tem uma instância de uma fila local a ser monitorada requer que uma instância do programa de amostra de monitoramento de fila de clusters esteja conectada a ele.

Roteamento de mensagem dinâmico

O programa de amostra de monitoramento da fila de clusters usa o valor de **IPPROCS** (aberto para contagem de processo de entrada) de uma fila para determinar se essa fila tem algum consumidor. Um valor maior que 0 indica que a fila tem pelo menos um aplicativo de consumo conectado. Essas filas estão ativas. Um valor igual a 0 indica que a fila não tem programas de consumo conectados. Essas filas estão inativas.

Para uma fila em cluster com várias instâncias em um cluster, o WebSphere MQ usa a propriedade de prioridade da carga de trabalho do cluster **CLWLPRTY** de cada instância de fila para determinar para quais instâncias enviar mensagens.. WebSphere MQ envia mensagens para as instâncias disponíveis de uma fila com o valor **CLWLPRTY** mais alto.

O programa de amostra do monitoramento da fila de clusters ativa uma fila de clusters ao configurar o valor do **CLWLPRTY** local como 1. O programa desativa uma fila de clusters ao configurar seu valor **CLWLPRTY** como 0.

A tecnologia de armazenamento em cluster do WebSphere MQ propaga a propriedade **CLWLPRTY** atualizada de uma fila em cluster para todos os gerenciadores de filas relevantes no cluster. Por exemplo,

- Um gerenciador de filas com um aplicativo conectado que coloca mensagens na fila.
- Um gerenciador de filas que tem uma fila local com o mesmo nome no mesmo cluster.

A propagação é feita usando os gerenciadores de filas de repositório completo do cluster. Novas mensagens para a fila de clusters são direcionadas para as instâncias com o valor de **CLWLPRTY** mais alto no cluster.

Transferência de mensagem enfileirada

A modificação dinâmica do valor de **CLWLPRTY** influencia o roteamento de novas mensagens. Essa modificação dinâmica não afeta as mensagens já enfileiradas em uma instância da fila sem consumidores conectados ou mensagens que passaram pelo mecanismo de balanceamento de carga de trabalho antes de um valor modificado de **CLWLPRTY** ter sido propagado pelo cluster. Como resultado, as mensagens permanecem em qualquer fila inativa e não serão processadas por um aplicativo de consumo. Para resolver isso, o programa de amostra de monitoramento da fila de clusters é capaz de obter mensagens de uma fila local sem nenhum consumidor e enviar essas mensagens para instâncias remotas da mesma fila nas quais há consumidores conectados.

O programa de amostra de monitoramento da fila de clusters transfere mensagens de uma fila local inativa para uma ou mais filas remotas ativas obtendo mensagens (usando **MQGET**) e colocando mensagens (usando **MQPUT**) na mesma fila em cluster. Essa transferência faz com que o gerenciamento de carga de trabalho do cluster do WebSphere MQ selecione uma instância de destino diferente, com base em um valor CLWLPRTY superior ao da instância da fila local. Persistência de mensagem e contexto são preservados durante a transferência de mensagem. Ordem de mensagens e quaisquer opções vinculantes não são preservadas.

Planejamento

O programa de amostra de monitoramento da fila de clusters modifica a configuração de cluster quando há uma mudança na conectividade de aplicativos de consumo. Modificações são transmitidas

dos gerenciadores de filas onde o programa de amostra de monitoramento da fila de clusters está monitorando as filas para os gerenciadores de filas de repositório completo no cluster. Os gerenciadores de filas de repositório completo processam as atualizações da configuração e reenviam as mesmas a todos os gerenciadores de filas relevantes no cluster. Gerenciadores de filas relevantes incluem os gerenciadores de filas que têm filas em cluster com o mesmo nome (em que uma instância do programa de amostra de monitoramento de fila de clusters está em execução) e qualquer gerenciador de filas em que um aplicativo abriu a fila de clusters para colocar mensagens na mesma nos últimos 30 dias.

As mudanças são processadas de forma assíncrona em todo o cluster. Portanto, após cada mudança, diferentes gerenciadores de filas no cluster podem ter diferentes visualizações da configuração por um período de tempo.

O programa de amostra de monitoramento da fila de clusters é adequado somente para sistemas nos quais os aplicativos de consumo conectam ou desconectam com pouca frequência; por exemplo, aplicativos de consumo de longa execução. Quando usado para monitorar sistemas nos quais aplicativos de consumo estão conectados somente por períodos curtos, a latência incorrida ao distribuir as atualizações de configuração pode resultar em gerenciadores de filas no cluster terem uma visualização incorreta das filas às quais consumidores estão conectados. Essa latência pode resultar em mensagens roteadas incorretamente.

Ao monitorar muitas filas, uma taxa relativamente baixa de mudança dos consumidores conectados entre todas as filas pode aumentar o tráfego de configuração de cluster por todo o cluster. O aumento do tráfego de configuração de cluster pode resultar em carga excessiva em um ou mais dos gerenciadores de filas a seguir.

- Os gerenciadores de filas em que o programa de amostra de monitoramento de fila de cluster está em execução
- Os gerenciadores de filas de repositório completo
- Um gerenciador de filas com um aplicativo conectado que coloca mensagens na fila
- Um gerenciador de filas que tem uma fila local com o mesmo nome no mesmo cluster

O uso do processador nos gerenciadores de filas de repositório completo deve ser avaliado. O uso do processador adicional é visível como tráfego de mensagens na fila do repositório completo SYSTEM.CLUSTER.COMMAND.QUEUE. Se mensagens se acumularem nessa fila, isso indica que os gerenciadores de filas do repositório completo não são capazes de acompanhar o ritmo de mudança da configuração de cluster no sistema.

Quando várias filas estão sendo monitoradas pelo programa de amostra de monitoramento de fila de clusters, há uma quantia de trabalho executada pelo programa de amostra e pelo gerenciador de filas. Esse trabalho é executado mesmo quando não há mudanças nos consumidores conectados. O argumento **-i** pode ser modificado para reduzir o uso do processador do programa de amostra no sistema local, reduzindo a frequência do ciclo de monitoramento.

Para ajudar a detectar atividade excessiva, o programa de amostra de monitoramento de fila de clusters relata o tempo médio de processamento por intervalo de pesquisa, o tempo de processamento decorrido e o número de mudanças na configuração. Os relatórios são entregues em uma mensagem informativa, **CLM0045I**, a cada 30 minutos ou a cada 600 intervalos de pesquisa, o que ocorrer primeiro.

Requisitos de uso de monitoramento da fila de clusters

O programa de amostra de monitoramento de fila de clusters tem requisitos e restrições. É possível modificar o código-fonte de amostra fornecido para mudar algumas dessas restrições em como pode ser usado. Os exemplos listados nesta seção detalham as modificações que podem ser feitas.

- O programa de amostra de monitoramento de fila de clusters é projetado para ser usado para monitorar filas quando aplicativos de consumo estão conectados ou não conectados. Se o sistema tiver aplicativos de consumo que estão frequentemente conectando e desconectando, o programa de amostra poderá gerar atividade excessiva de configuração de cluster em todo o cluster. Isso pode ter um impacto no desempenho dos gerenciadores de filas no cluster.

- O programa de amostra de monitoramento da fila do cluster depende da tecnologia subjacente do sistema e do cluster do WebSphere MQ O número de filas que estão sendo monitoradas, a frequência de monitoramento e a frequência da mudança do estado de cada fila afeta o carregamento no sistema geral. Esses fatores devem ser considerados ao selecionar as filas a serem monitoradas e o intervalo de pesquisa do monitoramento.
- Uma instância do programa de amostra de monitoramento de fila de clusters deve ser conectada a cada gerenciador de filas do cluster que tem uma instância de uma fila a ser monitorada. Não é necessário conectar o programa de amostra a gerenciadores de filas no cluster que não têm as filas.
- O programa de amostra de monitoramento de fila de clusters deve ser executado com autorização adequada para acessar todos os recursos WebSphere MQ necessários. Por exemplo,
  - O gerenciador de filas ao qual ser conectado
  - O SYSTEM.ADMIN.COMMAND.QUEUE
  - Todas as filas a serem monitoradas quando a transferência de mensagem é executada
- O servidor de comandos deve estar em execução para cada gerenciador de filas com o programa de amostra de monitoramento de fila de clusters conectado.
- Cada instância do programa de amostra de monitoramento de fila de cluster requer uso exclusivo de uma fila local (sem cluster) no gerenciador de filas ao qual está conectado. Essa fila local é usada para controlar o programa de amostra e receber mensagens de resposta de consultas feitas ao servidor de comandos do gerenciador de filas.
- Todas as filas a serem monitoradas por uma única instância do programa de amostra de monitoramento de fila de clusters devem estar no mesmo cluster. Se um gerenciador de filas tiver filas em vários clusters que requerem monitoramento, várias instâncias do programa de amostra serão necessárias. Cada instância precisa de uma fila local para mensagens de controle e de resposta.
- Todas as filas a serem monitoradas devem estar em um único cluster. Filas configuradas para usar uma lista de nomes de cluster não são monitoradas.
- Ativar a transferência de mensagens de filas inativas é opcional. Ela se aplica a todas as filas que estão sendo monitoradas pela instância do programa de amostra de monitoramento de fila de clusters. Se somente um subconjunto das filas que estão sendo monitoradas requerem a transferência de mensagem ativada, duas instâncias do programa de amostra de monitoramento de fila de clusters são necessárias. Um programa de amostra tem transferência de mensagem ativada e o outro tem transferência de mensagem desativada. Cada instância do programa de amostra precisa de uma fila local para mensagens de controle e de resposta.
- O balanceamento de carga de trabalho do cluster do WebSphere MQ enviará, por padrão, mensagens para instâncias de filas em cluster que residem no mesmo gerenciador de fila ao qual um aplicativo de colocação está conectado.. Deve ser desativado enquanto a fila local estiver inativa nas circunstâncias a seguir:
  - Os aplicativos de put se conectam a gerenciadores de filas que têm instâncias de uma fila inativa que estão sendo monitoradas
  - Mensagens enfileiradas estão sendo transferidas de filas inativas para filas ativas.

  A preferência de balanceamento de carga de trabalho local na fila pode ser desativada estaticamente por meio da configuração do valor de CLWLUSEQ para ANY. Nesta configuração, as mensagens colocadas em filas locais são distribuídas para instâncias de filas locais e remotas para balancear a carga de trabalho, mesmo quando houver aplicativos de consumo locais. Como alternativa, o programa de amostra de monitoramento de fila de clusters pode ser configurado para definir temporariamente o valor de **CLWLUSEQ** para ANY enquanto a fila não tiver consumidores conectados, o que resulta em somente mensagens locais indo para instâncias locais de uma fila enquanto essa fila estiver ativa.
- O sistema e aplicativos do WebSphere MQ não devem usar **CLWLPRTY** para as filas a serem monitoradas ou os canais que estão sendo usados. Caso contrário, as ações do programa de amostra de monitoramento de fila de clusters nos atributos da fila **CLWLPRTY** podem ter efeitos indesejados.
- O programa de amostra de monitoramento de fila de clusters registra informações de tempo de execução em um conjunto de arquivos de relatório. Um diretório para armazenar esses relatórios é

necessário e o programa de amostra de monitoramento de fila de clusters deve ter autorização para gravar nele.

AMQSCLM: preparando e executando a amostra

Para executar a amostra de monitoramento da fila de clusters, deve-se configurar o gerenciador de filas para aceitar com segurança as solicitações de conexão recebidas dos aplicativos em execução no modo cliente.

Antes de começar

As etapas a seguir devem ser concluídas antes de executar a amostra de monitoramento da fila de clusters.

1. Crie uma fila de trabalho em cada gerenciador de filas para o uso interno da amostra.

   Cada instância da amostra precisa de uma fila não cluster local para uso interno exclusivo. É possível escolher o nome da fila. O exemplo usa o nome `AMQSCLM.CONTROL.QUEUE`. Por exemplo, no Windows, é possível criar essa fila usando o comando **MQSC**

   ```
   DEFINE QLOCAL(AMQSCLM.CONTROL.QUEUE)
   ```

   Você pode deixar os valores de **MAXDEPTH** e **MAXMSGL** como padrão.

2. Crie um diretório para logs de mensagens de erro e informações.

   A amostra grava mensagens de diagnóstico para arquivos de relatório. Deve-se escolher um diretório no qual armazenar os arquivos. Por exemplo, no Windows, é possível criar um diretório usando o seguinte comando:

   ```
   mkdir C:\AMQSCLM\rpts
   ```

   Os arquivos de relatório criados pela amostra têm a convenção de nomenclatura a seguir:

   ```
   QmgrName.ClusterName.RPT0n.LOG
   ```

3. (Opcional) Defina a amostra de monitoramento da fila de cluster como um serviço do IBM WebSphere MQ.

   Para monitorar filas, a amostra deve sempre estar em execução. Para assegurar que a amostra de monitoramento de fila de clusters sempre esteja em execução, é possível definir a amostra como um serviço de gerenciador de filas. Definir a amostra como um serviço significa que AMQSCLM é iniciado quando o gerenciador de filas é iniciado. É possível usar o exemplo **RUNMQSC** a seguir para definir a amostra de monitoramento da fila de clusters como um serviço IBM WebSphere MQ .

   ```
   define service(AMQSCLM) +
       descr('Active Cluster Queue Message Distribution Monitor - AMQSCLM') +
       control(qmgr) +
       servtype(server) +
       startcmd('<Install Root>\tools\c\samples\Bin\AMQSCLM.exe') +
    startarg('-m +QMNAME+ -c CLUSTER1 -q ABC* -r AMQSCLM.CONTROL.QUEUE -l c:\AMQSCLM\rpts') +
    stdout('C:\AMQSCLM\rpts\+QMNAME+.TSTCLUS.stdout.log') +
    stderr('C:\AMQSCLM\rpts\+QMNAME+.TSTCLUS.stderr.log')
   ```

   em que <Install Root> é o local de sua instalação.

Definição	Descrição
service	Especifica o nome do serviço. É possível escolher o nome do serviço.
descr	Especifica uma descrição textual do serviço.
control	Indica que o serviço é iniciado e parado ao mesmo tempo que o gerenciador de filas.
servtype	Indica um objeto de serviço do servidor, significando que somente uma instância pode ser executada ao mesmo tempo para este gerenciador de filas.

Definição	Descrição
startcmd	Especifica o local e o nome do programa.
startarg	Especifica os argumentos da amostra. Observe o uso do *+QMNOME+*. O nome do gerenciador de filas é substituído automaticamente.
stdout	O nome completo do arquivo para o qual a saída padrão é redirecionada. A amostra grava nesse arquivo apenas mensagens confirmando que a amostra foi encerrada. A de amostra faz isso porque o arquivo de erro padrão já foi fechado em um estágio anterior do processo de finalização de amostra.
stderr	O nome do arquivo completo para o qual a saída de erro padrão é redirecionada. A amostra grava no arquivo de erro padrão quaisquer mensagens de erro anteriores ao término da amostra.

Sobre esta tarefa

Esta tarefa permite iniciar e parar a amostra de monitoramento da fila de clusters de diferentes maneiras. Ela também permite executar a amostra em um modo que gera arquivos de relatório que contêm informações estatísticas sobre as filas que estão sendo monitoradas.

O programa de amostra pode ser executado usando o comando a seguir.

```
AMQSCLM -m QMgrName -c ClusterName (-q QNameMask | -f QListFile) -r MonitorQName
[-l ReportDir] [-t] [-u ActiveVal] [-i Interval] [-d] [-s] [-v]
```

A tabela lista os argumentos que podem ser usados com a amostra de monitoramento de fila de clusters, juntamente com informações adicionais sobre cada um.

Argumento	Variável	Informações adicionais
-m	**QMgrName**	O gerenciador de filas a monitorar.
-c	**ClusterName**	O cluster que contém as filas a monitorar.
-q	**QNameMask**	A fila, ou filas, a monitorar. Um * à direita monitora todas as filas com nomes que correspondem a zero ou mais caracteres à direita.
-f	**QListFile**	O caminho completo e o nome do arquivo de um arquivo que contém uma lista de nomes de filas de máscaras de nomes de filas para monitorar. O arquivo deve conter um nome de fila/máscara por linha. É possível especificar **-q** ou **-f**, mas não ambos.
-r	**MonitorQName**	A fila local que está sendo usada exclusivamente pela amostra.
-l	**ReportDir**	O caminho do diretório no qual armazenar mensagens de informações registradas em um conjunto de agrupamento < fn> Para cada gerenciador de filas e combinação de filas, é gerado um arquivo de relatório que é limitado em um determinado tamanho O criador de logs sempre grava no mesmo arquivo, mas também mantém as duas versões anteriores do arquivo. < /fn> arquivos de relatório.
-t		(Opcional) Ativa a transferência de mensagens enfileiradas de filas locais inativas para as filas ativas. Se não estiver ativada, somente novas mensagens que entram no cluster são roteadas dinamicamente para instâncias ativas de uma fila.
-u	**ActiveVal**	(Opcional) Alterna automaticamente a propriedade **CLWLUSEQ** de uma instância da fila monitorada para ANY quando ela estiver inativa e para a propriedade **ActiveVal** quando ativa. **ActiveVal** pode ser LOCAL ou QMGR. Se esse argumento não for definido em um sistema no qual os aplicativos put se conectam ao mesmo gerenciador de filas ou onde a transferência de mensagem estiver ativada, as filas monitoradas deverão ter um valor **CLWLUSEQ** de ANY ou QMGR com o gerenciador de filas que tem um valor de ANY.
-i	**Interval**	(Opcional) O intervalo de tempo em segundos, no qual o monitor verifica as filas. O padrão é 300 segundos (5 minutos).
-d		(Opcional) Ativa a saída de diagnóstico adicional. A saída de depuração pode ser útil ao configurar inicialmente o sistema ou ao trabalhar com o código de amostra.
-s		(Opcional) Ativa a saída estatística mínima por intervalo.
-v		(Opcional) Registre as informações de relatório em `standard out`, além de nos arquivos de relatório.

Exemplos da lista de argumento:

```
-m QMGR1 -c CLUS1 -f c:\QList.txt -r CLMQ -l c:\amqsclm\rpts -s
-m QMGR2 -c CLUS1 -q ABC* -r CLMQ -l c:\amqsclm\rpts -i 600
-m QMGR1 -c CLUSDEV -q QUEUE.* -r CLMQ -l c:\amqsclm\rpts -t -u QMGR -d
```

Arquivo de lista de filas de exemplo:

```
Q1
QUEUE.*
ABC
ABD
```

Procedimento

1. Inicie a amostra de monitoramento de fila de clusters. É possível iniciar a amostra de uma das maneiras a seguir:
   - Use um prompt de comandos com as autorizações do usuário apropriadas.
   - Use o comando MQSC **START SERVICE**, se a amostra estiver configurada como um serviço do IBM WebSphere MQ.

   A lista de argumentos é a mesma em ambos os casos.

   A amostra não começa a monitorar as filas por 10 segundos depois que o programa for inicializado. Esse atraso permite que os aplicativos de consumo se conectem às filas monitoradas primeiro, impedindo mudanças desnecessárias no estado ativo da fila.
2. Pare a amostra de monitoramento de fila de clusters. A amostra é interrompida automaticamente quando o gerenciador de filas é interrompido, está parando, em quiesce ou se a conexão com o gerenciador de filas é interrompida. Há maneiras de parar a amostra sem encerrar o gerenciador de filas:
   - Configure a fila local usada exclusivamente pela amostra para desativar a função Get.
   - Envie uma mensagem com um **CorrelId** de `"STOP CLUSTER MONITOR\0\0\0\0"`, para a fila local usada exclusivamente pela amostra.
   - Finalize o processo de amostra. Isso pode resultar na perda de mensagens não persistentes sendo transferidas para filas ativas. Também pode resultar na fila local usada pela amostra sendo mantida aberta por um número de segundos após o encerramento. Esta situação impede que uma nova instância da amostra de monitoramento da fila de clusters seja iniciada imediatamente.

   Se a amostra foi iniciada como um serviço do IBM WebSphere MQ, **STOP SERVICE** não terá efeito. É possível usar um dos métodos de finalização descrito como um mecanismo **STOP SERVICE** configurado no gerenciador de filas.

Como proceder a seguir

Verifique o status da amostra.

Se o relatório estiver ativado, é possível revisar o status dos arquivos de relatório. Use o comando a seguir para revisar o arquivo de relatório mais recente.

```
QMgrName.ClusterName.RPT01.LOG
```

Para revisar os arquivos mais antigos do relatório, use os comandos a seguir.

```
QMgrName.ClusterName.RPT02.LOG
QMgrName.ClusterName.RPT03.LOG
```

Os arquivos de relatório aumentam até o tamanho máximo de aproximadamente 1 MB. Quando o arquivo RPT01 é preenchido, um novo arquivo RPT01 é criado. O antigo arquivo RPT01 é renomeado para RPT02. RPT02 é renomeado para RPT03. O antigo RPT03 é descartado.

A amostra cria mensagens de informações nas situações a seguir:

- na inicialização
- no encerramento
- Quando ele marca uma fila como **ACTIVE** ou **INACTIVE**
- quando ela recoloca as mensagens a partir de uma fila inativa para uma instância, ou instâncias, ativas

A amostra cria uma mensagem de erro *CLMnnnnE* para relatar um problema que requer atenção.

A cada 30 minutes minutos, o tempo médio de processamento de relatórios de amostra por intervalo de pesquisa e tempo de processamento decorrido. Essas informações são mantidas na mensagem CLM0045I.

Quando as mensagens de estatística estão ativadas **-s**, a amostra relata as informações estatísticas a seguir sobre cada verificação de fila:

- Tempo gasto para processar as filas (em milissegundos)
- Número de filas verificadas
- Número de mudanças feitas ativas/inativas
- Número de mensagens transferidas

Essas informações são relatadas na mensagem CLM0048I.

Os arquivos de relatório podem aumentar rapidamente no modo de depuração e se quebrar rapidamente. Nesta situação, o limite de tamanho de 1 MB para arquivos individuais pode ser excedido.

AMQSCLM: resolução de problemas

As seções a seguir contêm informações sobre os cenários que podem ser encontrados ao usar a amostra. As informações sobre possíveis explicações para um cenário e opções sobre como resolvê-lo são fornecidas.

Cenário: AMQSCLM não está iniciando

Explicação em potencial: sintaxe incorreta.

Ação: verifique a saída de erro padrão para a sintaxe correta

Explicação em potencial: o gerenciador de filas não está disponível.

Ação: verifique o arquivo de relatório para o id de mensagem CLM0010E.

Explicação em potencial: não é possível abrir ou criar o arquivo ou arquivos de relatório.

Ação: verifique a saída de erro padrão para mensagens de erro durante a inicialização.

Cenário: AMQSCLM não está mudando uma fila para ACTIVE ou INACTIVE

Explicação em potencial: a fila não está na lista de filas a serem monitoradas

Ação: verifique os valores de parâmetro **-q** e **-f**.

Explicação em potencial: a fila não é uma fila local no cluster correto.

Ação: verifique se a fila é local e se está no cluster correto.

Explicação em potencial: AMQSCLM não está em execução para este gerenciador de filas e cluster.

Ação: inicie AMQSCLM para o gerenciador de filas e cluster relevantes.

Explicação potencial: a fila é deixada INATIVO, **CLWLPRTY**= 0, porque não tem consumidores. Como alternativa, ele é deixado ACTIVE **CLWLPRTY**> =1, porque possui pelo menos um consumidor.

Ação: verifique se os aplicativos de consumo estão conectados à fila.

Explicação em potencial: o servidor de comandos do gerenciador de filas não está em execução.

Ação: verifique se há erros nos arquivos de relatórios.

Cenário: mensagens não estão sendo roteadas em torno de filas INACTIVE

Explicação em potencial: as mensagens são colocadas diretamente no gerenciador de filas que tem a fila inativa e o valor **CLWLUSEQ** da fila não é ANY e o argumento **-u** não está sendo usado para AMQSCLM.

Ação: verifique o valor de **CLWLUSEQ** do gerenciador de filas relevante ou assegure que o argumento **-u** seja usado para AMQSCLM.

Explicação em potencial: não há filas ativas em quaisquer gerenciadores de filas. As mensagens têm a carga de trabalho igualmente balanceada entre todas as filas inativas até que uma fila se torne ativa.

Ação: verifique o status das filas em todos os gerenciadores de filas.

Explicação em potencial: mensagens são colocadas em um gerenciador de filas diferente no cluster daquele que tem a fila inativa e o valor de **CLWLPRTY** 0 atualizado não é propagado para o gerenciador de filas do aplicativo de put.

Ação: verifique se os canais de cluster entre o gerenciador de filas monitorado e o gerenciador de filas de repositório completo estão em execução. Verifique se os canais entre o gerenciador de filas de put e o gerenciador de filas de repositório completo estão em execução. Verifique os logs de erros dos gerenciadores de filas monitorado, de put e de repositório completo.

Explicação em potencial: as instâncias da fila remota estão ativas (CLWLPRTY=1), mas as mensagens não podem ser roteadas para essas instâncias de filas porque o canal emissor de cluster do gerenciador de filas locais não está em execução.

Ação: verifique o status dos canais emissores de cluster do gerenciador de filas locais para o gerenciador, ou gerenciadores, de filas remotas com uma instância ativa da fila.

Cenário: AMQSCLM não está transferindo mensagens de uma fila inativa

Explicação potencial: a transferência de mensagens não está ativada (**-t**).

Ação: assegure-se de que a transferência de mensagens esteja ativada (**-t**).

Explicação em potencial: a fila não está na lista de filas a serem monitoradas.

Ação: verifique os valores de parâmetro **-q** e **-f**.

Explicação em potencial: AMQSCLM não está em execução para este ou outros gerenciadores de filas no cluster que têm instâncias da mesma fila.

Ação: inicie AMQSCLM.

Explicação potencial: A fila tem **CLWLUSEQ**=LOCAL ou **CLWLUSEQ**=QMGRe o argumento **-u** não está configurado.

Ação: Configure o parâmetro **-u** ou mude a configuração da fila ou do gerenciador de filas para ANY.

Explicação em potencial: não há instâncias ativas da fila no cluster.

Ação: Verifique as instâncias da fila com um valor de **CLWLPRTY**de 1 ou superior

Explicação potencial:as instâncias da fila remota têm consumidores (**IPPROCS**> = 1), mas estão inativas nesses gerenciadores de fila (**CLWLPRTY**= 0) porque AMQSCLM não está monitorando essas instâncias remotas.

Ação: assegure que AMQSCLM esteja em execução nesses gerenciadores de filas e/ou que a fila esteja na lista de filas a serem monitoradas verificando os valores dos parâmetros **-q** e **-f**.

Explicação potencial: As instâncias da fila remota estão ativas (**CLWLPRTY**= 1), mas são vistas como inativas no gerenciador de filas locais (**CLWLPRTY**= 0). Essa situação é atribuída ao valor atualizado de **CLWLPRTY** não ter sido propagado para esse gerenciador de filas.

Ação: assegure que os gerenciadores de filas remotas estejam conectados a pelo menos um dos gerenciadores de filas de repositório completo no cluster. Assegure que os gerenciadores de filas de repositório completo estejam funcionando corretamente. Verifique se os canais entre os gerenciadores de filas de repositório completo e os gerenciadores de filas monitorados estão em execução.

Explicação em potencial: As mensagens não estão confirmadas, portanto, elas não são recuperáveis.

Ação: verifique se o aplicativo de envio está funcionando corretamente.

Explicação em potencial: AMQSCLM não tem acesso à fila local na qual as mensagens estão enfileiradas.

Ação: Este cenário pode ser devido ao AMQSCLM não estar em execução como um usuário com autorização suficiente para acessar a fila...

Explicação em potencial: o servidor de comandos do gerenciador de filas não está em execução.

Ação: inicie o servidor de comandos do gerenciador de filas.

Explicação em potencial: AMQSCLM encontrou um erro.

Ação: verifique se há erros nos arquivos de relatórios.

Explicação em potencial: as instâncias da fila remota estão ativas (CLWLPRTY=1), mas as mensagens não podem ser transferidas para essas instâncias de filas porque o canal emissor de cluster do gerenciador de filas locais não está em execução. Isso é frequentemente acompanhado de um aviso CLM0030W no log de relatório do amqsclm.

Ação: verifique o status dos canais emissores de cluster do gerenciador de filas locais para o gerenciador, ou gerenciadores, de filas remotas com uma instância ativa da fila.

Programa de amostra para Connection Endpoint Lookup (CEPL)

IBM WebSphere MQ A amostra de Consulta do Terminal de Conexão fornece um módulo de saída simples, mas poderoso, que oferece aos usuários do WebSphere MQ uma maneira de recuperar definições de conexão de um repositório LDAP, como o Tivoli Directory Server.

O cliente Tivoli Directory Server v6.3 deve ser instalado para usar CEPL.

Um conhecimento de trabalho da administração do WebSphere MQ nas plataformas suportadas é necessário para usar essa amostra..

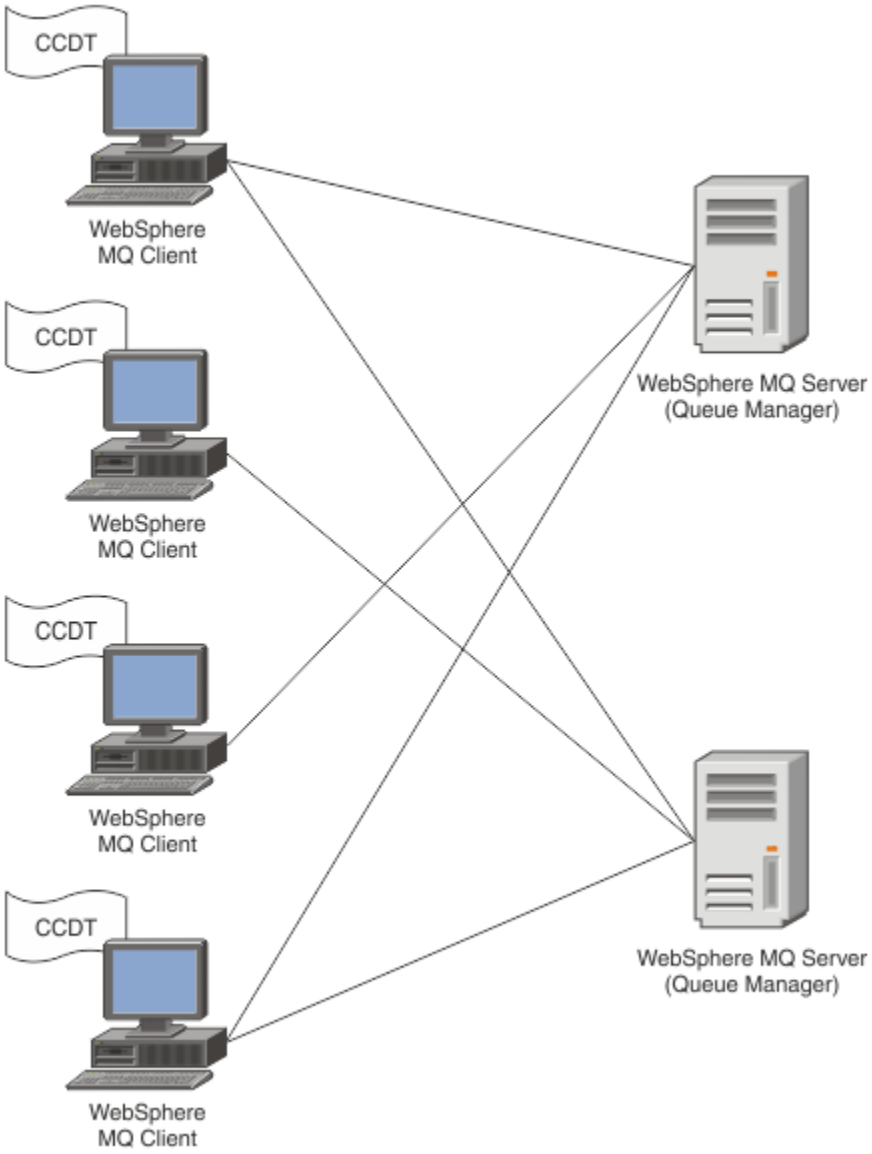
Apresentação

Configure um repositório global, por exemplo, um diretório LDAP (Lightweight Directory Access Protocol), para armazenar as definições de conexão do cliente para ajudar na manutenção e administração.

Usando um aplicativo IBM WebSphere MQ Client para estabelecer uma conexão com um Gerenciador de filas por meio de uma Tabela de definição de conexão de cliente (CCDT).

A CCDT é criada por meio da interface de Administração MQSC padrão do WebSphere MQ . O usuário deve estar conectado a um Gerenciador de filas para criar definições de conexão de cliente, embora os dados contidos na definição não sejam restritos ao Gerenciador de filas. O

arquivo da CCDT gerado deve ser distribuído manualmente entre máquinas clientes e aplicativos.

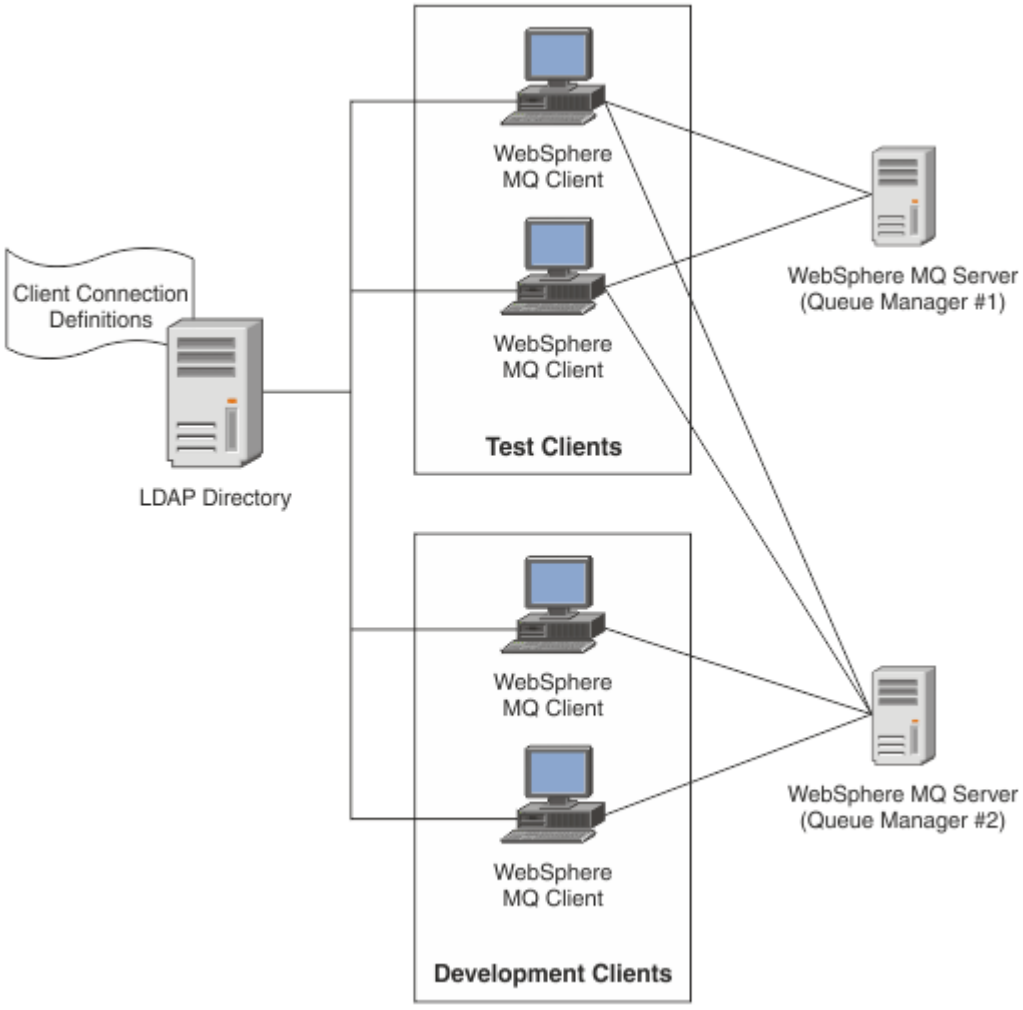


O arquivo CCDT deve ser distribuído para cada cliente WebSphere MQ . Quando milhares de clientes podem existir local ou globalmente, logo se tornaria difícil manter e administrar. Uma abordagem mais flexível é necessária para ajudar a assegurar que cada cliente tenha as definições de cliente corretas disponíveis para eles.

Uma dessas abordagens é armazenar as definições de conexão de cliente em um repositório global, como um diretório LDAP (Lightweight Directory Access Protocol). Um diretório LDAP também pode fornecer segurança adicional, indexação e recursos de procura, permitindo, assim, que cada cliente acesse somente as definições de conexão referentes a eles.

O diretório LDAP pode ser configurado de forma que somente definições específicas estejam disponíveis para determinados grupos de usuários. Por exemplo, os Clientes de teste podem acessar o Gerenciador

de filas nº 1 e nº 2, enquanto os Clientes de desenvolvimento podem acessar somente o Gerenciador de



filas nº 2.

O módulo de saída pode consultar um repositório LDAP, por exemplo, IBM Tivoli Directory Server, para recuperar definições de canal. Usando essas definições de conexão, um aplicativo cliente do WebSphere MQ pode estabelecer conexão com um gerenciador de fila

O módulo de saída é um módulo de saída preconnect que permite a definição de canal seja obtida durante a chamada MQCONN/MQCONNX a partir de um repositório LDAP.

O módulo de saída e o esquema podem ser implementados por:

- Clientes que já construíram uma base de qualificação usando a tecnologia baseada no arquivo existente da CCDT e desejam aliviar os custos de administração e distribuição.
- Os clientes que já usam sua próprias tecnologia proprietária para distribuir as definições de conexão de cliente.
- Clientes novos ou existentes que atualmente não empregam qualquer tipo de solução de conexão de cliente e desejam usar os recursos oferecidos pelo IBM WebSphere MQ.
- Clientes novos ou existentes que desejam usar diretamente ou ajustar seu modelo de sistema de mensagens em linha com qualquer arquitetura de negócios LDAP atual.

Ambientes suportados

Verifique se você tem um sistema operacional suportado e o software relevante antes de executar a amostra Connection Endpoint Lookup.

O programa de amostra para o IBM WebSphere MQ Connection Endpoint Lookup requer o software a seguir:

- IBM WebSphere MQ V7.0ou posterior
- Tivoli Directory Server V6.3 Client ou posterior

Sistemas operacionais suportados:

1. Windows (XP/2003/2008)
2. Solaris (SPARC e x86-64)
3. AIX
4. Linux
   - RHEL v4 e v5 no System p
   - SUSE v9 e v10 em System p
   - RHEL v4 e v5 System x32 bits e x64 bits
   - SUSE v9 e v10 Sistema x32 bits e x64 bits
5. HP IA64.

Nota: O programa de amostra não está disponível para as plataformas z/OS, i/5e HP PARISC

Instalação e Configuração

Instalando e configurando o módulo de saída e o esquema Connection Endpoint.

Instalando o módulo de saída

Durante a instalação do WebSphere MQ, o módulo de saída é instalado em `tools/samples/c/preconnexit/bin`.. Para plataformas de 32 bits, o módulo de saída deve ser copiado para `exit/<install name>/` antes que ele possa ser utilizado Para plataformas de 64 bits, o módulo de saída deve ser copiado para exit64/<installation name>/ antes de poder ser usado.

Instalando o esquema Connection Endpoint

A saída usa o esquema do Terminal de Conexão, *ibm-amq.schema* O arquivo de esquema deve ser importado para qualquer servidor LDAP antes que a saída possa ser usada. Após importar o esquema, valores dos atributos deverão ser incluídos.

Segue um exemplo para importar o esquema Connection Endpoint. O exemplo assume que o IBM Tivoli Directory Server (ITDS) está sendo usado

- Assegure-se de que o IBM Tivoli Directory Server esteja em execução e, em seguida, copie ou envie por FTP o arquivo *ibm-amq.schema* para o servidor ITDS.
- No servidor ITDS, insira o comando a seguir para instalar o esquema no armazenamento ITDS, no qual o ID de LDAP e a senha de LDAP são o DN raiz e a senha para o servidor LDAP:

  ldapadd -D "ID LDAP" -w "Senha LDAP" -f ibm-amq.schema
- Em uma janela de comandos, insira o comando a seguir ou use uma ferramenta de terceiro para procurar o esquema para verificação:

  ldapsearch objectclass=ibm-amqClientConnection

Consulte a documentação do Servidor LDAP para obter detalhes adicionais sobre como importar o arquivo de esquema.

Configuração

Uma nova seção denominada **PreConnect** deve ser incluída no arquivo de configuração do cliente, digamos *mqclient.ini*. A seção PreConnect contém as palavras-chave a seguir:

Module : o nome do módulo que contém o código de saída da API. Se esse campo contiver o caminho completo do módulo, ele será usado como *exit* ou *exit64* na instalação do WebSphere MQ será procurado.

Function: nome do ponto de entrada funcional na biblioteca que contém o código de saída PreConnect. A definição de função segue o protótipo MQ_PRECONNECT_EXIT.

Data : URI do repositório LDAP que contém as definições de canal.

O fragmento a seguir é um exemplo das mudanças necessárias no arquivo *mqclient.ini*.

```
PreConnect:
  Module=amqlcelp
  Function=PreConnectExit
  Data=ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com
  Sequence=1
```

Visão geral de saída e esquema

Sintaxe e parâmetros usados para estabelecer uma conexão com um gerenciador de filas.

WebSphere MQ v7.5 define a seguinte sintaxe para um ponto de entrada em um módulo de saída.

```
void MQENTRY MQ_PRECONNECT_EXIT ( PMQNXP  pExitParms
                                , PMQCHAR pQMgrName
                                , PPMQCNO ppConnectOpts
                                , PMQLONG pCompCode
                                , PMQLONG pReason)
```

Durante a execução da chamada MQCONN/X, o Cliente C do WebSphere MQ carrega o módulo de saída contendo uma implementação da sintaxe da função. Em seguida, chama uma função de saída para recuperar as definições de canal. As definições de canal recuperadas são então usadas para estabelecer a conexão com um gerenciador de filas.

Parâmetros

pExitParms

Tipo: entrada/saída PMQNXP

A estrutura do parâmetro de saída PreConnection. A estrutura é alocada e mantida pelo responsável pela chamada da saída.

```
struct tagMQNXP
{
  MQCHAR4    StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQLONG    ExitId;             /* Type of exit */
  MQLONG    ExitReason;         /* Reason for invoking exit */
  MQLONG    ExitResponse;       /* Response from exit */
  MQLONG    ExitResponse2;      /* Secondary response from exit */
  MQLONG    Feedback;           /* Feedback code (reserved) */
  MQLONG    ExitDataLength;        /* Exit data length */
  PMQCHAR      pExitDataPtr;           /* Exit data */
  MQPTR        pExitUserAreaPtr;       /* Exit user area */
  PMQCD *    ppMQCDArrayPtr;        /* Array of pointers to MQCDs */
  MQLONG    MQCDArrayCount;        /* Number of entries found */
  MQLONG    MaxMQCDVersion;        /* Maximum MQCD version */
};
```

pQMgrName

Tipo: entrada/saída PMQCHAR

Nome do gerenciador de filas. Na entrada, este parâmetro é a sequência de filtros fornecida para a chamada de API do MQCONN por meio do parâmetro **QMgrName**. Esse campo pode ficar em branco, ser explícito ou conter determinados caracteres curingas. O campo é mudado pela saída. O parâmetro é NULL quando a saída é chamada com MQXR_TERM.

ppConnectOpts

Tipo: entrada/saída ppConnectOpts

Opções que controlam a ação de MQCONNX. Esse é um ponteiro para uma estrutura de opções de conexão MQCNO que controla a ação da chamada de API MQCONN. O parâmetro é NULL quando a saída é chamada com MQXR_TERM. O cliente MQI sempre fornece uma estrutura MQCNO para a saída, mesmo que ela não tenha sido fornecida originalmente pelo aplicativo. Se um aplicativo fornecer uma estrutura MQCNO, o cliente fará uma duplicata para passá-la para a saída em que é modificado. O cliente retém a propriedade do MQCNO. Um MQCD referenciado por meio do MQCNO

tem precedência sobre qualquer definição de conexão fornecida por meio da matriz. O cliente usa a estrutura MQCNO para se conectar ao gerenciador de filas e os outros são ignorados.

pCompCode

Tipo: entrada/saída PMQLONG

Código de conclusão. Ponteiro para um MQLONG que recebe o código de conclusão de saídas. Deve ser um dos valores a seguir:

MQCC_OK - Conclusão bem-sucedida

MQCC_WARNING - Aviso (conclusão parcial)

MQCC_FAILED - Falha na chamada

pReason

Tipo: entrada/saída PMQLONG

Razão que qualifica pCompCode. Ponteiro para um MQLONG que recebe o código de razão de saída. Se o código de conclusão for MQCC_OK, o único valor válido será:

MQRC_NONE - (0, x'000') Nenhuma razão para o relatório.

Se o código de conclusão for MQCC_FAILED ou MQCC_WARNING, a função de saída poderá configurar o campo de código de razão como qualquer valor MQRC_* válido.

Informações de contexto de LDAP do MQ

A saída usa a estrutura de dados a seguir para informações de contexto.

MQNLDAPCTX

A estrutura MQNLDAPCTX tem o protótipo C a seguir.

```
typedef struct tagMQNLDAPCTX MQNLDAPCTX;
typedef MQNLDAPCTX MQPOINTER PMQNLDAPCTX;

struct tagMQNLDAPCTX
{
  MQCHAR4    StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  LDAP *    objectDirectory;    /* LDAP Instance */
  MQLONG       ldapVersion;        /* Which LDAP version to use? */
  MQLONG       port;            /* Port number for LDAP server*/
  MQLONG       sizeLimit;        /* Size limit */
  MQBOOL    ssl;            /* SSL enabled? */
  MQCHAR *    host;            /* Hostname of LDAP server */
  MQCHAR *    password;        /* Password of LDAP server */
  MQCHAR *    searchFilter;        /* LDAP search filter */
  MQCHAR *    baseDN;            /* Base Distinguished Name */
  MQCHAR *    charSet;        /* Character set */
};
```

Código de amostra para construir a saída de consulta de endpoint de conexão

Fragmentos de código para compilar a origem no Windows e nas plataformas distribuídas

Compilando a origem

É possível compilar a origem com quaisquer bibliotecas do cliente LDAP, por exemplo IBM Tivoli Directory Server 6.3 bibliotecas do cliente. Esta documentação supõe que você esteja usando bibliotecas do cliente do Tivoli Directory Server 6.3

Nota: A biblioteca de saída de pré-conexão foi testada com os seguintes servidores LDAP:

- IBM Tivoli Directory Server V6.3
- Novell eDirectory V8.2

Os fragmentos de código a seguir descrevem como compilar as saídas no Windowse outras plataformas distribuídas:

Compilando a Saída na Plataforma Windows

É possível usar o fragmento a seguir para compilar a origem de saída no Windows:

```
CC=cl.exe
LL=link.exe
CCARGS=/c /I. /DWIN32 /W3 /DNDEBUG /EHsc /D_CRT_SECURE_NO_DEPRECATE /Zl

# The libraries to include
LDLIBS=Ws2_32.lib Advapi32.lib libibmldapstatic.lib libibmldapdbgstatic.lib  \
kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib \
shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib msvcrt.lib

OBJS=amqlcel0.obj

all: amqlcelp.dll

amqlcelp.dll: $(OBJS)
        $(LL) /OUT:amqlcelp.dll /INCREMENTAL /NOLOGO /DLL /SUBSYSTEM:WINDOWS /MACHINE: X86 /
DEF:amqlcelp.def $(OBJS) $(LDLIBS) /NODEFAULTLIB:msvcrt.lib

# The exit source
amqlcel0.obj: amqlcel0.c
              $(CC) $(CCARGS) $*.c
```

Nota: É possível obter avisos ao compilar as bibliotecas do cliente IBM Tivoli Directory Server 6.3 com o compilador Microsoft Visual Studio 2005 ou superior, se você estiver usando as bibliotecas do cliente IBM Tivoli Directory Server 6.3 compiladas com o compilador Microsoft Visual Studio 2003.

Compilando a saída em outras plataformas distribuídas

É possível usar o fragmento a seguir para compilar a origem de saída em outras plataformas distribuídas, por exemplo, Linux. Algumas opções do compilador podem diferir em outras plataformas distribuídas.

```
#Make file to build exit
CC=gcc

MQML=/opt/mqm/lib
MQMI=/opt/mqm/inc
TDSI=/opt/ibm/ldap/V6.3/include
XFLAG=-m32

TDSL=/opt/ibm/ldap/V6.3/lib
```

O IBM Tivoli Directory Server envia bibliotecas de link estáticas e dinâmicas, mas apenas uma forma das bibliotecas pode ser usada Este script supõe que você esteja usando as bibliotecas estáticas.

```
#Use static libraries.
LDLIBS=-L$(TDSL) -libibmldapstatic

CFLAGS=-I. -I$(MQMI) -I$(TDSI)

all:amqlcepl

amqlcepl: amqlcel0.c
        $(CC) -o cepl amqlcel0.c  -shared -fPIC $(XFLAG) $(CFLAGS) $(LDLIBS)
```

Chamada do Módulo de Saída

O módulo de saída PreConnect pode ser chamado com três códigos de razão diferentes. Esta seção descreve cada motivo de saída em maior profundidade..

MQXR_INIT

A saída é chamada com código de razão MQXR_INIT para inicializar e estabelecer conexão com um servidor LDAP.

Antes da chamada *MQXR_INIT* , o campo *pExitDataPtr* da estrutura MQNXP teria sido preenchido com o atributo Data da sub-rotina PreConnect dentro do arquivo *mqclient.ini* (ou seja, o LDAP).

Uma URL de LDAP consiste em pelo menos o protocolo, o nome do host, o número da porta e o DN base da procura. A saída analisa a URL LDAP contida no campo *pExitDataPtr* , aloca uma estrutura de Contexto de Consulta LDAP MQNLDAPCTX e a preenche adequadamente. O endereço dessa estrutura

é armazenado no campo *pExitUserAreaPtr* Falha ao analisar corretamente a URL LDAP resulta no erro MQCC_FAILED.

Neste ponto, a saída se conecta e se liga ao servidor LDAP usando os parâmetros MQNLDAPCTX. Os identificadores de API de LDAP resultantes também são armazenados nessa estrutura.

MQXR_PRECONNECT
O módulo de saída é chamado com o código de razão MQXR_PRECONNECT para recuperar definições de canais de um servidor LDAP.

A saída procura no servidor LDAP definições de canal que correspondem ao filtro fornecido. Se o parâmetro *QMgrName* contiver um nome de gerenciador de filas específico, a procura retornará todas as definições de canal cujo valor de atributo LDAP *ibm-amqQueueManagerName* corresponde ao nome do gerenciador de filas especificado.

Se o parâmetro *QMgrName* for '*' ou ' ' (em branco), a procura retorna todas as definições de canal cujo atributo de terminal de conexão *ibm-amqIsClientDefault* é configurado como true.

Após uma procura bem-sucedida, a saída prepara um ou uma matriz de definições de MQCD e retorna para o responsável pela chamada.

MQXR_TERM
A saída é chamada com esse código de razão quando a saída deve ser limpa.. Durante isso, a saída se desconecta do servidor LDAP, libera toda a memória alocada e mantida pela saída. Isso incluirá a estrutura *MQNLDAPCTX* , a matriz de ponteiro e cada MQCD que ele referenciar Todos os outros campos são configurados para os valores padrão. Os parâmetros de saída *pQMgrName* e *ppConnectOpts* não são usados durante *MQXR_TERM* e podem ser NULL.

Esquemas LDAP
Os dados de conexão do cliente são armazenados em um repositório global denominado diretório LDAP (Lightweight Directory Access Protocol). Um cliente WebSphere MQ usa um diretório LDAP para obter as definições de conexão. A estrutura das definições de conexão do cliente WebSphere MQ no diretório LDAP é conhecida como o esquema LDAP. Um esquema LDAP é uma coleta de definições de tipo de atributo, definições de classe de objeto e outras informações que um servidor usa para determinar se uma asserção de valor de atributo ou filtro corresponde aos atributos de uma entrada e se deve permitir, incluir e modificar operações.

Armazenando dados no diretório LDAP

As definições de conexão do cliente estão localizadas sob uma ramificação específica dentro da árvore de diretórios conhecida como o ponto de conexão. Como todos os outros nós dentro de um diretório LDAP, o ponto de conexão possui um Nome Distinto (DN) associado a ele. É possível usar este nó como o ponto de início para quaisquer consultas que você fizer no diretório. Use a filtragem ao consultar o diretório LDAP para retornar um subconjunto de definições de conexão do cliente. É possível restringir o acesso a subárvores com base nas permissões concedidas em outras partes da árvore de diretórios – por exemplo, para os usuários, departamentos ou grupos.

Definindo seus próprios atributos e classes
Armazene a definição de canal do cliente, modificando o esquema LDAP. Todas as definições de dados LDAP requerem objetos e atributos. Os objetos e atributos são identificados por um número de identificador de objeto (OID) que identifica exclusivamente o objeto ou atributo. Todas as classes dentro de um esquema LDAP herdarão de forma direta ou indireta do objeto superior. O objeto de definição de canal do cliente contém os atributos do objeto superior. Todas as definições de dados LDAP requerem objetos e atributos:

- definições de objeto são coletas de atributos LDAP.
- Atributos são tipos de dados LDAP.

A descrição de cada atributo e como eles são mapeados para as propriedades normais do WebSphere MQ são descritas em Atributos LDAP

Atributos LDAP

Os atributos LDAP definidos são específicos do WebSphere MQ e mapeiam diretamente para as propriedades de conexão do cliente.

Atributos de Sequência do Diretório do Canal do Cliente do WebSphere MQ

Os atributos de sequência de caracteres com seu mapeamento para as propriedades do WebSphere MQ são listados na tabela a seguir: Os atributos podem conter valores da sintaxe de directoryString (Unicode codificado por UTF-8, ou seja, um sistema de codificação de byte variável que inclui IA5/ ASCII como um subconjunto). A sintaxe é especificada por seu número de identificação de objeto (OID).

Tabela 21. Atributos de sequência do diretório do canal do cliente WebSphere MQ

Atributo LDAP	Descrição	Propriedade do WebSphere MQ
CN	O nome comum que consiste no nome do canal e no nome do gerenciador de filas de definição.	
ibm-amqChannelName	O nome da definição do canal.	CHANNEL
ibm-amqConnectionName	O identificador de conexão de comunicação.	CONNAME
ibm-amqDescription	A descrição do canal.	DESCR
ibm-amqLocalAddress	O endereço de comunicação local do canal.	LOCLADDR
ibm-amqModeName	s a bThe LU 6.2 .	MODENAME
ibm-amqPassword	A senha que pode ser usada.	SENHA
ibm-amqQueueManagerName	O nome do gerenciador de filas ou grupo de gerenciadores de filas para o qual um aplicativo cliente WebSphere MQ pode solicitar conexão.	QMNAME
ibm-amqSecurityExitUserData	Os dados do usuário que são passados à saída de segurança.	SCYDATA
ibm-amqSecurityExitName	O nome do programa de saída a ser executado pela saída de segurança do canal.	SCYEXIT
ibm-amqSslCipherSpec	Um único CipherSpec para uma conexão SSL.	SSLCIPH
ibm-amqSslPeerName	Verifica o Nome Distinto (DN) do certificado do gerenciador de filas ou cliente peer na outra extremidade de um canal do WebSphere MQ .	SSLPEER
ibm-amqTransactionProgramName	O nome do programa de transação.	TPNAME
ibm-amqUserID	O ID do usuário a ser usado pelo MCA ao tentar iniciar uma sessão SNA segura com um MCA remoto.	USERID

Atributos de número inteiro da conexão do cliente do WebSphere MQ

Os atributos com valores predefinidos (por exemplo, um tipo enumerado) são armazenados como números inteiros padrão. Esses valores são armazenados no diretório LDAP como valores de números inteiros e não usando o nome de constante associado.

Tabela 22. Atributos de número inteiro do diretório do canal do cliente WebSphere MQ		
Atributo LDAP	**Descrição**	**Propriedade do WebSphere MQ**
ibm-amqConnectionAffinity	Determina se os aplicativos clientes, que se conectam várias vezes por meio do mesmo nome do gerenciador de filas, usam o mesmo canal do cliente.	AFFINITY
ibm-amqClientChannelWeight	Um peso para influenciar qual definição de canal de conexão do cliente é usada.	CLNTWGHT
ibm-amqHeartBeatInterval	O tempo aproximado entre fluxos de pulsação que devem ser passados de um MCA de envio quando não há mensagens na fila de transmissão.	HBINT
ibm-amqKeepAliveInterval	Um valor de tempo limite para um canal.	KAINT
ibm-amqMaximumMessageLength	O comprimento máximo de uma mensagem que pode ser transmitida no canal.	MAXMSGL
ibm-amqSharingConversations	O número máximo de conversas que compartilham cada instância do canal TCP/IP.	SHARECNV
ibm-amqTransportType	O tipo de transporte a ser usado.	TRPTYPE

Atributo booleano do canal do cliente do WebSphere MQ

Este atributo booleano não é mapeado para nenhuma propriedade WebSphere MQ . A sintaxe desse atributo indica um valor booleano.

Tabela 23. Atributo booleano do canal do cliente do WebSphere MQ	
Atributo LDAP	**Descrição**
ibm-amqIsClientDefault	Esse atributo booleano é definido para resolver o problema de procura de entradas cujo atributo ibm-amqQueueManagerName não foi definido.

Atributos da lista de canais do cliente WebSphere MQ

As propriedades do WebSphere MQ são armazenadas como um atributo de lista separado por vírgula de valor único no diretório LDAP. Os atributos são definidos da mesma maneira que os outros atributos de sequência do diretório. Os atributos da lista juntamente com seu mapeamento para as propriedades do WebSphere MQ são descritos na tabela a seguir.

Tabela 24. Atributos da lista de canais do cliente WebSphere MQ		
Atributo LDAP	**Descrição**	**Propriedade do WebSphere MQ**
ibm-amqHeaderCompression	Uma lista de técnicas de compactação de dados de cabeçalho suportadas pelo canal.	COMPHDR
ibm-amqMessageCompression	Uma lista de técnicas de compactação de dados de mensagem suportadas pelo canal.	COMPMSG
ibm-amqSendExitUserData	Os dados do usuário que são passados à saída de envio.	SENDDATA
ibm-amqSendExitUserName	O nome do programa de saída a ser executado pela saída de envio do canal.	SENDEXIT
ibm-amqReceiveExitUserData	Os dados do usuário que são passados à saída de recebimento.	RCVDATA

Tabela 24. Atributos da lista de canais do cliente WebSphere MQ (continuação)		
Atributo LDAP	**Descrição**	**Propriedade do WebSphere MQ**
ibm-amqReceiveExitName	O nome do programa de saída de usuário a ser executado pelo canal recebe a saída de usuário.	RCVEXIT

Nome Comum
O nome comum (CN) consiste no nome do canal e no nome do gerenciador de filas de definição.

É um atributo preexistente.

O formato do CN é:

```
CN=CHANNEL_NAME(DEFINING_Q_MGR_NAME)
```

Por exemplo:

```
CN=TC1(QM_T1)
```

É possível especificar somente um valor para esse atributo.

Este atributo é um atributo de sequência, e os valores não fazem distinção entre maiúsculas e minúsculas. A subsequência correspondente é ignorada. A correspondência de subsequência é uma regra de correspondência usada no subesquema que especifica o comportamento do atributo em um filtro de procura, usando uma subsequência (por exemplo, CN=jim*, em que CN é um atributo) e contém um ou mais curingas.

ibm-amqChannelName
Esse atributo especifica o nome da definição de canal.

Esse atributo possui um único valor de sequência de caracteres com um máximo de 20 caracteres que não fazem distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A correspondência de subsequência é uma regra de correspondência usada no subesquema que especifica o comportamento do atributo em um filtro de procura, usando uma subsequência e contendo um ou mais curingas.

ibm-amqDescription
Este atributo do LDAP fornece a descrição do canal.

Esse atributo tem um valor de sequência única com um máximo de 64 bytes, que não faz distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

ibm-amqConnectionName
Esse atributo de LDAP é o identificador de conexão de comunicações. Ele especifica os links de comunicações específicas que serão usadas por esse canal.

Esse atributo possui um único valor de sequência de caracteres com um máximo de 264 caracteres, que não fazem distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

ibm-amqLocalAddress
Este atributo especifica o endereço de comunicações local para o canal.

Esse atributo tem um único valor de sequência com um máximo de 48 caracteres, que não faz distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

ibm-amqModeName
Este atributo é para uso com conexões LU 6.2. Ele fornece definição adicional para as características da sessão da conexão quando uma alocação de sessão de comunicação é executada.

Esse atributo tem um único valor da sequência de caracteres de exatamente oito caracteres, que não fazem distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

ibm-amqPassword
Esse atributo LDAP especifica uma senha que possa ser usada pelo MCA ao tentar iniciar uma sessão segura de LU 6.2 com um MCA remoto.

Esse atributo tem um único valor de número inteiro com no máximo 12 dígitos. Não é um atributo pré-existente.

ibm-amqQueueManagerName
Este atributo especifica o nome do gerenciador de filas para o qual um aplicativo cliente do WebSphere MQ pode solicitar conexão.

Esse atributo tem um único valor de sequência com um máximo de 48 caracteres, que não faz distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

ibm-amqSecurityExitUserData
Este atributo LDAP especifica os dados do usuário que são transmitidos para a saída de segurança.

Esse atributo tem um único valor de sequência com um máximo de 999 caracteres, que não fazem distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

ibm-amqSecurityExitName
Este atributo LDAP especifica o nome do programa de saída a ser executado pela saída de segurança do canal.

Deixe em branco se nenhuma saída de segurança do canal estiver em vigor.

Esse atributo tem um único valor de sequência com um máximo de 999 caracteres, que não fazem distinção entre maiúsculas e minúsculas. Este atributo não é um pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

ibm-amqSslCipherSpec
Esse atributo LDAP especifica um único CipherSpec para uma conexão SSL.

Esse atributo tem um único valor de sequência com um máximo de 32 caracteres, que não faz distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

ibm-amqSslPeerName
Esse atributo LDAP é usado para verificar o Nome Distinto (DN) do certificado do gerenciador de filas de peer ou do cliente na outra extremidade de um canal do WebSphere MQ .

Este atributo LDAP tem um único valor de sequência com um máximo de 1024 bytes, que não fazem distinção entre maiúsculas e minúsculas. Não é um pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

ibm-amqTransactionProgramName
Este atributo LDAP especifica o nome do programa de transação. Ele deve ser usado com conexões LU 6.2.

Este atributo possui um único valor de sequência com um máximo de 64 caracteres, que não fazem distinção entre maiúsculas e minúsculas. Não é um pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

ibm-amqUserID
Esse atributo LDAP especifica o ID do usuário a ser usado pelo MCA ao tentar iniciar uma sessão SNA segura com um MCA remoto.

Esse atributo tem um único valor de sequência de exatamente 12 caracteres, que não faz distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

ibm-amqConnectionAffinity
Esse atributo LDAP especifica se aplicativos clientes, que se conectam várias vezes usando o mesmo nome de gerenciador de filas, usam o mesmo canal do cliente.

Esse atributo tem um único valor de número inteiro. Não é um atributo pré-existente.

ibm-amqClientChannelWeight
Esse atributo LDAP especifica um peso que influencia qual definição de canal de conexão do cliente é usada.

O atributo de peso do canal do cliente é usado para influenciar a seleção de definições de canal do cliente quando mais de uma definição apropriada estiver disponível.

Esse atributo tem um único valor de número inteiro. Não é um atributo pré-existente.

ibm-amqHeartBeatInterval
Esse atributo LDAP especifica o tempo aproximado entre fluxos de pulsação que devem ser passados a partir de um MCA de envio quando não houver mensagens na fila de transmissão.

Esse atributo tem um único valor de número inteiro. Não é um atributo pré-existente. O valor padrão é 1. O padrão é configurado na operação da variável de ambiente MQSERVER atual.

ibm-amqKeepAliveInterval
Esse atributo LDAP é usado para especificar um valor de tempo limite para um canal.

O valor desse atributo é passado para a pilha de comunicações especificando a sincronização keep-alive para o canal. É possível usar essa opção para especificar um valor de keep-alive diferente para cada canal.

Esse atributo tem um único valor de número inteiro. Não é um atributo pré-existente.

ibm-amqMaximumMessageLength
Esse atributo LDAP especifica o comprimento máximo de uma mensagem que pode ser transmitida no canal.

O valor padrão desse atributo é 104857600 de acordo com a operação da variável de ambiente MQSERVER atual. Esse atributo tem um valor de número inteiro único e não é um atributo pré-existente.

ibm-amqSharingConversations
Esse atributo LDAP especifica o número máximo de conversas que compartilham cada instância do canal TCP/IP.

Esse atributo tem um único valor de número inteiro. Esse atributo não é um atributo pré-existente.

ibm-amqTransportType
Esse atributo LDAP especifica o tipo de transporte a ser usado.

Esse atributo tem um único valor de número inteiro. Não é um atributo pré-existente.

ibm-amqIsClientDefault
Esse atributo booleano soluciona o problema de procura de entradas quando o atributo ibm-amqQueueManagerName não tiver sido definido.

Módulos de saída Preconnect geralmente procuram nos servidores LDAP com o valor do atributo ibm-amqQueueManagerName como o critério de procura. Essa consulta retornaria todas as entradas em que o valor do atributo ibm-amqQueueManagerName corresponda ao nome do gerenciador de filas especificado na chamada MQCONN/X. No entanto, ao usar as tabelas de definição de canal do cliente (CCDT), é possível configurar o nome do gerenciador de filas em uma chamada MQCONN/X como em branco ou prefixar o nome com um asterisco (*). Se o nome do gerenciador de filas estiver em branco, o cliente se conectará ao gerenciador de filas padrão. Se o nome tiver um asterisco (*) como prefixo para o gerenciador de filas, então, o cliente conecta qualquer gerenciador de filas.

De forma semelhante, o atributo ibm-amqQueueManagerName em uma entrada pode ser deixado indefinido. Nesse caso, é esperado que o cliente usando essas informações do endpoint possa se conectar a qualquer gerenciador de filas. Por exemplo, uma entrada contém as linhas a seguir:

```
ibm-amqChannelName = "CHANNEL1"
ibm-amqConnectionName = myhost(1414)
```

Neste exemplo, o cliente tenta se conectar ao gerenciador de filas especificado em execução em `myhost`.

No entanto, em Servidores LDAP, uma procura não é feita em um valor de atributo que não foi definido. Por exemplo, se uma entrada contiver as informações de conexão, exceto ibm-amqQueueManagerName, então, os resultados da procura não incluiriam esta entrada. Para superar esse problema, é possível configurar ibm-amqIsClientDefault. Esse é um atributo booleano e supõe-se que tenha um valor de FALSE, se não definido.

Para entradas em que ibm-amqQueueManagerName não foi definido e que se espera que façam parte da procura, configure ibm-amqIsClientDefault para TRUE. Quando um espaço em branco ou um asterisco (*) é especificado como o nome do gerenciador de filas em uma chamada para MQCONN/X, a saída preconnect procura no servidor LDAP todas as entradas em que o valor do atributo ibm-amqIsClientDefault está configurado para TRUE.

Nota: Não configure ou defina o atributo ibm-amqQueueManagerName se ibm-amqIsClientDefault estiver configurado para TRUE.

ibm-amqHeaderCompression
Este atributo LDAP é uma lista de técnicas de compactação de dados de cabeçalho suportadas pelo canal.

O tamanho máximo desse atributo é de 48 caracteres. Não é um atributo pré-existente.

É possível especificar somente um valor para esse atributo.

Esse atributo de lista é especificado como sequências de diretório usando um formato separado por vírgula. Por exemplo, os valores especificados para **ibm-amqHeaderCompression** é `0` que é mapeado para `NONE`. Qualquer valor que exceder o limite máximo permitido será ignorado pelo cliente. Por exemplo, ibm-amqHeaderCompression contém um máximo de 2 números inteiros na lista.

ibm-amqMessageCompression
Esse atributo LDAP é uma lista de técnicas de compactação de dados de mensagem suportadas pelo canal.

O tamanho máximo desse atributo é de 48 caracteres. Não é um atributo pré-existente.

Esse atributo não suporta diversos valores.

Esse atributo de lista é especificado como sequências de diretório usando um formato separado por vírgula. Por exemplo, o valor especificado para esse atributo é 1,2,4, que é mapeado para a sequência de compactação subjacente RLE, ZLIBFAST e ZLIBHIGH.

Qualquer valor que exceda o limite máximo permitido é ignorado pelo cliente. Por exemplo, ibm-amqMessageCompression contém um máximo de 16 números inteiros na lista.

ibm-amqSendExitUserData
Este atributo LDAP especifica dados do usuário que são transmitidos para a saída de envio.

Este atributo LDAP possui um valor de sequência de caracteres única, com um máximo de 999 caracteres, que não fazem distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

Nota: ibm-amqSendExitName e **ibm-amqSendExitUserData** precisam ser sincronizados em pares. Os dados do usuário devem ser sincronizados com o nome de saída. Portanto, se um for especificado, o outro também deve ser especificado simetricamente, mesmo se não contiver dados.

ibm-amqSendExitName
Esse atributo LDAP especifica o nome do programa de saída a ser executado pela saída de envio de canal.

Esse atributo tem um único valor de sequência com um máximo de 999 caracteres, que não fazem distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

Nota: ibm-amqSendExitName e **ibm-amqSendExitUserData** devem ser sincronizados em pares. Os dados do usuário devem ser sincronizados com o nome de saída. Portanto, se um for especificado, o outro também deverá ser simetricamente especificado, mesmo que não contenha nenhum dado.

ibm-amqReceiveExitUserData
Este atributo LDAP especifica os dados do usuário que são transmitidos para a saída de recebimento.

É possível executar uma sequência de saídas de recebimento. A sequência de dados do usuário para uma série de saídas é separada por uma vírgula, espaços ou ambos.

Esse atributo tem um único valor de sequência com um máximo de 999 caracteres, que não fazem distinção entre maiúsculas e minúsculas. Não é um atributo pré-existente.

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

Nota: **ibm-amqReceiveExitName** e **ibm-amqReceiveExitUserData** devem ser sincronizados em pares. Os dados do usuário devem ser sincronizados com o nome de saída. Portanto, se um for especificado, o outro também deverá ser simetricamente especificado, mesmo que não contenha nenhum dado.

ibm-amqReceiveExitName
Este atributo LDAP especifica o nome do programa de saída de usuário a ser executado pela saída de usuário de recebimento do canal.

Este atributo é uma lista de nomes de programas que devem ser executados em sucessão. Deixe em branco se nenhuma saída de usuário de recebimento do canal estiver em vigor.

Esse atributo tem um único valor de sequência com um máximo de 999 caracteres, que não fazem distinção entre maiúsculas e minúsculas. Ele não é um atributo pré-existente

A subsequência correspondente é ignorada. A subsequência correspondente é uma regra de correspondência usada em sub-esquema que especifica o comportamento do atributo em um filtro de procura.

Nota: **ibm-amqReceiveExitName** e **ibm-amqReceiveExitUserData** devem ser sincronizados em pares. Os dados do usuário devem ser sincronizados com o nome de saída. Portanto, se um for especificado, o outro também deverá ser especificado simetricamente, mesmo que não contenha nenhum dado.

Gravando um Aplicativo de Enfileiramento

Use essas informações para aprender como escrever aplicativos de enfileiramento, conectar e desconectar de um gerenciador de filas, publicar/assinar e abrir e fechar objetos.

Use os links a seguir para descobrir mais sobre como escrever aplicativos:

Conceitos relacionados

“Conceitos de desenvolvimento de aplicativos” na página 8
É possível usar uma opção de linguagens orientadas a objeto ou processuais para gravar aplicativos IBM WebSphere MQ. Use os links neste tópico para obter informações sobre conceitos do IBM WebSphere MQ que são úteis para desenvolvedores de aplicativos.

“Decidindo qual linguagem de programação usar” na página 80
Use estas informações para descobrir sobre linguagens de programação e estruturas suportadas pelo IBM WebSphere MQe algumas considerações para usá-las.

“Projetando aplicativos IBM WebSphere MQ” na página 91
Quando você tiver decidido como seus aplicativos podem aproveitar as plataformas e ambientes disponíveis para você, será necessário decidir como usar os recursos oferecidos pelo WebSphere MQ.

“Programas de amostra do WebSphere MQ” na página 98
Use esta coleção de tópicos para aprender sobre programas de amostra do WebSphere MQ em diferentes plataformas

“Gravando aplicativos clientes” na página 356
O que você precisa saber para gravar os aplicativos clientes no WebSphere MQ

“Usando serviços da Web no WebSphere MQ” na página 958
É possível desenvolver aplicativos IBM WebSphere MQ para serviços da web usando o transporte IBM WebSphere MQ para SOAP ou a ponte IBM WebSphere MQ para HTTP (Protocolo de Transporte de Hipertexto)

“Construindo um aplicativo IBM WebSphere MQ” na página 435
Use estas informações para saber como construir um aplicativo IBM WebSphere MQ em diferentes plataformas.

“Manipulando erros do programa” na página 555
Estas informações explicarão os erros associados às chamadas MQI de aplicativos, ao realizar uma chamada ou quando sua mensagem for entregue para seu destino final.

Visão geral da Message Queue Interface

Aprenda sobre os componentes de Message Queue Interface (MQI).

A Message Queue Interface consiste no seguinte:

- *Chamadas* por meio das quais os programas podem acessar o gerenciador de filas e seus recursos
- *Estruturas* que os programas usam para passar os dados e obter os dados do gerenciador de filas
- *Tipos de dados elementares* para passar os dados e obter os dados do gerenciador de filas

WebSphere MQ para Windows e WebSphere MQ em sistemas UNIX and Linux também fornecem:

- Chamadas por meio das quais os programas de sistemas WebSphere MQ for Windows e WebSphere MQ on UNIX and Linux podem confirmar e retornar mudanças.
- *Arquivos de inclusão* que definem os valores das constantes fornecidas nessas plataformas.
- *Arquivos de biblioteca* para vincular seus aplicativos.
- Um conjunto de programas de amostra que demonstra como usar a MQI nessas plataformas. Para obter informações adicionais sobre essas amostras, consulte “Programas de amostra para plataformas distribuídas” na página 98.
- Código de amostra e código de executável para ligações aos gerenciadores de transações externos.

Use os links a seguir para descobrir mais sobre a MQI:

- “Chamadas MQI” na página 199
- “Chamadas de ponto de sincronização” na página 199
- “Conversão de dados, tipos de dados, definições de dados e estruturas” na página 200
- “Programas stub e arquivos de biblioteca do IBM WebSphere MQ” na página 200
- “Parâmetros comuns a todas as chamadas” na página 205
- “Especificando buffers” na página 206
- “UNIX and Linux manipulação de sinal” na página 206

Conceitos relacionados

“Conectando-se e desconectando-se de um gerenciador de filas” na página 209
Para usar WebSphere MQ serviços de programação, um programa deve ter uma conexão com um gerenciador de filas. Use essas informações para aprender como conectar-se e desconectar-se de um gerenciador de filas.

“Abrindo e fechando objetos” na página 218
Estas informações fornecem um insight sobre como abrir e fechar objetos do WebSphere MQ .

“Colocando mensagens em uma fila” na página 228
Use estas informações para aprender como colocar mensagens em uma fila.

“Obtendo mensagens de uma fila” na página 243
Use estas informações para aprender como obter mensagens de uma fila.

“Consultando e configurando atributos de objeto” na página 324
Atributos são as propriedades que definem as características de um objeto WebSphere MQ .

“Confirmando e fazendo backup de unidades de trabalho” na página 327
Essas informações descrevem como confirmar e restaurar quaisquer operações de GET e PUT que ocorreram em uma unidade de trabalho.

“Iniciando aplicativos IBM WebSphere MQ usando acionadores” na página 333
Aprenda sobre acionadores e como iniciar aplicativos IBM WebSphere MQ usando acionadores.

“Trabalhando com MQI e clusters” na página 351

Há opções especiais em chamadas e códigos de retorno relacionadas ao armazenamento em cluster.

Chamadas MQI

Use estas informações para aprender sobre as chamadas no MQI

As chamadas no MQI podem ser agrupadas da seguinte maneira:

MQCONN, MQCONNX e MQDISC
Use essas chamadas para conectar um programa a (com ou sem as opções), e desconectar um programa de, um gerenciador de filas. Se você gravar programas CICS para z/OS, não precisará usar essas chamadas. No entanto, é recomendado que você as utilize se desejar portar seu aplicativo para outras plataformas.

MQOPEN e MQCLOSE
Use essas chamadas para abrir e fechar um objeto, como uma fila.

MQPUT e MQPUT1
Use essas chamadas para colocar uma mensagem em uma fila.

MQGET
Use esta chamada para procurar mensagens em uma fila ou remover mensagens de uma fila.

MQSUB, MQSUBRQ
Use essas chamadas para registrar uma assinatura em um tópico e para solicitar publicações que correspondam à assinatura.

MQINQ
Use esta chamada para pesquisar sobre os atributos de um objeto.

MQSET
Use esta chamada para configurar alguns dos atributos de uma fila. Não é possível configurar os atributos de outros tipos de objeto.

MQBEGIN, MQCMIT e MQBACK
Use essas chamadas quando WebSphere MQ for o coordenador de uma unidade de trabalho. MQBEGIN inicia a unidade de trabalho. MQCMIT e MQBACK terminam a unidade de trabalho, confirmando ou retrocedendo as atualizações feitas durante a unidade de trabalho Os comandos native start commitment control, commit e rollback são usados.

MQCRTMH, MQBUFMH, MQMHBUF, MQDLTMH
Use essas chamadas para criar um identificador de mensagens, para converter um identificador de mensagens em um buffer ou um buffer em um identificador de mensagens e para excluir um identificador de mensagens.

MQSETMP, MQINQMP, MQDLTMP
Use essas chamadas para configurar uma propriedade da mensagem em um identificador de mensagens, pesquisar sobre uma propriedade da mensagem e excluir uma propriedade de um identificador de mensagens.

MQCB, MQCB_FUNCTION, MQCTL
Use essas chamadas para registrar e controlar uma função de retorno de chamada.

MQSTAT
Use esta chamada para recuperar as informações de status sobre as operações put assíncronas anteriores.

Consulte Descrições de chamadas para obter uma descrição das chamadas MQI.

Chamadas de ponto de sincronização

Use estas informações para descobrir chamadas de ponto de sincronização em diferentes plataformas.

Chamadas de ponto de sincronização estão disponíveis da seguinte forma:

Chamadas IBM WebSphere MQ nas plataformas Windows, UNIX e Linux

Windows Linux UNIX

Os produtos a seguir fornecem as chamadas MQCMIT e MQBACK:

- IBM WebSphere MQ para Windows
- IBM WebSphere MQ em UNIX and Linux sistemas

Use chamadas de ponto de sincronização em programas para informar ao gerenciador de filas que todas as operações MQGET e MQPUT desde o último ponto de sincronização devem se tornar permanentes (confirmadas) ou devem ser restauradas. Para confirmar e retornar mudanças no ambiente CICS , use comandos como EXEC CICS SYNCPOINT e EXEC CICS SYNCPOINT ROLLBACK.

Conversão de dados, tipos de dados, definições de dados e estruturas

Use estas informações para aprender sobre conversões de dados, tipos de dados elementares, definições de dados do WebSphere MQ e estruturas ao usar a Interface da Fila de Mensagens.

Conversão de Dados

A chamada MQXCNVC (converter caracteres) converte dados de caracteres de mensagens de um conjunto de caracteres para outro. Exceto no WebSphere MQ para z/OS, essa chamada é usada apenas a partir de uma saída de conversão de dados

Consulte MQXCNVC - Converter caracteres para a sintaxe usada com a chamada MQXCNVC e “Escrevendo saídas de conversão de dados” na página 421 para obter orientação sobre como gravar e chamar saídas de conversão de dados.

Tipos de dados elementares

Para as linguagens de programação suportadas, o MQI fornece tipos de dados elementares ou campos não estruturados.

Esses tipos de dados são totalmente descritos em Tipos de dados elementares.

WebSphere MQ definições de dados

Os arquivos de definição de dados fornecidos com o WebSphere MQ contêm:

- Definições de todas as constantes e códigos de retorno do WebSphere MQ
- Definições das estruturas e tipos de dados do WebSphere MQ
- Definições de constantes para inicializar as estruturas
- Protótipos de funções para cada uma das chamadas (para PL/I e a linguagem C apenas)

Para obter uma descrição completa dos arquivos de definição de dados do WebSphere MQ , consulte “IBM WebSphere MQ arquivos de definição de dados” na página 82

Estruturas

Estruturas, usadas com as chamadas MQI listadas no “Chamadas MQI” na página 199, são fornecidas nos campos de definição de dados para cada uma das linguagens de programação suportadas.

Consulte Resumo dos tipos de dados da estrutura para obter um resumo das estruturas.

Programas stub e arquivos de biblioteca do IBM WebSphere MQ

Os programas de stub e arquivos de biblioteca fornecidos estão listados aqui, para cada plataforma.

Para obter mais informações sobre como usar os programas stub e arquivos de biblioteca ao construir um aplicativo executável, consulte “Construindo um aplicativo IBM WebSphere MQ” na página 435. Para obter informações sobre a vinculação a arquivos de biblioteca C + +, consulte Using C++ *WebSphere MQ Usando C++*.

IBM WebSphere MQ para Windows

No IBM WebSphere MQ para Windows, deve-se vincular seu programa aos arquivos de biblioteca MQI fornecidos para o ambiente no qual você está executando seu aplicativo, além daqueles fornecidos pelo sistema operacional: .

Tabela 25. Arquivos de biblioteca para aplicativos Windows

Arquivo de biblioteca	**Ambiente**
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	Server for C (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\mqic.lib	Client for C (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\mqmxa.lib	Server XA interface for C (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\mqcxa.lib	Client XA interface for C (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib	Client MTS for C (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcics4.lib	Suporte do servidor TXSeries CICS para C (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\mqccics4.lib	Suporte do cliente TXSeries CICS para C (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\mqmzf.lib	Installable services exits for C (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb.lib	Servidor para COBOL IBM (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcb.lib	Server for Micro Focus COBOL (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccbb.lib	Cliente para COBOL IBM (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccb.lib	Client for Micro Focus COBOL (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\imqs23vn.lib	Server for C++ (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\imqc23vn.lib	Client for C++ (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\imqb23vn.lib	Base for C++ (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib\imqx23vn.lib	Client MTS for C++ (32 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	Server for C (64 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	Client for C (64 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmxa.lib	Server XA interface for C (64 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\mqcxa.lib	Client XA interface for C (64 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	Client MTS for C (64 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb.lib	Servidor para COBOL IBM (64 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb.lib	Server for Micro Focus COBOL (64 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccbb.lib	Cliente para COBOL IBM (64 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb.lib	Client for Micro Focus COBOL (64 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\imqs23vn.lib	Server for C++ (64 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\imqc23vn.lib	Client for C++ (64 bits)
MQ_INSTALLATION_PATH\Tools\Lib64\imqb23vn.lib	Base for C++ (64 bits)

Tabela 25. Arquivos de biblioteca para aplicativos Windows (continuação)	
Arquivo de biblioteca	**Ambiente**
MQ_INSTALLATION_PATH\Tools\Lib64\imqx23vn.lib	Client MTS for C++ (64 bits)

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

Use `amqmdnet.dll` para compilar programas .NET.. Consulte "Compilando Programas do WebSphere MQ .NET" na página 601 na seção "Usando .NET" na página 567 para obter mais informações.

Estes arquivos são enviados para compatibilidade com liberações anteriores:

mqic32.lib
mqic32xa.lib

IBM WebSphere MQ para AIX

No IBM WebSphere MQ para AIX, deve-se vincular seu programa aos arquivos de biblioteca MQI fornecidos para o ambiente no qual você está executando seu aplicativo, além daqueles fornecidos pelo sistema operacional

Em um aplicativo não encadeado:

Tabela 26. Arquivos de biblioteca para aplicativos AIX não encadeados	
Arquivo de bilioteca	**Ambiente**
libmqm.a	Servidor para C
libmqic.a & **libmqm.a**	Cliente para C
libmqmzf.a	Saídas de serviço instalável para C
libmqmxa.a	Interface XA do servidor
libmqmxa64.a	Interface XA alternativa do servidor
libmqcxa.a	Interface XA do cliente
libmqcxa64.a	Interface XA alternativa do cliente
libmqmcbrt.o	WebSphere MQ para suporte do Micro Focus COBOL
libmqmcb.a	Servidor para COBOL
libmqicb.a	Cliente para COBOL
libimqc23ia.a	Client for C++
libimqs23ia.a	Servidor para C++

Em um aplicativo encadeado:

Tabela 27. Arquivos de biblioteca para aplicativos AIX encadeados	
Arquivo de bilioteca	**Ambiente**
libmqm_r.a	Servidor para C
libmqic_r.a & **libmqm_r.a**	Cliente para C
libmqmzf_r.a	Saídas de serviço instalável para C
libmqmxa_r.a	Interface XA do servidor
libmqmxa64_r.a	Interface XA alternativa do servidor

Tabela 27. Arquivos de biblioteca para aplicativos AIX encadeados (continuação)	
Arquivo de bilioteca	**Ambiente**
libmqcxa_r.a	Interface XA do cliente
libmqcxa64_r.a	Interface XA alternativa do cliente
libimqc23ia_r.a	Client for C++
libimqs23ia_r.a	Servidor para C++

IBM WebSphere MQ para HP-UX

No IBM WebSphere MQ para HP-UX, deve-se vincular seu programa aos arquivos de biblioteca MQI fornecidos para o ambiente no qual você está executando seu aplicativo, além daqueles fornecidos pelo sistema operacional

Plataforma IA64 (IPF)

Em um aplicativo não encadeado:

Tabela 28. Arquivos de biblioteca para aplicativos HP-UX não encadeados	
Arquivo de bilioteca	**Ambiente**
libmqm.so	Servidor para C
libmqic.so & **libmqm.so**	Cliente para C
libmqmzf.so	Saídas de serviço instalável para C
libmqmxa.so	Interface XA do servidor
libmqmxa64.so	Interface XA alternativa do servidor
libmqcxa.so	Interface XA do cliente
libmqcxa64.so	Interface XA alternativa do cliente
libimqi23ah.so	C++
libmqmcbrt.o	WebSphere MQ para suporte do Micro Focus COBOL
libmqmcb.so	Servidor para COBOL
libmqicb.so	Cliente para COBOL

Em um aplicativo encadeado:

Tabela 29. Arquivos de Biblioteca para Aplicativos HP-UX Encadeados	
Arquivo de bilioteca	**Ambiente**
libmqm_r.so	Servidor para C
libmqmzf_r.so & **libmqm_r.so**	Saídas de serviço instalável para C
libmqmxa_r.so	Interface XA do servidor
libmqmxa64_r.so	Interface XA alternativa do servidor
libmqcxa_r.so	Interface XA do cliente
libmqcxa64_r.so	Interface XA alternativa do cliente
libimqi23ah_r.so	C++

IBM WebSphere MQ for Linux

No IBM WebSphere MQ for Linux, deve-se vincular seu programa aos arquivos de bibliotecas MQI fornecidos para o ambiente no qual você está executando seu aplicativo, além daqueles fornecidos pelo sistema operacional

Em um aplicativo não encadeado:

Tabela 30. Arquivos de biblioteca para aplicativos Linux não encadeados

Arquivo de bilioteca	**Ambiente**
libmqm.so	Servidor para C
libmqic.so & **libmqm.so**	Cliente para C
libmqmzf.so	Saídas de serviço instalável para C
libmqmxa.so	Interface XA do servidor
libmqmxa64.so	Interface XA alternativa do servidor
libmqcxa.so	Interface XA do cliente
libmqcxa64.so	Interface XA alternativa do cliente
libimqc23gl.so	Client for C++
libimqs23gl.so	Servidor para C++

Em um aplicativo encadeado:

Tabela 31. Arquivos de biblioteca para aplicativos Linux encadeados

Arquivo de bilioteca	**Ambiente**
libmqm_r.so	Servidor para C
libmqic_r.so & **libmqm_r.so**	Cliente para C
libmqmzf_r.so	Saídas de serviço instalável para C
libmqmxa_r.so	Interface XA do servidor
libmqmxa64_r.so	Interface XA alternativa do servidor
libmqcxa_r.so	Interface XA do cliente
libmqcxa64_r.so	Interface XA alternativa do cliente
libimqc23gl_r.so	Client for C++
libimqs23gl_r.so	Servidor para C++

IBM WebSphere MQ Para Solaris

No IBM WebSphere MQ para Solaris, deve-se vincular seu programa aos arquivos de biblioteca MQI fornecidos para o ambiente no qual você está executando seu aplicativo, além daqueles fornecidos pelo sistema operacional

Tabela 32. Arquivos de Biblioteca para Aplicativos Solaris

Arquivo de biblioteca	**Ambiente**
libmqm.so	Servidor e cliente para C
libmqmzse.so	Para C
libmqic.so	Cliente para C

Tabela 32. Arquivos de Biblioteca para Aplicativos Solaris (continuação)	
Arquivo de biblioteca	**Ambiente**
libmqmcs.so	Serviços comuns para C
libmqmzf.so	Saídas de serviço instalável para C
libmqmxa.so	Interface XA do servidor
libmqmxa64.so	Interface XA alternativa do servidor
libmqcxa.so	Interface XA do cliente
libmqcxa64.so	Interface XA alternativa do cliente
libimqc23as.a	Client for C++
libimqs23as.a	Servidor para C++

Parâmetros comuns a todas as chamadas

Há dois tipos de parâmetros comuns a todas as chamadas: identificadores e códigos de retorno.

Usando identificadores

Todas as chamadas MQI usam um ou mais *identificadores*. Eles identificam o gerenciador de filas, a fila ou outro objeto, mensagem ou assinatura, conforme apropriado para a chamada.

Para um programa se comunicar com um gerenciador de filas, o programa deve ter um identificador exclusivo pelo qual conhece aquele gerenciador de filas. Esse identificador é chamado de *manipulação de conexões*, às vezes referido como *Hconn*. Para programas CICS , a manipulação de conexões é sempre zero. Para todas as outras plataformas ou estilos de programas, a manipulação de conexões é retornada pela chamada MQCONN ou MQCONNX quando o programa se conecta ao gerenciador de filas. Os programas passam a manipulação de conexões como um parâmetro de entrada quando eles usam as outras chamadas.

Para que um programa funcione com um objeto do WebSphere MQ , o programa deve ter um identificador exclusivo pelo qual ele sabe esse objeto. Esse identificador é chamado de *manipulação de objetos*, às vezes referido como um *Hobj*. O identificador é retornado pela chamada MQOPEN quando o programa abre o objeto para trabalhar com ele. Programas passam a manipulação de objetos como um parâmetro de entrada quando eles usam chamadas MQPUT, MQGET, MQINQ, MQSET ou MQCLOSE subsequentes.

De forma semelhante, a chamada MQSUB retorna um *identificador de assinatura* ou *Hsub*, que é usado para identificar a assinatura em chamadas MQGET, MQCB ou MQSUBRQ subsequentes e determinadas chamadas que processam propriedades de mensagens usam um *identificador de mensagem* ou *Hmsg*.

Entendendo códigos de retorno

Um código de conclusão e um código de razão são retornados como parâmetros de saída por cada chamada. Eles são conhecidos coletivamente como *códigos de retorno*.

Para mostrar se uma chamada é bem-sucedida, cada chamada retorna um *código de conclusão* quando a chamada é concluída. O código de conclusão geralmente é MQCC_OK, indicando sucesso ou MQCC_FAILED, indicando falha. Algumas chamadas podem retornar um estado intermediário, MQCC_WARNING, indicando sucesso parcial.

Cada chamada também retorna um *código de razão* que mostra a razão da falha, ou sucesso parcial, da chamada. Há vários códigos de razão, cobrindo circunstâncias como uma fila cheia, operações get não permitidas para uma fila e uma fila específica não estando definida para o gerenciador de filas. Os programas podem usar o código de razão para decidir como proceder. Por exemplo, podem solicitar aos usuários que mudem seus dados de entrada, em seguida, façam a chamada novamente ou podem retornar uma mensagem de erro ao usuário.

Quando o código de conclusão for MQCC_OK, o código de razão é sempre MQRC_NONE.

Os códigos de conclusão e de razão para cada chamada são listados com a descrição da chamada. Consulte Descrições de chamada e selecione a chamada apropriada na lista.

Para obter informações mais detalhadas, incluindo ideias para ação corretiva, consulte:

- Códigos de razão para todas as outras plataformas WebSphere MQ

Especificando buffers

O gerenciador de filas se refere a buffers somente se eles forem necessários. Se não for necessário um buffer em uma chamada ou o buffer for zero em comprimento, é possível usar um ponteiro nulo para um buffer.

Sempre use datalength ao especificar o tamanho do buffer que você requer.

Ao usar um buffer para reter a saída de uma chamada (por exemplo, para reter os dados da mensagem para uma chamada MQGET ou os valores de atributos consultados pela chamada MQINQ), o gerenciador de filas tentará retornar um código de razão se o buffer que você especificar não for válido ou estiver em armazenamento de leitura. No entanto, pode nem sempre ser capaz de retornar um código de razão.

Considerações do UNIX and Linux

Considerações de que você precisa estar ciente.

Anote os pontos a seguir ao desenvolver aplicativos UNIX and Linux.

A chamada de sistema fork em sistemas UNIX and Linux

Observe estas considerações ao usar uma chamada do sistema fork em aplicativos IBM WebSphere MQ.

Se o seu aplicativo desejar usar `fork`, o processo pai desse aplicativo deve chamar `fork` antes de fazer qualquer chamada do IBM WebSphere MQ, por exemplo, `MQCONN` ou criar um objeto do IBM WebSphere MQ usando **`ImqQueueManager`**.

Se o seu aplicativo deseja criar um processo filho após fazer quaisquer chamadas do IBM WebSphere MQ, o código do aplicativo deve usar um `fork()` com `exec()` para assegurar que o filho seja uma nova instância e não uma cópia exata do pai.

Se o seu aplicativo não usar o `exec()`, a chamada de API do IBM WebSphere MQ feita dentro do processo filho retorna MQRC_ENVIRONMENT_ERROR.

UNIX and Linux manipulação de sinal

Isso não se aplica ao WebSphere MQ para z/OS ou WebSphere MQ para Windows..

Em geral, os sistemas UNIX, Linux e IBM i foram movidos de um ambiente não encadeado (processo) para um ambiente multiencadeado. No ambiente não encadeado, algumas funções podiam ser implementadas somente usando sinais, embora a maioria dos aplicativos não precisasse estar ciente de sinais e da manipulação de sinais. No ambiente multiencadeado, as primitivas baseadas em encadeamento suportam algumas das funções que costumavam ser implementadas nos ambientes não encadeados usando sinais.

Em muitas instâncias, sinais e manipulação de sinais, embora suportados, não se ajustam bem no ambiente multiencadeado e existem diversas restrições. Isso pode ser problemático quando você estiver integrando o código do aplicativo com diferentes bibliotecas de middleware (em execução como parte do aplicativo) em um ambiente multiencadeado em que cada uma está tentando manipular sinais. A abordagem tradicional de salvar e restaurar os manipuladores de sinais (definida por processo), que funcionava quando havia apenas um encadeamento de execução dentro de um processo, não funciona em um ambiente multiencadeado. Isso ocorre porque muitos encadeamentos de execução poderiam estar tentando salvar e restaurar um recurso de todo o processo, com resultados imprevisíveis.

Aplicativos não encadeados
Não aplicável no Solaris, pois todos os aplicativos são considerados encadeados, mesmo se eles usarem apenas um único encadeamento

Cada função MQI configura seu próprio manipulador de sinais para os sinais:

SIGALRM
SIGBUS
SIGFPE
SIGSEGV
SIGILL

Manipuladores de usuários para esses são substituídos para a duração da chamada de função MQI. Outros sinais podem ser capturados da maneira normal por manipuladores escritos pelo usuário. Se você não instalar um manipulador, as ações padrão (por exemplo, ignore, core dump ou exit) são deixadas no local.

Após o WebSphere MQ manipular um sinal síncrono (SIGSEGV, SIGBUS, SIGFPE, SIGILL), ele tenta transmitir o sinal para qualquer manipulador de sinal registrado antes de fazer a chamada de função MQI.

Aplicativos encadeados
Um encadeamento é considerado conectado ao WebSphere MQ de MQCONN (ou MQCONNX) até MQDISC.

Sinais síncronos

Sinais síncronos surgem em um encadeamento específico.

Sistemas UNIX and Linux permitem de forma segura a configuração de um manipulador de sinais para esses sinais para todo o processo. No entanto, o WebSphere MQ configura seu próprio manipulador para os sinais a seguir, no processo de aplicativo, enquanto qualquer encadeamento é conectado ao WebSphere MQ:

SIGBUS
SIGFPE
SIGSEGV
SIGILL

Se você estiver escrevendo aplicativos multiencadeados, há somente um manipulador de sinais para todo o processo para cada sinal. Quando o WebSphere MQ configura seus próprios manipuladores de sinais síncronos, ele salva todos os manipuladores registrados anteriormente para cada sinal Após o WebSphere MQ manipular um dos sinais listados, o WebSphere MQ tenta chamar o manipulador de sinal que estava em vigor no momento da primeira conexão do WebSphere MQ dentro do processo. Os manipuladores registrados anteriormente serão restaurados quando todos os encadeamentos de aplicativos tiverem sido desconectados do WebSphere MQ

Como os manipuladores de sinais são salvos e restaurados pelo WebSphere MQ, os encadeamentos de aplicativos não devem estabelecer manipulações de sinais para estes sinais enquanto há qualquer possibilidade de que outro encadeamento do mesmo processo também esteja conectado ao WebSphere MQ.

Nota: Quando um aplicativo, ou uma biblioteca de middleware (em execução como parte de um aplicativo), estabelece um manipulador de sinais enquanto um encadeamento está conectado ao WebSphere MQ, o manipulador de sinais do aplicativo deve chamar o manipulador correspondente do WebSphere MQ durante o processamento desse sinal.

Ao estabelecer e restaurar manipuladores de sinais, o princípio geral é que o último manipulador de sinais a ser salvo deve ser o primeiro a ser restaurado:

- Quando um aplicativo estabelece um manipulador de sinal após se conectar ao WebSphere MQ, o manipulador de sinal anterior deve ser restaurado antes de o aplicativo se desconectar do WebSphere MQ.

- Quando um aplicativo estabelece um manipulador de sinal antes de se conectar ao WebSphere MQ, o aplicativo deve desconectar do WebSphere MQ antes de restaurar seu manipulador de sinal.

Nota: A falha em observar o princípio geral de que o último manipulador de sinais a ser salvo deve ser o primeiro a ser restaurado pode resultar em manipulação de sinais inesperada no aplicativo e, potencialmente, na perda de sinais pelo aplicativo.

Sinais assíncronos

WebSphere MQ não usa sinais assíncronos em aplicativos encadeados, a menos que sejam aplicativos clientes.

Considerações adicionais para aplicativos clientes não encadeados

WebSphere MQ manipula os seguintes sinais durante a E/S para um servidor. Esses sinais são definidos pela pilha de comunicações. O aplicativo não deve estabelecer um manipulador de sinais para esses sinais enquanto um encadeamento estiver conectado a um gerenciador de filas:

SIGPIPE (para TCP/IP)

Considerações Adicionais
Observe essas considerações ao usar a manipulação de sinal do UNIX .

Aplicativos Fastpath (confiáveis)

Os aplicativos de atalho são executados no mesmo processo que WebSphere MQ e, portanto, estão em execução no ambiente multiencadeado.

Neste ambiente, o WebSphere MQ manipula os sinais síncronos SIGSEGV, SIGBUS, SIGFPE e SIGILL Todos os outros sinais não devem ser entregues ao aplicativo Fastpath enquanto ele está conectado ao WebSphere MQ. Em vez disso, eles devem ser bloqueados ou manipulados pelo aplicativo. Se um aplicativo Fastpath interceptar tal evento, o gerenciador de filas deve ser interrompido e reiniciado ou pode ser deixado em um estado indefinido. Para obter uma lista completa das restrições para aplicativos Fastpath sob MQCONNX, consulte “Conectando-se a um gerenciador de filas usando a chamada MQCONNX” na página 212.

Chamadas de função MQI nos manipuladores de sinais

Enquanto você estiver em um manipulador de sinal, não chame uma função MQI.

Se você tentar chamar uma função MQI a partir de um manipulador de sinal enquanto outra função MQI estiver ativo, MQRC_CALL_IN_PROGRESS será retornado. Se você tentar chamar uma função MQI a partir de um manipulador de sinal enquanto nenhuma outra função MQI estiver ativa, provavelmente falhará em algum momento durante a operação devido às restrições do sistema operacional em que somente chamadas seletivas possam ser emitidas a partir de um manipulador ou dentro dele.

Para métodos destruidores C++, que podem ser chamado automaticamente durante a saída do programa, pode ser que não seja possível parar a chamada de funções MQI. Ignore qualquer erro sobre MQRC_CALL_IN_PROGRESS. Se um manipulador de sinal chamar exit (), o WebSphere MQ volta mensagens não confirmadas no ponto de sincronização como de costume e fecha quaisquer filas abertas.

Sinais durante chamadas MQI

Funções de MQI não retornam o código EINTR ou qualquer equivalente para programas aplicativos.

Se um sinal ocorrer durante uma chamada MQI e o manipulador chamar *return*, a chamada continuará a ser executada como se o sinal não tivesse acontecido. Especificamente, MQGET não pode ser interrompido por um sinal para retornar controle imediatamente para o aplicativo. Se desejar sair de um MQGET, configure a fila para GET_DISABLED; como alternativa, use um loop em torno de uma chamada para MQGET com um tempo de expiração finito (MQGMO_WAIT com gmo.WaitInterval configurado) e

use seu manipulador de sinal (em um ambiente não encadeado) ou função equivalente em um ambiente encadeado para configurar uma sinalização que interrompe o loop.

No ambiente AIX , WebSphere MQ requer que as chamadas do sistema interrompidas por sinais sejam reiniciadas. Ao estabelecer seu próprio manipulador de sinais com sigaction (2), configure o sinalizador SA_RESTART no campo sa_flags da nova estrutura de ação, caso contrário, o WebSphere MQ poderá não conseguir concluir qualquer chamada interrompida por um sinal.

Saídas de usuário e serviços instaláveis

As saídas de usuário e os serviços instaláveis que são executados como parte de um processo do WebSphere MQ em um ambiente multiencadeado têm as mesmas restrições que para os aplicativos de atalho Considere-os permanentemente conectados ao WebSphere MQ e, portanto, não usando sinais ou chamadas do sistema operacional não thread-safe.

Manipuladores de saída de VMS

Os usuários podem instalar manipuladores de saída para um aplicativo WebSphere MQ usando o serviço do sistema **SYS$DCLEXH** .

O manipulador de saída recebe controle quando uma imagem sai. Uma saída de imagem normalmente ocorre ao chamar o serviço Exit ($EXIT) ou Force Exit ($FORCEX). $FORCEX interrompe o processo de destino no modo de usuário. Em seguida, todos os manipuladores de saída de modo de usuário (estabelecidos por $DCLEXH) começam a executar na ordem inversa de estabelecimento. Para obter mais detalhes sobre os manipuladores de saída e $FORCEX, consulte o *Manual de conceitos de programação de VMS* e o *Manual do VMS System Services*.

Se você chamar uma função MQI a partir de um manipulador de saída, o comportamento da função depende da maneira que a imagem foi finalizada. Se a imagem tiver sido finalizada enquanto outra função MQI estava ativa, um `MQRC_CALL_IN_PROGRESS` será retornado.

É possível chamar uma função MQI de dentro de um manipulador de saída se nenhuma outra função MQI estiver ativa e upcalls estiverem desativados para o aplicativo WebSphere MQ . Se upcalls estiverem ativados para o aplicativo WebSphere MQ , ele falhará com o código de razão `MQRC_HCONN_ERROR`

O escopo de uma chamada MQCONN ou MQCONNX é geralmente o encadeamento que o emitiu. Se upcalls forem ativados, o manipulador de saída é executado como um encadeamento separado e as manipulações de conexões não podem ser compartilhadas.

Manipuladores de saída são iniciados no contexto interrompido do processo de destino. Depende do aplicativo assegurar se as ações tomadas por um manipulador sejam seguras e confiáveis, para o contexto interrompido de forma assíncrona do qual são chamada.

Conectando-se e desconectando-se de um gerenciador de filas

Para usar WebSphere MQ serviços de programação, um programa deve ter uma conexão com um gerenciador de filas. Use essas informações para aprender como conectar-se e desconectar-se de um gerenciador de filas.

A forma com que essa conexão é feita depende da plataforma e do ambiente no qual o programa está operacional:

z/OS em lote, WebSphere MQ para IBM i, WebSphere MQ em sistemas UNIX , WebSphere MQ em sistemas Linux e WebSphere MQ para Windows

Os programas que são executados nesses ambientes podem usar a chamada MQI MQCONN para conectar-se a, e a chamada MQDISC para desconectar-se de, um gerenciador de filas. Como alternativa, programas podem usar a chamada MQCONNX.

Os programas em lote do z/OS podem se conectar, consecutivamente ou simultaneamente, a diversos gerenciadores de filas no mesmo TCB.

IMS

A região de controle IMS é conectada a um ou mais gerenciadores de fila quando é iniciada. Essa conexão é controlada por comandos do IMS No entanto, os gravadores de programas IMS de enfileiramento de mensagens devem usar a chamada MQI MQCONN para especificar o gerenciador de filas ao qual eles desejam se conectar Eles podem usar a chamada MQDISC para desconectar do gerenciador de filas.

Após uma chamada do IMS que estabelece um ponto de sincronização e antes de processar uma mensagem para outro usuário, o adaptador do IMS assegura que o aplicativo fecha identificadores e desconecta-se do gerenciador de filas.

Os programas IMS podem se conectar consecutivamente ou simultaneamente a vários gerenciadores de filas no mesmo TCB.

CICS Transaction Server para z/OS e CICS para MVS/ESA

Os programas CICS não precisam fazer nenhum trabalho para se conectar a um gerenciador de fila porque o próprio sistema CICS está conectado.. Essa conexão é geralmente feita automaticamente na inicialização, mas você também é possível usar a transação CKQC, que éfornecido com WebSphere MQ para z/OS.

As tarefas CICS podem se conectar apenas ao gerenciador de filas ao qual a própria região CICS está conectada.

Nota: Programas CICS também podem usar as chamadas de conexão e desconexão MQI (MQCONN e MQDISC). Você pode desejar fazer isso para que seja possível portar esses aplicativos para ambientes não CICS com um mínimo de recodificação No entanto, essas chamadas *sempre* são concluídas com êxito em um ambiente do CICS Isso significa que o código de retorno pode não refletir o estado verdadeiro da conexão com o gerenciador de filas.

TXSeries para Windows e sistemas abertos

Esses programas não precisam executar nenhum trabalho para se conectar a um gerenciador de fila porque o próprio sistema CICS está conectado.. Portanto, somente uma conexão por vez é suportada. Os aplicativos CICS devem emitir uma chamada MQCONN para obter uma manipulação de conexões, e uma chamada MQDISC antes de saírem da rede....

Use os links a seguir para descobrir mais sobre como se conectar e desconectar de um gerenciador de filas:

- “Conectando-se a um gerenciador de filas usando a chamada MQCONN” na página 211
- “Conectando-se a um gerenciador de filas usando a chamada MQCONNX” na página 212
- “Desconectando programas de um gerenciador de filas usando MQDISC” na página 217

Conceitos relacionados

“Visão geral da Message Queue Interface” na página 198
Aprenda sobre os componentes de Message Queue Interface (MQI).

“Abrindo e fechando objetos” na página 218
Estas informações fornecem um insight sobre como abrir e fechar objetos do WebSphere MQ .

“Colocando mensagens em uma fila” na página 228
Use estas informações para aprender como colocar mensagens em uma fila.

“Obtendo mensagens de uma fila” na página 243
Use estas informações para aprender como obter mensagens de uma fila.

“Consultando e configurando atributos de objeto” na página 324
Atributos são as propriedades que definem as características de um objeto WebSphere MQ .

“Confirmando e fazendo backup de unidades de trabalho” na página 327
Essas informações descrevem como confirmar e restaurar quaisquer operações de GET e PUT que ocorreram em uma unidade de trabalho.

“Iniciando aplicativos IBM WebSphere MQ usando acionadores” na página 333
Aprenda sobre acionadores e como iniciar aplicativos IBM WebSphere MQ usando acionadores.

“Trabalhando com MQI e clusters” na página 351
Há opções especiais em chamadas e códigos de retorno relacionadas ao armazenamento em cluster.

Conectando-se a um gerenciador de filas usando a chamada MQCONN

Use estas informações para aprender como se conectar a um gerenciador de filas utilizando a chamada MQCONN.

Em geral, é possível conectar tanto a um gerenciador de filas específico, quanto para o gerenciador de filas padrão:

- Para IBM WebSphere MQ for z/OS, no ambiente de lote, o gerenciador de fila padrão é especificado no módulo CSQBDEFV..
- Para sistemas IBM WebSphere MQ para Windows, IBM i, UNIXe Linux , o gerenciador de fila padrão é especificado no arquivo mqs.ini ..

Como alternativa, nos ambientes z/OS MVS em lote, TSO e RRS, é possível se conectar a qualquer gerenciador de filas em um grupo de filas compartilhadas. O pedido MQCONN ou MQCONNX seleciona qualquer um dos membros ativos do grupo.

Ao se conectar a um gerenciador de filas, ele deve ser local para a tarefa. Ele deve pertencer ao mesmo sistema que o aplicativo IBM WebSphere MQ.

No ambiente IMS, o gerenciador de filas deve ser conectado à região de controle do IMS e à região dependente que o programa usa. O gerenciador de filas padrão é especificado no módulo CSQQDEFV quando o IBM WebSphere MQ for z/OS está instalado

Com o ambiente TXSeries CICS e TXSeries para Windows e AIX, o gerenciador de filas deve ser definido como um recurso XA para CICS.

Para se conectar ao gerenciador de filas padrão, faça a chamada MQCONN especificando um nome que consista inteiramente de espaços em branco ou começando com um caractere nulo (X'00').

Um aplicativo deve ser autorizado para que possa se conectar com êxito a um gerenciador de filas. Para obter mais informações, consulte Segurança.

A saída de MQCONN é:

- Uma manipulação de conexões (**Hconn**)
- Um código de conclusão
- Um código de razão

Use a manipulação de conexões em chamadas MQI subsequentes.

Se o código de razão indicar que o aplicativo já está conectado a esse gerenciador de filas, a manipulação de conexões que é retornada será a mesma que foi retornada quando o primeiro aplicativo se conectou. O aplicativo não deve emitir a chamada MQDISC nessa situação, pois o aplicativo de chamada espera permanecer conectado.

O escopo da manipulação de conexões é o mesmo que o escopo da manipulação de objetos (consulte “Abrindo objetos usando a chamada MQOPEN” na página 219).

Descrições dos parâmetros são fornecidas na descrição da chamada MQCONN em MQCONN.

A chamada MQCONN falhará se o gerenciador de filas estiver em um estado de quiesce quando a chamada for emitida ou se o gerenciador de filas estiver encerrando.

Escopo de MQCONN ou MQCONNX

O escopo de uma chamada MQCONN ou MQCONNX é geralmente o encadeamento que o emitiu. Ou seja, a manipulação de conexões retornada da chamada é válida apenas dentro do encadeamento que a emitiu. Apenas uma chamada pode ser feita a qualquer momento usando a manipulação. Se for usada a partir de um encadeamento diferente, será rejeitado como inválida. Se houver vários encadeamentos em

seu aplicativo e cada um desejar usar chamadas IBM WebSphere MQ, cada um deve emitir MQCONN ou MQCONNX.

Não é necessário que cada chamada seja feita para o gerenciador de filas mesmo quando um processo fizer várias chamadas MQCONN. No entanto, apenas uma conexão do WebSphere MQ pode ser feita de um encadeamento por vez. Como alternativa, considere "Conexões compartilhadas (independentes de encadeamento) com MQCONNX" na página 215 para permitir que várias conexões do WebSphere MQ de um único encadeamento e uma conexão do WebSphere MQ sejam usadas a partir de qualquer encadeamento.[1]

Se o seu aplicativo estiver em execução como um cliente, ele pode se conectar a mais de um gerenciador de filas em um encadeamento.

Conectando-se a um gerenciador de filas usando a chamada MQCONNX

A chamada MQCONNX é semelhante à chamada MQCONN, mas inclui opções para controlar a maneira com que a chamada funciona.

Como entrada para MQCONNX, é possível fornecer um nome do gerenciador de filas ou um nome do grupo de filas compartilhadas nos sistemas de filas compartilhadas z/OS . A saída de MQCONNX é:

- Uma manipulação de conexões (Hconn)
- Um código de conclusão
- Um código de razão

A manipulação de conexões é usada em chamadas MQI subsequentes.

Uma descrição de todos os parâmetros de MQCONNX é fornecida em MQCONNX. O campo *Options* permite configurar STANDARD_BINDING, FASTPATH_BINDING, SHARED_BINDING ou ISOLATED_BINDING para qualquer versão de MQCNO. Também é possível fazer conexões compartilhadas (independente de encadeamento) usando uma chamada MQCONNX. Consulte "Conexões compartilhadas (independentes de encadeamento) com MQCONNX" na página 215 para obter mais informações sobre estas.

MQCNO_STANDARD_BINDING

Por padrão, MQCONNX (como MQCONN) implica dois encadeamentos lógicos em que o aplicativo WebSphere MQ e o agente do gerenciador de fila local são executados em processos separados. O aplicativo WebSphere MQ solicita a operação WebSphere MQ e o agente do gerenciador de filas locais atende a solicitação. Isso é definido pela opção MQCNO_STANDARD_BINDING na chamada MQCONNX.

Se você especificar MQCNO_STANDARD_BINDING, a chamada MQCONNX usará MQCNO_SHARED_BINDING ou MQCNO_ISOLATED_BINDING, dependendo do valor do atributo de Tipo DefaultBinddo gerenciador de fila, que é definido em qm.ini ou no registro Windows .

Esse é o valor-padrão.

Se estiver vinculando à biblioteca `mqm`, uma conexão do servidor padrão usando o tipo de ligação padrão será tentada primeiro. Se o carregamento da biblioteca do servidor subjacente tiver falhado, uma conexão do cliente será tentada ao invés.

- Se a variável de ambiente MQ_CONNECT_TYPE for especificada, uma das opções a seguir poderá ser fornecida para mudar o comportamento de MQCONN ou MQCONNX se MQCNO_STANDARD_BINDING for especificado. (A exceção a isto é se MQCNO_FASTPATH_BINDING estiver especificado com MQ_CONNECT_TYPE configurado como LOCAL ou STANDARD para permitir que as conexões de atalho façam o downgrade pelo administrador sem uma mudança relacionada com o aplicativo:

[1] Ao usar aplicativos multiencadeados com sistemas IBM WebSphere MQ on UNIX and Linux , é necessário assegurar que os aplicativos tenham um tamanho de pilha suficiente para os encadeamentos Considere usar um tamanho de pilha de 256 KB, ou maior, quando aplicativos multiencadeados estiverem fazendo chamadas MQI, seja por eles mesmos ou, com outros manipuladores de sinal (por exemplo, CICS).

Value	Significado
CLIENTE	É tentada apenas uma conexão do cliente.
FASTPATH	Esse valor era suportado em liberações anteriores, mas agora ele será ignorado se for especificado.
LOCAL	É tentada apenas uma conexão do servidor. Conexões de atalho são transferidas por downgrade para uma conexão padrão do servidor.
STANDARD	Suportado para compatibilidade com liberações anteriores. Esse valor agora é tratado como LOCAL.

- Se a variável de ambiente MQ_CONNECT_TYPE não estiver configurada quando MQCONN for chamado, uma conexão padrão de servidor usando o tipo de ligação padrão será tentada. Se o carregamento da biblioteca do servidor tiver falhado, uma conexão do cliente será tentada.

MQCNO_FASTPATH_BINDING

Aplicativos confiáveis implica que o aplicativo WebSphere MQ e o agente do gerenciador de fila local se tornam o mesmo processo Como o processo do agente não precisa mais usar uma interface para acessar o gerenciador de filas, esses aplicativos tornam-se uma extensão do gerenciador de filas. Isso é definido pela opção MQCNO_FASTPATH_BINDING na chamada MQCONNX.

É necessário vincular aplicativos confiáveis às bibliotecas encadeadas do WebSphere MQ . Para obter instruções sobre como configurar um aplicativo WebSphere MQ para executar como confiável, consulte Opções MQCNO.

Esta opção fornece o melhor desempenho.

Nota: Esta opção compromete a integridade do gerenciador de filas: não há proteção de sobrescrever seu armazenamento. Isso também se aplica se o aplicativo contiver erros que possam ser expostos para mensagens e outros dados no gerenciador de filas. Considere esses problemas antes de usar essa opção.

MQCNO_SHARED_BINDING

Especifique essa opção para que o aplicativo e o agente do gerenciador de filas locais sejam executados em processos separados. Isso mantém a integridade do gerenciador de filas, ou seja, protege o gerenciador de filas de programas com erros. No entanto, o aplicativo e o agente do gerenciador de filas local compartilham alguns recursos.

Essa opção é intermediária entre MQCNO_FASTPATH_BINDING e MQCNO_ISOLATED_BINDING, tanto em termos de proteger a integridade do gerenciador de filas quanto em termos de desempenho das chamadas do MQI.

MQCNO_SHARED_BINDING será ignorado se o gerenciador de filas não suportar esse tipo de ligação. O processamento continuará, embora a opção não tenha sido especificada.

Se um aplicativo tiver sido conectado ao gerenciador de filas locais usando MQCNO_SHARED_BINDING, o gerenciador de filas poderá ser parado enquanto o aplicativo estiver em execução. Se o gerenciador de filas for reiniciado enquanto o aplicativo ainda estiver em execução, a tentativa de iniciar o gerenciador de filas falhará com o erro AMQ7018 já que o aplicativo ainda estará mantendo os recursos necessários pelo gerenciador de filas.

Para iniciar o gerenciador de filas, deve-se parar o aplicativo.

MQCNO_ISOLATED_BINDING

Especifique essa opção para que o aplicativo e o agente do gerenciador de filas locais sejam executados em processos separados, como para MQCNO_SHARED_BINDING. No entanto, neste caso,

o processo do aplicativo e o agente do gerenciador de filas local são isolados um do outro no que eles não compartilhem recursos.

Esta é a opção mais segura para proteger a integridade do gerenciador de filas, mas ela fornece o desempenho mais lento de chamadas MQI.

MQCNO_ISOLATED_BINDING será ignorada se o gerenciador de filas não suportar esse tipo de ligação. O processamento continuará, embora a opção não tenha sido especificada.

MQCNO_CLIENT_BINDING

Especifique essa opção para que o aplicativo tente apenas uma conexão do cliente. Essa opção tem as seguintes limitações:

- MQCNO_CLIENT_BINDING é rejeitado no z/OS com MQRC_OPTIONS_ERROR.
- MQCNO_CLIENT_BINDING é rejeitado com MQRC_OPTIONS_ERROR se for especificado com qualquer opção de ligação MQCNO que não MQCNO_STANDARD_BINDING.
- MQCNO_CLIENT_BINDING não está disponível para Java porque ele tem seus próprios mecanismos para escolher o tipo de ligação.
- V 7.5.0.7 Antes da IBM WebSphere MQ Version 7.5.0, Fix Pack 7, MQCNO_CLIENT_BINDING não estava disponível para .NET, pois tinha seus próprios mecanismos para escolher o tipo de ligação. A partir da Version 7.5.0, Fix Pack 7, a restrição sobre o uso de .NET para MQCNO_CLIENT_BINDING foi removida.
- Se a variável de ambiente MQ_CONNECT_TYPE não for configurada quando MQCONNX for chamado, uma conexão do servidor padrão usando o tipo de ligação padrão será tentada.. Se o carregamento da biblioteca do servidor tiver falhado, uma conexão do cliente será tentada.

MQCNO_LOCAL_BINDING

Especifique essa opção para que o aplicativo tente uma conexão do servidor. Se MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING ou MQCNO_SHARED_BINDING também estiverem especificadas, então ao invés a conexão será desse tipo e será documentada nesta seção. Caso contrário, uma conexão do servidor padrão será tentada usando o tipo de ligação padrão. MQCNO_LOCAL_BINDING tem as seguintes limitações:

- MQCNO_LOCAL_BINDING é ignorado no z/OS.
- MQCNO_LOCAL_BINDING é rejeitado com MQRC_OPTIONS_ERROR se for especificado com qualquer opção de reconexão MQCNO além de MQCNO_RECONNECT_AS_DEF.
- MQCNO_LOCAL_BINDING não está disponível para Java porque ele tem seus próprios mecanismos para escolher o tipo de ligação.
- V 7.5.0.7 Antes da IBM WebSphere MQ Version 7.5.0, Fix Pack 7, MQCNO_LOCAL_BINDING não estava disponível para .NET, pois tinha seus próprios mecanismos para escolher o tipo de ligação. Em Version 7.5.0, Fix Pack 7, a restrição sobre o uso de .NET para MQCNO_LOCAL_BINDING é removida.
- Se a variável de ambiente MQ_CONNECT_TYPE não for configurada quando MQCONNX for chamado, uma conexão do servidor padrão usando o tipo de ligação padrão será tentada.. Se o carregamento da biblioteca do servidor tiver falhado, uma conexão do cliente será tentada.

No z/OS , essas opções são tolerados, mas apenas uma conexão de limite padrão é executada MQCNO Versão 3, para z/OS, permite quatro opções alternativas:

MQCNO_SERIALIZE_CONN_TAG_QSG

Isso permite que um aplicativo solicite que apenas uma instância de um aplicativo seja executada em qualquer momento em um grupo de filas compartilhadas. Isso é obtido registrando o uso de uma tag de conexão com um valor que é especificado ou derivado pelo aplicativo. A tag é uma sequência de caracteres de 128 bytes especificada no MQCNO Versão 3.

MQCNO_RESTRICT_CONN_TAG_QSG

Isso é usado quando um aplicativo consiste em mais de um processo (ou um TCB), cada um dos quais pode se conectar a um gerenciador de filas. A conexão é permitida apenas se não houver uso atual da identificação ou se o aplicativo de pedido estiver dentro do mesmo escopo de processamento. Esse é o espaço de endereço do MVS dentro do mesmo grupo de filas compartilhadas que o proprietário da tag

MQCNO_SERIALIZE_CONN_TAG_Q_MGR

Isso é semelhante a MQCNO_SERIALIZE_CONN_TAG_QSG, mas apenas o gerenciador de filas local é interrogado para ver se a identificação solicitada já está em uso.

MQCNO_RESTRICT_CONN_TAG_Q_MGR

Isso é semelhante a MQCNO_RESTRICT_CONN_TAG_QSG, mas apenas o gerenciador de filas local é interrogado para ver se a identificação solicitada já está em uso.

Restrições para aplicativos confiáveis

As restrições a seguir se aplicam a aplicativos confiáveis:

- Deve-se desconectar explicitamente aplicativos confiáveis do gerenciador de filas.
- Deve-se parar aplicativos confiáveis antes de terminar o gerenciador de filas com o comando endmqm.
- Não deve-se usar sinais assíncronos e interrupções do cronômetro (como `sigkill`) com MQCNO_FASTPATH_BINDING.
- Em todas as plataformas, um encadeamento dentro de um aplicativo confiável não pode se conectar a um gerenciador de filas enquanto outro encadeamento no mesmo processo estiver conectado a um gerenciador de filas diferente.
- Em sistemas WebSphere MQ on UNIX and Linux , deve-se usar mqm como o userID e o groupID efetivos para todas as chamadas MQI. É possível mudar esses IDs antes de fazer uma chamada não MQI que requeira autenticação (por exemplo, abrindo um arquivo), mas *deve*-se mudá-los de volta para mqm antes de fazer a próxima chamada MQI.
- No WebSphere MQ para HP-UX, aplicativos de atalho multiencadeados provavelmente precisarão configurar um tamanho de pilha maior do que o padrão. Use um tamanho de 256 KB.
- No WebSphere MQ para Windows aplicativos confiáveis de 64 bits não são suportados. Se você tentar executar um aplicativo confiável de 64 bits, ele feito downgrade para uma conexão limite padrão.
- Em sistemas WebSphere MQ on UNIX and Linux aplicativos confiáveis de 32 bits não são suportados. Se você tentar executar um aplicativo confiável de 32 bits, será feito downgrade para uma conexão limite padrão.

Conexões compartilhadas (independentes de encadeamento) com MQCONNX

Use estas informações para aprender sobre conexões Compartilhadas com MQCONNX e algumas observações de uso a considerar.

Nota: Não é suportado no WebSphere MQ para z/OS

Em plataformas WebSphere MQ diferentes de WebSphere MQ para z/OS, uma conexão feita com `MQCONN` está disponível apenas para o encadeamento que fez a conexão. As opções na chamada MQCONNX permitem criar uma conexão que pode ser compartilhada por todos os encadeamentos em um processo. Se o seu aplicativo estiver em execução em um ambiente transacional que requer que chamadas MQI sejam emitidas no mesmo encadeamento, deve-se usar a opção padrão a seguir:

MQCNO_HANDLE_SHARE_NONE
Cria uma conexão não compartilhada.

Na maioria dos outros ambientes, é possível usar uma das opções de conexão compartilhada independente do encadeamento:

MQCNO_HANDLE_SHARE_BLOCK
Cria uma conexão compartilhada. Em uma conexão MQCNO_HANDLE_SHARE_BLOCK, se a conexão estiver atualmente em uso por uma chamada MQI em outro encadeamento, a chamada MQI espera até que a chamada MQI atual tenha sido concluída.

MQCNO_HANDLE_SHARE_NO_BLOCK
Cria uma conexão compartilhada. Em uma conexão MQCNO_HANDLE_SHARE_NO_BLOCK, se a conexão estiver atualmente em uso por uma chamada MQI em outro encadeamento, a chamada MQI falhará imediatamente com uma razão MQRC_CALL_IN_PROGRESS.

Exceto para o ambiente MTS (Microsoft Transaction Server), o valor padrão é MQCNO_HANDLE_SHARE_NONE. No ambiente do MTS, o valor padrão é MQCNO_HANDLE_SHARE_BLOCK.

Uma manipulação de conexões é retornada da chamada MQCONNX. O identificador pode ser usado pelas chamadas MQI subsequentes a partir de qualquer encadeamento no processo, associando essas chamadas ao identificador retornado de MQCONNX. As chamadas MQI que usam um único identificador compartilhado são serializadas entre encadeamentos.

Por exemplo, a sequência de atividade a seguir é possível com um identificador compartilhado:

1. O encadeamento 1 emite MQCONNX e obtém um identificador compartilhado *h1*
2. O encadeamento 1 abre uma fila e emite uma solicitação get usando *h1*
3. O encadeamento 2 emite uma solicitação put usando *h1*
4. O encadeamento 3 emite uma solicitação put usando *h1*
5. O encadeamento 2 emite MQDISC usando *h1*

Enquanto o identificador estiver em uso por algum encadeamento, o acesso à conexão estará indisponível para outros encadeamentos. Em circunstâncias em que é aceitável que um encadeamento espere a conclusão de qualquer chamada anterior de outro encadeamento, use MQCONNX com a opção MQCNO_HANDLE_SHARE_BLOCK.

No entanto, o bloqueio pode causar dificuldades. Suponha que na etapa “2” na página 216, o encadeamento 1 emita uma solicitação get que espera mensagens que podem ainda não ter chegado (um get com espera). Nesse caso, os encadeamentos 2 e 3 também ficam esperando (bloqueados) pelo tempo que a solicitação get no encadeamento 1 levar. Se preferir que uma chamada MQI retorne com um erro se outra chamada MQI já estiver em execução no identificador, use MQCONNX com a opção MQCNO_HANDLE_SHARE_NO_BLOCK.

Notas de uso da conexão compartilhada

1. Qualquer identificador de objeto (Hobj) criado abrindo um objeto está associado a um Hconn; portanto, para um Hconn compartilhado, os Hobjs também são compartilhados e utilizáveis por qualquer encadeamento que estiver usando o Hconn. De forma semelhante, qualquer unidade de trabalho iniciada sob um Hconn está associada a esse Hconn; de forma que este também seja compartilhado entre encadeamentos com o Hconn compartilhado.
2. *Qualquer* encadeamento pode chamar MQDISC para desconectar um Hconn compartilhado, não somente o encadeamento que chamou o MQCONNX correspondente. O MQDISC finaliza o Hconn tornando-o indisponível para todos os encadeamentos.
3. Um único encadeamento pode usar vários Hconns compartilhados em série, por exemplo, usar MQPUT para colocar uma mensagem sob um Hconn compartilhado, em seguida, colocar outra mensagem usando outro Hconn compartilhado, com cada operação sendo sob uma unidade de trabalho local diferente.
4. Hconns compartilhados não podem ser usados em uma unidade de trabalho global.

Uso de opções de chamada MQCONNX com MQ_CONNECT_TYPE

Use estas informações para entender as diferentes opções de chamadas MQCONNX como elas são usadas com MQ_CONNECT_TYPE.

No WebSphere MQ para IBM i, WebSphere MQ para Windowse WebSphere MQ em sistemas UNIX and Linux , é possível usar a variável de ambiente, MQ_CONNECT_TYPE em combinação com o tipo de ligação especificado no campo *Options* da estrutura MQCNO usada em uma chamada MQCONNX.

Tabela 33. A variável de ambiente MQ_CONNECT_TYPE

opção de chamada MQCONNX	**variável de ambiente MQ_CONNECT_TYPE**	**Resultado**
STANDARD	UNDEFINED	STANDARD
STANDARD	STANDARD	STANDARD
STANDARD	FASTPATH	STANDARD
STANDARD	CLIENTE	CLIENTE
STANDARD	LOCAL	STANDARD

Se o MQCNO_STANDARD_BINDING não for especificado, será possível usar o MQCNO_NONE, que é padronizado para MQCNO_STANDARD_BINDING.

Desconectando programas de um gerenciador de filas usando MQDISC

Use essas informações para aprender sobre desconectar programas de um gerenciador de filas usando MQDISC.

Quando um programa que foi conectado a um gerenciador de filas usando a chamada MQCONN ou MQCONNX tiver concluído toda a interação com o gerenciador de filas, ele interromperá a conexão usando a chamada MQDISC, exceto:

- No CICS Transaction Server for z/OS aplicativos, em que a chamada é opcional, a menos que MQCONNX foi usado e você deseja eliminar a tag de conexão antes que o aplicativo termine.
- No WebSphere MQ para IBM i , em que, ao efetuar sign off do sistema operacional, uma chamada MQDISC implícita é feita

Como entrada para a chamada MQDISC, deve-se fornecer a manipulação de conexões (Hconn) que foi retornada pelo MQCONN ou MQCONNX ao se conectar ao gerenciador de filas.

Exceto no CICS no z/OS, após MQDISC ser chamado o identificador de conexão (Hconn) não é mais válido e não é possível emitir nenhuma chamada MQI adicional até que você chame MQCONN ou MQCONNX novamente. MQDISC faz uma chamada MQCLOSE implícita para quaisquer objetos que ainda estiverem abertos usando essa manipulação.

Se você usar MQCONNX para se conectar no WebSphere MQ para z/OS, MQDISC também terminará o escopo da tag de conexão estabelecida pelo MQCONNX. No entanto, em um aplicativo CICS, IMSou RRS, se houver uma unidade de recuperação ativa associada a uma tag de conexão, o MQDISC será rejeitado com um código de razão de MQRC_CONN_TAG_NOT_LIBERADO

Descrições dos parâmetros são fornecidas na descrição da chamada MQDISC no MQDISC.

Quando nenhum MQDISC for emitido

Uma conexão padrão não compartilhada (Hconn) será limpa quando a criação do encadeamento finalizar. Uma conexão compartilhada será implicitamente restaurada e desconectada apenas quando o processo inteiro for finalizado. Se o encadeamento que criou a Hconn compartilhada é finalizada enquanto o Hconn ainda existe, o Hconn ainda será utilizável.

Verificação de autoridade

As chamadas MQCLOSE e MQDISC geralmente não executam nenhuma verificação de autoridade.

No curso normal de eventos, uma tarefa que tem autoridade para abrir ou se conectar a um objeto WebSphere MQ é fechada ou desconectada desse objeto. Mesmo se a autoridade de uma tarefa que

se conectou ou abriu um objeto WebSphere MQ for revogada, as chamadas MQCLOSE e MQDISC serão aceitas.

Abrindo e fechando objetos

Estas informações fornecem um insight sobre como abrir e fechar objetos do WebSphere MQ .

Para executar qualquer uma das operações a seguir, você deve primeiro *abrir* o objeto WebSphere MQ relevante:

- Colocar mensagens em uma fila
- Obter (procurar ou recuperar) mensagens de uma fila
- Defina os atributos de um objeto
- Consulta sobre os atributos de qualquer objeto

Use a chamada MQOPEN para abrir o objeto, usando as opções da chamada para especificar o que você deseja fazer com o objeto. A única exceção é se você desejar colocar uma única mensagem em uma fila e então fechar a fila imediatamente. Neste caso, é possível ignorar a etapa de *abrir* usando a chamada MQPUT1 (veja "Colocando uma mensagem em uma fila usando a chamada MQPUT1" na página 237).

Antes de abrir um objeto usando a chamada MQOPEN, deve-se conectar o programa a um gerenciador de filas. Isso é explicado em detalhes, para todos os ambientes, em "Conectando-se e desconectando-se de um gerenciador de filas" na página 209.

Há quatro tipos de objeto do WebSphere MQ que podem ser abertos:

- Fila
- Lista de Nomes
- Definição de processo
- Gerenciador de Filas

Todos esses objetos são abertos de maneira semelhante usando a chamada MQOPEN. Para obter mais informações sobre objetos WebSphere MQ , consulte Objetos.

É possível abrir o mesmo objeto mais de uma vez, e, a cada vez, você obterá uma nova manipulação de objetos.Você pode desejar procurar mensagens em uma fila usando um manipulador e remover mensagens da mesma fila usando outro manipulador. Isso salva usando os recursos para fechar e reabrir o mesmo objeto. Também é possível abrir uma fila para procurar *e* remover mensagens ao mesmo tempo.

Além disso, é possível abrir vários objetos com uma única chamada MQOPEN e feche-os usando MQCLOSE. Veja "Listas de distribuição" na página 238 para obter informações sobre como fazer isso.

Ao tentar abrir um objeto, o gerenciador de filas verifica se você está autorizado a abrir esse objeto para as opções especificadas na chamada MQOPEN.

Os objetos são fechados automaticamente quando um programa se desconecta do gerenciador de filas. No ambiente do IMS , a desconexão é forçada quando um programa inicia o processamento para um novo usuário após uma chamada GU (get unique) IMS . Na plataforma IBM i, os objetos são fechados automaticamente quando uma tarefa é concluída.

É uma prática recomendada de programação fechar objetos que você abriu. Use a chamada MQCLOSE para fazer isso.

Use os seguintes links para descobrir mais sobre como abrir e fechar objetos:

- "Abrindo objetos usando a chamada MQOPEN" na página 219
- "Criando filas dinâmicas" na página 226
- "Abrindo filas remotas" na página 227
- "Fechando objetos usando a chamada MQCLOSE" na página 227

Conceitos relacionados

"Visão geral da Message Queue Interface" na página 198

Aprenda sobre os componentes de Message Queue Interface (MQI).

"Conectando-se e desconectando-se de um gerenciador de filas" na página 209
Para usar WebSphere MQ serviços de programação, um programa deve ter uma conexão com um gerenciador de filas. Use essas informações para aprender como conectar-se e desconectar-se de um gerenciador de filas.

"Colocando mensagens em uma fila" na página 228
Use estas informações para aprender como colocar mensagens em uma fila.

"Obtendo mensagens de uma fila" na página 243
Use estas informações para aprender como obter mensagens de uma fila.

"Consultando e configurando atributos de objeto" na página 324
Atributos são as propriedades que definem as características de um objeto WebSphere MQ .

"Confirmando e fazendo backup de unidades de trabalho" na página 327
Essas informações descrevem como confirmar e restaurar quaisquer operações de GET e PUT que ocorreram em uma unidade de trabalho.

"Iniciando aplicativos IBM WebSphere MQ usando acionadores" na página 333
Aprenda sobre acionadores e como iniciar aplicativos IBM WebSphere MQ usando acionadores.

"Trabalhando com MQI e clusters" na página 351
Há opções especiais em chamadas e códigos de retorno relacionadas ao armazenamento em cluster.

Abrindo objetos usando a chamada MQOPEN

Use essas informações para aprender sobre como abrir objetos usando a chamada MQOPEN.

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

- Uma manipulação de conexões. Para aplicativos CICS no z/OS, é possível especificar a constante MQHC_DEF_HCONN (que possui o valor zero) ou usar o identificador de conexão retornado pela chamada MQCONN ou MQCONNX. Para outros programas, sempre use a manipulação de conexões retornada pela chamada MQCONN ou MQCONNX.
- Uma descrição do objeto que você deseja abrir, usando a estrutura do descritor de objeto (MQOD).
- Uma ou mais opções que controlam a ação da chamada.

A saída de MQOPEN é:

- Uma manipulação de objetos que representa seu acesso ao objeto. Use-o em entrada para quaisquer chamadas MQI subsequentes.
- Uma estrutura do descritor de objeto modificado, se você estiver criando uma fila dinâmica (e for suportada em sua plataforma).
- Um código de conclusão.
- Um código de razão.

Escopo de uma manipulação de objetos

O escopo de uma manipulação de objetos (Hobj) é o mesmo que o escopo de uma manipulação de conexões (Hconn).

Isso é coberto em "Escopo de MQCONN ou MQCONNX" na página 211 e "Conexões compartilhadas (independentes de encadeamento) com MQCONNX" na página 215. No entanto, há considerações adicionais em alguns ambientes:

CICS
Em um programa CICS , é possível usar o identificador apenas dentro da mesma tarefa CICS a partir da qual você fez a chamada MQOPEN

IMS e z/OS em lote
Nos ambientes IMS e em lote, é possível usar o identificador na mesma tarefa, mas não em nenhuma subtarefa.

Descrições dos parâmetros da chamada MQOPEN são fornecidas em MQOPEN.

As seções a seguir descrevem as informações que deve-se fornecer como entrada para MQOPEN.

Identificando objetos (a estrutura MQOD)

Use a estrutura MQOD para identificar o objeto que deseja abrir. Essa estrutura é um parâmetro de entrada para a chamada MQOPEN. (A estrutura é modificada pelo gerenciador de filas quando a chamada MQOPEN é usada para criar uma fila dinâmica.)

Para obter detalhes completos da estrutura MQOD, consulte MQOD.

Para obter informações sobre como usar a estrutura MQOD para listas de distribuição, consulte "Usando a estrutura MQOD" na página 239 em "Listas de distribuição" na página 238.

Resolução do Nome

Como a chamada MQOPEN resolve nomes de fila e de gerenciador de filas.

Nota: Um alias do gerenciador de filas é uma definição de fila remota sem um campo RNAME.

Ao abrir uma fila do WebSphere MQ , a chamada MQOPEN executa uma função de resolução de nome no nome da fila especificado. Isso determina em qual fila o gerenciador de filas executa operações subsequentes. Isso significa que quando você especifica o nome de uma fila de alias ou uma fila remota em seu descritor de objeto (MQOD), a chamada resolve o nome ou para uma fila local ou para uma fila de transmissão. Se uma fila é aberta para qualquer tipo de entrada, procurar ou configurar, ele resolve para uma fila local, se houver uma, e falhará se não houver nenhuma. Ele resolve para uma fila que não seja local apenas se ele for aberto somente para saída, consulta ou saída e consulta. Consulte Tabela 34 na página 220 para obter uma visão geral do processo de resolução de nome. O nome que você fornece em *ObjectQMgrName* será resolvido *antes* que em *ObjectName*.

O Tabela 34 na página 220 também mostra como é possível usar uma definição local de uma fila remota para definir um alias para o nome de um gerenciador de filas. Isso permite selecionar qual fila de transmissão é usada quando se coloca mensagens em uma fila remota, portanto, é possível, por exemplo, usar uma única fila de transmissão para mensagens destinadas a muitos gerenciadores de filas remotas.

Para usar a tabela a seguir, primeiro leia a duas colunas à esquerda, sob o título **Entrada para MQOD** e selecione o caso apropriado. Em seguida, leia toda a linha correspondente, seguindo todas as instruções. Seguindo as instruções nas colunas **Nomes resolvidos**, é possível retornar para as colunas **Entrada para MQOD** e inserir valores conforme orientado ou sair da tabela com os resultados fornecidos. Por exemplo, pode ser necessário inserir o *ObjectName*.

Tabela 34. Resolvendo nomes de filas ao usar MQOPEN

Entrada para MQOD		**Nomes resolvidos**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Fila de transmissão**
Gerenciador de filas em branco ou local	Fila local sem o atributo CLUSTER	Gerenciador de filas locais	Entrada de *ObjectName*	Não aplicável (fila local usada)
Gerenciador de filas em branco	Fila local com o atributo CLUSTER	Gerenciador de filas de cluster selecionado do gerenciamento de carga de trabalho ou gerenciador de filas de cluster específico selecionado em PUT	Entrada de *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE e fila local usada SYSTEM.QSG.TRANSMIT.QUEUE (veja a nota)
Gerenciador de filas locais	Fila local com o atributo CLUSTER	Gerenciador de filas locais	Entrada de *ObjectName*	Não aplicável (fila local usada)

Tabela 34. Resolvendo nomes de filas ao usar MQOPEN (continuação)

Entrada para MQOD		**Nomes resolvidos**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Fila de transmissão**
Gerenciador de filas em branco ou local	Fila modelo	Gerenciador de filas locais	Nome gerado	Não aplicável (fila local usada)
Gerenciador de filas em branco ou local	Fila de alias com ou sem o atributo CLUSTER	Execute a resolução de nome novamente com *ObjectQMgrName* inalterado, e a entrada *ObjectName* configurada para o *BaseQName* no objeto de definição de fila de alias. Não deve ser resolvido para um alias definido localmente, no qual o *ObjectQMgrName* é especificado, mas pode ser resolvido para um alias em cluster (hospedado em outros gerenciadores de filas) no qual o *ObjectQMgrName* está em branco.		
Gerenciador de filas locais	Fila de alias com o atributo CLUSTER	O alias não deve ser resolvido para uma fila de clusters que não é definida localmente ou uma fila de cluster que possui o mesmo *ObjectName* como o alias.		
Gerenciador de filas em branco	Fila de alias com o atributo CLUSTER	O alias pode ser resolvido para uma fila de clusters com o mesmo *ObjectName* como o alias.		
Gerenciador de filas em branco ou local	Definição local de uma fila remota	Execute a resolução de nome novamente com *ObjectQMgrName* configurado como *RemoteQMgrName* e *ObjectName* configurado como *RemoteQName*. Não é necessário resolver as filas remotas		Nome do atributo *XmitQName*, se não estiver em branco; caso contrário, *RemoteQMgrName* no objeto de definição de fila remota. SYSTEM.QSG.TRANSMIT.QUEUE (veja a nota)

Tabela 34. Resolvendo nomes de filas ao usar MQOPEN (continuação)

Entrada para MQOD		**Nomes resolvidos**		
ObjectQMgrName	***ObjectName***	***ObjectQMgrName***	***ObjectName***	**Fila de transmissão**
Gerenciador de filas em branco	Nenhum objeto local correspondente; fila de cluster localizada	Gerenciador de filas de cluster selecionado do gerenciamento de carga de trabalho ou gerenciador de filas de cluster específico selecionado em PUT	Entrada de *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (veja a nota)
Gerenciador de filas em branco ou local	Nenhum objeto local correspondente; fila de cluster não localizada		Erro, fila não foi localizada	Não-aplicável
Nome do gerenciador de filas no mesmo grupo de filas compartilhadas que o gerenciador de ficas locais	Fila compartilhada local	Gerenciador de filas locais	Entrada de *ObjectName*	Não-aplicável
Nome de uma fila de transmissão local	(Não resolvido)	Insira *ObjectQMgrName*	Entrada de *ObjectName*	Insira *ObjectQMgrName* SYSTEM.QSG.TRANSMIT.QUEUE (veja a nota)
Definição de alias do gerenciador de filas (*RemoteQMgrName* pode ser o gerenciador de filas locais)	(Não resolvido, fila remota)	Execute resolução de nome novamente com *ObjectQMgrName* configurado como *RemoteQMgrName*. Não se deve resolver para filas remotas	Entrada de *ObjectName*	Nome do atributo *XmitQName*, se não estiver em branco; caso contrário, *RemoteQMgrName* no objeto de definição de fila remota. SYSTEM.QSG.TRANSMIT.QUEUE (veja a nota)
O gerenciador de filas não é o nome de qualquer objeto local; gerenciadores de filas do cluster ou alias do gerenciador de filas localizado	(Não resolvido)	*ObjectQMgrName* ou gerenciador de filas do cluster específico selecionado em PUT	Entrada de *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (veja a nota)
O gerenciador de filas não é o nome de nenhum objeto local; nenhum objeto de cluster foi localizado	(Não resolvido)	Insira *ObjectQMgrName*	Entrada de *ObjectName*	O atributo *DefXmitQName* do gerenciador de filas no qual o *DefXmitQName* é suportado. SYSTEM.QSG.TRANSMIT.QUEUE (veja a nota)

Notes:

1. *BaseQName* é o nome da fila base da definição da fila de alias.
2. *RemoteQName* é o nome da fila remota da definição local da fila remota.
3. *RemoteQMgrName* é o nome do gerenciador de filas remotas da definição local da fila remota.
4. *XmitQName* é o nome da fila de transmissão da definição local da fila remota.
5. Ao usar gerenciadores de filas do WebSphere MQ para z/OS que fazem parte de um grupo de filas compartilhadas (QSG), o nome do QSG pode ser usado em vez do nome do gerenciador de fila local no Tabela 34 na página 220.

   Se o gerenciador de filas locais não puder abrir a fila de destino ou colocar uma mensagem na fila, a mensagem será transferida para o Nome do ObjectQMgrespecificado por meio da fila intragrupo ou de um canal do WebSphere MQ .
6. Na coluna *ObjectName* da tabela, CLUSTER se refere aos atributos CLUSTER e CLUSNL da fila.
7. O SYSTEM.QSG.TRANSMIT.QUEUE será usado se os gerenciadores de filas locais e remotas estiverem no mesmo grupo de filas compartilhadas; o enfileiramento intragrupo é ativado.
8. Se você tiver designado uma fila de transmissão do cluster diferente para cada canal do emissor de clusters, SYSTEM.CLUSTER.TRANSMIT.QUEUE não poderá ser o nome da fila de transmissão do cluster. Para obter mais informações sobre várias filas de transmissão do cluster, consulte Armazenamento em Cluster: Planejando como Configurar Filas de Transmissão do Cluster .
9. Na situação em que o gerenciador de filas não é o nome de nenhum objeto local; gerenciadores de filas do cluster ou alias do gerenciador de filas localizados.

   Quando você tiver fornecido um nome do gerenciador de filas usando **ObjectQMgrName** e houver múltiplos canais de cluster com diferentes nomes de cluster conhecidos pelo gerenciador de filas locais que atingem esse destino, qualquer um desses canais pode ser usado para mover a mensagem, independentemente do nome do cluster da fila de destino.

   Isso pode ser inesperado, se você estava antecipando mensagens para essa fila somente para serem enviadas por meio de um canal que tem o mesmo nome do cluster que a fila.

   No entanto, o **ObjectQMgrName** tem precedência neste caso e o balanceamento de carga de trabalho do cluster leva em consideração todos os canais que podem atingir esse gerenciador de filas, independentemente do nome do cluster em que eles estão.

Abrir uma fila de alias também abre a fila de base para a qual o alias resolve, e abrir uma fila remota também abre a fila de transmissão. Portanto, não é possível excluir a fila que você especifica ou a fila para a qual ela resolve enquanto a outra está aberta.

Enquanto uma fila de alias é incapaz de resolver para outra fila de alias definida localmente (compartilhada em um cluster ou não), resolver para uma fila de alias do cluster definido remotamente é permitido e, portanto, pode ser especificado como a fila base.

O nome da fila resolvida e o nome do gerenciador de filas resolvido são armazenados nos campos *ResolvedQName* e *ResolvedQMgrName* na MQOD.

Para obter mais informações sobre a resolução de nome em um ambiente de enfileiramento distribuído, consulte O que é a resolução de nome de fila?.

Usando as opções da chamada MQOPEN

No parâmetro *Options* da chamada MQOPEN, deve-se escolher um ou mais opções para controlar o acesso ao objeto que estiver abrindo que foi atribuído a você. Com essas opções, é possível:

- Abrir uma fila e especificar que todas as mensagens colocadas nessa fila devem ser direcionadas para a mesma instância dela
- Abra uma fila para permitir que você coloque mensagens nela
- Abra uma fila para permitir que você procure mensagens nela
- Abra uma fila para permitir que você remova as mensagens dela

- Abra um objeto para permitir que você consulte sobre e configure seus atributos (mas será possível configurar os atributos somente de filas)
- Abra um tópico ou uma sequência de tópicos para publicar mensagens nele
- Informações de contexto associada com uma mensagem
- Denomine um identificador de usuários alternativo para ser usado para verificações de segurança
- Controle a chamada se o gerenciador de filas estiver em um estado de quiesce

Opção MQOPEN para fila de cluster
A ligação usada para a manipulação de fila é obtida a partir do atributo da fila *DefBind*, que pode obter o valor MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXED ou MQBND_BIND_ON_GROUP.

Para rotear todas as mensagens colocadas em uma fila usando MQPUT para o mesmo gerenciador de filas pela mesma rota, use a opção MQOO_BIND_ON_OPEN na chamada MQOPEN.

Para especificar que um destino deve ser selecionado no tempo do MQPUT, ou seja, mensagem por mensagem, use a opção MQOO_BIND_NOT_FIXED na chamada MQOPEN.

Para especificar que todas as mensagens em um grupos de mensagens colocados em uma fila usando MQPUT sejam alocadas para a mesma instância de destino, use a opção MQOO_BIND_ON_GROUP na chamada MQOPEN.

MQOO_BIND_ON_OPEN ou MQOO_BIND_ON_GROUP deve ser especificado ao usar grupos de mensagens com clusters para assegurar que todas as mensagens no grupo sejam processadas no mesmo destino.

Se você não especificar nenhuma dessas opções, o padrão, MQOO_BIND_AS_Q_DEF, será usado.

Se você especificar o nome de um gerenciador de filas no MQOD, a fila nesse gerenciador de filas será selecionada. Se o nome do gerenciador de filas estiver em branco, qualquer instância poderá ser selecionada. Consulte o "MQOPEN e clusters" na página 352 para obter informações adicionais.

Se você abrir uma fila de clusters usando uma definição QALIAS, alguns atributos da fila são definidos pela fila de alias e não a fila base. Os atributos de cluster estão entre os atributos da definição de fila base que são substituídos pela fila de alias. Por exemplo, no fragmento a seguir, a fila de clusters é aberta com MQOO_BIND_NOT_FIXED e não com MQOO_BIND_ON_OPEN. A definição de fila de clusters é anunciada em todo o cluster, a definição de fila de alias é local para o gerenciador de filas.

```
DEFINE QLOCAL(CLQ1)  CLUSTER(MYCLUSTER) DEFBIND(OPEN) REPLACE
DEFINE QALIAS(ACLQ1) TARGET(CLQ1) DEFBIND(NOTFIXED) REPLACE
```

Opção de MQOPEN para colocação de mensagens
Para abrir uma fila ou um tópico para colocar mensagens nele, use a opção MQOO_OUTPUT.

Opção MQOPEN para procurar mensagens
Para abrir uma fila de modo que seja possível *procurar* as mensagens nele, use a chamada MQOPEN com a opção MQOO_BROWSE.

Isso cria um *cursor de procura* que o gerenciador de filas usa para identificar a próxima mensagem na fila. Para obter informações adicionais, consulte "Procurando mensagens em uma fila" na página 275.

Nota:

1. Não é possível procurar as mensagens em uma fila remota; não abra uma fila remota usando a opção MQOO_BROWSE.
2. Não é possível especificar essa opção ao abrir uma lista de distribuição. Para obter informações adicionais sobre listas de distribuição, consulte "Listas de distribuição" na página 238.
3. Use o MQOO_CO_OP em conjunto com MQOO_BROWSE se estiver usando a navegação cooperativa; consulte Opções

Opções de MQOPEN para remoção de mensagens
Três opções controlam a abertura de uma fila para remover as mensagens dela.

É possível usar somente uma delas em qualquer chamada MQOPEN. Essas opções definem se o seu programa tem acesso exclusivo ou compartilhado para a fila. *Acesso exclusivo* significa que, até você fechar a fila,apenas é possível remover mensagens dela. Se outro programa tenta abrir a fila para remover mensagens, a chamada MQOPEN falha. *Acesso compartilhado* significa que mais de um programa pode removermensagens da fila.

A abordagem mais aconselhável é aceitar o tipo de acesso que foi destinado para a fila quando a fila foi definida. A definição de fila envolveu a configuração do *Shareability* e oAtributos *DefInputOpenOption*. Para aceitar esse acesso, use a opção MQOO_INPUT_AS_Q_DEF. Consulte Tabela 35 na página 225 para ver como a definição desses atributos afeta o tipo de acesso que você receberá quando usar esta opção.

Tabela 35. Como atributos da fila e opções de chamada MQOPEN afetam o acesso a filas

Atributos da Fila		**Tipo de acesso com as opções MQOPEN**		
Shareability	*DefInputOpenOption*	**AS_Q_DEF**	**SHARED**	**EXCLUSIVE**
SHAREABLE	SHARED	compartilhado	compartilhado	exclusivo
SHAREABLE	EXCLUSIVE	exclusivo	compartilhado	exclusivo
NOT_SHAREABLE*	SHARED*	exclusivo	exclusivo	exclusivo
NOT_SHAREABLE	EXCLUSIVE	exclusivo	exclusivo	exclusivo

Nota: * Embora seja possível definir que uma fila tenha essa combinação de atributos, a opção de abertura de entrada padrão é substituída pelo atributo de compartilhamento.

Alternativamente:

- Se você souber que seu aplicativo pode funcionar com êxito, mesmo se outros programas puderem remover mensagens da fila ao mesmo tempo, use a opção MQOO_INPUT_SHARED. O Tabela 35 na página 225 mostra como, em alguns casos, você terá acesso exclusivo à fila, mesmo com essa opção.
- Se você souber que seu aplicativo pode funcionar com êxito somente se outros programas forem impedidos de remover mensagens da fila ao mesmo tempo, use a opção MQOO_INPUT_EXCLUSIVE.

Nota:

1. Não é possível remover mensagens de uma fila remota. Portanto, não é possível abrir uma fila remota usando qualquer uma das opções MQOO_INPUT_*.
2. Não é possível especificar essa opção ao abrir uma lista de distribuição. Veja informações adicionais na publicação “Listas de distribuição” na página 238.

Opções de MQOPEN para configuração e consulta de atributos
Para abrir uma fila de modo que seja possível configurar seus atributos, use a opção MQOO_SET.

Não é possível configurar os atributos de qualquer outro tipo de objeto (consulte “Consultando e configurando atributos de objeto” na página 324).

Para abrir um objeto para que seja possível consultar sobre seus atributos, use a opção MQOO_INQUIRE.

Nota: Não é possível especificar essa opção ao abrir uma lista de distribuição.

Opções de MQOPEN relacionadas ao contexto da mensagem
Se deseja conseguir associar as informações a uma mensagem ao colocá-la em uma fila, deve-se usar uma das opções de contexto da mensagem ao abrir a fila.

As opções permitem diferenciar entre as informações de contexto que se relacionam ao *usuário* que originou a mensagem e que se relacionam ao *aplicativo* que originou a mensagem. Além disso, é possível escolher configurar as informações de contexto ao colocar a mensagem na fila ou ter o contexto obtido automaticamente a partir de outro identificador de filas.

Conceitos relacionados

"Contexto da mensagem" na página 39
As informações de *Contexto da Mensagem* permitem que o aplicativo recupere a mensagem para descobrir sobre o originador da mensagem.

"Controlando informações de contexto" na página 235
Quando você usa a chamada MQPUT ou MQPUT1 para colocar uma mensagem em uma fila, é possível especificar se o gerenciador de filas deve incluir algumas informações de contexto padrão para o descritor de mensagens. Aplicativos que possuem o nível apropriado de autoridade podem incluir informações de contexto adicionais. É possível usar o campo de opções na estrutura MQPMO para controlar informações de contexto.

Opção MQOPEN para a autoridade de usuário alternativo
Quando você tenta abrir um objeto usando a chamada MQOPEN, o gerenciador de filas verifica se você tem autoridade para abrir esse objeto. Se você não estiver autorizado, a chamada falhará.

No entanto, os programas do servidor podem desejar que o gerenciador de filas verifique a autorização do usuário para o qual estão trabalhando em vez da autorização própria do servidor. Para fazer isso, eles devem usar a opção MQOO_ALTERNATE_USER_AUTHORITY da chamada MQOPEN e especificar o ID de usuário alternativo no campo *AlternateUserId* da estrutura MQOD. Geralmente, o servidor poderia obter o ID do usuário das informações de contexto na mensagem que ele está processando.

Opção de MQOPEN para quiesce do gerenciador de filas
No ambiente CICS no z/OS, se você usar a chamada MQOPEN quando o gerenciador de filas estiver em um estado de quiesce, a chamada sempre falhará

Em outros ambientes do z/OS , sistemas IBM i, Windows e em ambientes de sistemas do UNIX and Linux , a chamada falhará quando o gerenciador de filas estiver em quiesce somente se você usar a opção MQOO_FAIL_IF_QUIESCING da chamada MQOPEN

Opção de MQOPEN para resolver nomes de filas locais
Ao abrir uma fila local, de alias ou de modelo, a fila local é retornada.

No entanto, ao abrir uma fila remota ou uma fila de clusters, os campos *ResolvedQName* e *ResolvedQMgrName* da estrutura MQOD são preenchidos com os nomes da fila remota e o gerenciador de filas remotas está localizado na definição de fila remota ou com a fila de clusters remotos escolhida.

Use a opção MQOO_RESOLVE_LOCAL_Q da chamada MQOPEN para preencher *ResolvedQName* na estrutura MQOD com o nome da fila local que foi aberta. *ResolvedQMgrName* é preenchido de forma semelhante com o nome do gerenciador de filas locais que hospeda a fila local. Esse campo está disponível somente com a Versão 3 da estrutura MQOD; se a estrutura for anterior à Versão 3, MQOO_RESOLVE_LOCAL_Q será ignorado sem que um erro seja retornado.

Se você especificar MQOO_RESOLVE_LOCAL_Q ao abrir, por exemplo, uma fila remota, *ResolvedQName* será o nome da fila de transmissão para a qual as mensagens serão colocadas. *ResolvedQMgrName*é o nome do gerenciador de fila local que hospeda a fila de transmissão

Criando filas dinâmicas

Use uma fila dinâmica quando não precisar da fila depois que o seu aplicativo for finalizado.

Por exemplo, você poderia usar uma fila dinâmica para a sua fila de resposta. Especifique o nome da fila de resposta no campo *ReplyToQ* da estrutura MQMD quando colocar uma mensagem em uma fila (consulte "Definindo mensagens usando a estrutura MQMD" na página 230).

Para criar uma fila dinâmica, use um modelo conhecido como uma fila modelo, junto com a chamada MQOPEN. Você cria uma fila modelo usando os comandos WebSphere MQ ou as operações e os painéis de controle A fila dinâmica que você cria assume os atributos da fila modelo.

Quando chamar MQOPEN, especifique o nome da fila modelo no campo *ObjectName* da estrutura MQOD. Quando a chamada é concluída, o campo *ObjectName* é configurado como o nome da fila dinâmica que é criada. Além disso, o campo *ObjectQMgrName* é configurado como o nome do gerenciador de filas local.

É possível especificar o nome da fila dinâmica criada de três maneiras:

- Forneça o nome completo que você deseja no campo *DynamicQName* da estrutura MQOD.
- Especifique um prefixo (menos de 33 caracteres) para o nome e deixe que o gerenciador de filas gere o restante do nome. Isso significa que o gerenciador de filas gera um nome exclusivo, mas você ainda tem algum controle (por exemplo, você talvez queira que cada usuário use um certo prefixo ou queira dar uma classificação de segurança especial para filas com um certo prefixo em seus nomes). Para usar esse método, especifique um asterisco (*) para o último caractere que não estiver em branco do campo *DynamicQName*. Não especifique um único asterisco (*) para o nome da fila dinâmica.
- Deixe que o gerenciador de filas gere o nome completo. Para usar esse método, especifique um asterisco (*) na primeira posição do caractere do campo *DynamicQName*.

Para obter informações adicionais sobre esses métodos, consulte a descrição do campo DynamicQName.

Há mais informações sobre filas dinâmicas em filas Dinâmicas e Modelos .

Abrindo filas remotas

Uma fila remota é uma fila que é de propriedade de um gerenciador de filas diferente daquela à qual o aplicativo está conectado.

Para abrir uma fila remota, use a chamada MQOPEN como para uma fila local. É possível especificar o nome da fila da seguinte forma:

1. No campo *ObjectName* da estrutura MQOD, especifique o nome da fila remota conforme conhecida para o gerenciador de filas *locais*.

   Nota: Deixe o campo *ObjectQMgrName* em branco neste caso.
2. No campo *ObjectName* da estrutura MQOD, especifique o nome da fila remota, conforme conhecido para o gerenciador de filas *remotas*. No campo *ObjectQMgrName*, especifique um dos seguintes:
   - O nome da fila de transmissão que tem o mesmo nome que o gerenciador de filas remotas. O nome e as letras maiúsculas, minúsculas ou uma combinação delas deve corresponder *exatamente*.
   - O nome de um objeto de alias do gerenciador de filas que é resolvido para o gerenciador de filas de destino ou para a fila de transmissão.

   Isso informa ao gerenciador de filas o destino da mensagem, assim como a fila de transmissão na qual precisa ser colocada para chegar lá.
3. Se *DefXmitQname* for suportado, no campo *ObjectName* da estrutura MQOD, especifique o nome da fila remota, conforme é conhecido pelo gerenciador de filas *remotas*.

   Nota: Configure o campo *ObjectQMgrName* para o nome do gerenciador de filas remotas (não pode ser deixado em branco neste caso).

Somente nomes locais são validados quando você chama MQOPEN; a última verificação é para a existência da fila de transmissão a ser usada.

Esses métodos são resumidos em Tabela 34 na página 220

Fechando objetos usando a chamada MQCLOSE

Para fechar um objeto, use a chamada MQCLOSE.

Se o objeto for uma fila, observe o seguinte:

- Não é preciso esvaziar uma fila dinâmica temporária antes de fechá-la.

  Ao fechar uma fila dinâmica temporária, ela será excluída juntamente com quaisquer mensagens que ainda possam estar nela. Isso é verdadeiro mesmo se houver chamadas MQGET, MQPUT ou MQPUT1 não confirmadas pendentes na fila.
- No WebSphere MQ para z/OS, se você tiver quaisquer solicitações MQGET com uma opção MQGMO_SET_SIGNAL pendente para essa fila, elas serão cancelados
- Se você abriu a fila usando a opção MQOO_BROWSE, o cursor de pesquisa é destruído.

O fechamento não está relacionado ao ponto de sincronização, portanto, é possível fechar filas antes ou após o ponto de sincronização.

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

- Uma manipulação de conexões. Use a mesma manipulação de conexões usada para abri-la ou, como alternativa, para aplicativos CICS no z/OS, é possível especificar a constante MQHC_DEF_HCONN (que tem o valor zero)...
- O manipulador do objeto que você deseja fechar. Obtenha este a partir da saída da chamada MQOPEN.
- MQCO_NONE no campo *Options* (a menos que você esteja fechando uma fila dinâmica permanente).
- A opção de controle para determinar se o gerenciador de filas deve excluir a fila mesmo se ainda houver mensagens nela (ao fechar uma fila dinâmica permanente).

A saída de MQCLOSE é:

- Um código de conclusão
- Um código de razão
- A manipulação de objetos, redefinido para o valor MQHO_UNUSABLE_HOBJ

Descrições dos parâmetros da chamada MQCLOSE são fornecidas em MQCLOSE.

Colocando mensagens em uma fila

Use estas informações para aprender como colocar mensagens em uma fila.

Use a chamada MQPUT para colocar mensagens na fila. É possível usar o MQPUT repetidamente para colocar várias mensagens na mesma fila, após a chamada MQOPEN inicial. Chame MQCLOSE quando você tiver concluído todas as suas mensagens na fila.

Se você deseja colocar uma mensagem única em uma fila e fechar a fila imediatamente depois, é possível usar a chamada MQPUT1. MQPUT1 executa as mesmas funções que a seguinte sequência de chamadas:

- MQOPEN
- MQPUT
- MQCLOSE

, no entanto, se você tiver mais de uma mensagem para colocar na fila, é mais eficiente usar a chamada MQPUT. Isso depende do tamanho da mensagem e da plataforma em que se está trabalhando.

Use os seguintes links para descobrir mais sobre colocar mensagens em uma fila:

- “Colocando mensagens em uma fila local usando a chamada MQPUT” na página 229
- “Colocando mensagens em uma fila remota” na página 234
- “Configurando propriedades de uma mensagem” na página 234
- “Controlando informações de contexto” na página 235
- “Colocando uma mensagem em uma fila usando a chamada MQPUT1” na página 237
- “Listas de distribuição” na página 238
- “Alguns casos em que as chamadas put falham” na página 243

Conceitos relacionados

“Visão geral da Message Queue Interface” na página 198
Aprenda sobre os componentes de Message Queue Interface (MQI).

“Conectando-se e desconectando-se de um gerenciador de filas” na página 209
Para usar WebSphere MQ serviços de programação, um programa deve ter uma conexão com um gerenciador de filas. Use essas informações para aprender como conectar-se e desconectar-se de um gerenciador de filas.

“Abrindo e fechando objetos” na página 218

Estas informações fornecem um insight sobre como abrir e fechar objetos do WebSphere MQ .

"Obtendo mensagens de uma fila" na página 243
Use estas informações para aprender como obter mensagens de uma fila.

"Consultando e configurando atributos de objeto" na página 324
Atributos são as propriedades que definem as características de um objeto WebSphere MQ .

"Confirmando e fazendo backup de unidades de trabalho" na página 327
Essas informações descrevem como confirmar e restaurar quaisquer operações de GET e PUT que ocorreram em uma unidade de trabalho.

"Iniciando aplicativos IBM WebSphere MQ usando acionadores" na página 333
Aprenda sobre acionadores e como iniciar aplicativos IBM WebSphere MQ usando acionadores.

"Trabalhando com MQI e clusters" na página 351
Há opções especiais em chamadas e códigos de retorno relacionadas ao armazenamento em cluster.

Colocando mensagens em uma fila local usando a chamada MQPUT

Use estas informações para aprender sobre colocar mensagens em uma fila local usando a chamada MQPUT.

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

- Uma manipulaçao de conexões (Hconn).
- Um identificador de fila (Hobj).
- Uma descrição da mensagem que você deseja colocar na fila. Isso no formato de uma estrutura do descritor de mensagens (MQMD).
- Informações de controle, no formato de uma estrutura de opções put-message (MQPMO).
- O comprimento dos dados contidos dentro da mensagem (MQLONG).
- Os dados da mensagem em si.

A saída da chamada MQPUT é a seguinte:

- Um código de razão (MQLONG)
- Um código de conclusão (MQLONG)

Se a chamada for concluída com sucesso, ela também retornará a estrutura de suas opções e a estrutura de ser descritor de mensagens. A chamada modifica a estrutura de suas opções para mostrar o nome da fila e o gerenciador de filas para o qual a mensagem foi enviada. Se você solicitar que o gerenciador de filas gere um valor exclusivo para o identificador da mensagem para a qual está efetuando put (especificando zero binário no campo *MsgId* da estrutura MQMD), a chamada insere o valor no campo *MsgId* antes de retornar essa estrutura para você. Reconfigure esse valor antes de emitir outro MQPUT.

Existe uma descrição da chamada MQPUT em MQPUT.

Para obter mais descrição sobre as informações necessárias como entrada para a chamada MQPUT, consulte os links a seguir:

- "Especificando identificadores" na página 229
- "Definindo mensagens usando a estrutura MQMD" na página 230
- "Especificando opções usando a estrutura MQPMO" na página 230
- "Os dados em sua mensagem" na página 233
- "Efetuando put de mensagens: usando identificadores de mensagens" na página 234

Especificando identificadores

Para a manipulação de conexões (*Hconn*) no CICS em aplicativos z/OS , é possível especificar a constante MQHC_DEF_HCONN (que tem o valor zero), ou é possível usar a manipulação de conexões retornada

pela chamada MQCONN ou MQCONNX. Para outros aplicativos, use sempre a manipulação de conexões retornada pela chamada MQCONN ou MQCONNX.

Independentemente do ambiente no qual está trabalhando, use o mesmo identificador de fila (*Hobj*) que é retornado pela chamada MQOPEN.

Definindo mensagens usando a estrutura MQMD

A estrutura do descritor de mensagens (MQMD) é um parâmetro de entrada/saída para as chamadas MQPUT e MQPUT1. Use-a para definir a mensagem que está colocando em uma fila.

Se MQPRI_PRIORITY_AS_Q_DEF ou MQPER_PERSISTENCE_AS_Q_DEF for especificado para a mensagem e a fila for uma fila de cluster, os valores usados serão aqueles da fila para a qual a chamada MQPUT é resolvida. Se essa fila estiver desativada para MQPUT, a chamada falhará. Consulte Configurando um cluster de gerenciador de filas para obter mais informações.

Nota: Use MQPMO_NEW_MSG_ID e MQPMO_NEW_CORREL_ID antes de efetuar put de uma nova mensagem para assegurar que *MsgId* e *CorrelId* sejam exclusivos. Os valores nesses campos são retornados em um MQPUT bem-sucedido.

Há uma introdução para as propriedades de mensagens que o MQMD descreve no “Mensagens IBM WebSphere MQ” na página 9 e há uma descrição da própria estrutura no MQMD.

Especificando opções usando a estrutura MQPMO

Use a estrutura de MQPMO (Put Message Option) para passar opções para as chamadas MQPUT e MQPUT1.

As seções a seguir fornecem ajuda sobre como preencher os campos dessa estrutura. Há uma descrição da estrutura em MQPMO.

A estrutura inclui os campos a seguir:

- *StrucId*
- *Version*
- *Options*
- *Context*
- *ResolvedQName*
- *ResolvedQMgrName*
- *RecsPresent*
- *PutMsgRecsFields*
- *ResponseRecOffset and ResponseRecPtr*
- *OriginalMsgHandle*
- *NewMsgHandle*
- *Action*
- *PubLevel*

O conteúdo desses campos é o seguinte:

StrucId
Identifica a estrutura como uma estrutura de opções put-message. Esse é um campo de quatro caracteres. Sempre especifique MQPMO_STRUC_ID.

Versão
Descreve o número da versão da estrutura. O padrão é MQPMO_VERSION_1. Se inserir MQPMO_VERSION_2, será possível usar listas de distribuição (consulte “Listas de distribuição” na página 238). Se você inserir MQPMO_VERSION_3, será possível usar identificadores de mensagens e propriedades de mensagens. Se inserir MQPMO_CURRENT_VERSION, seu aplicativo será configurado para sempre usar o nível mais recente.

Opções
Isso controla o seguinte:

- Se a operação put está incluída em uma unidade de trabalho
- Quanto de informações de contexto está associado a uma mensagem
- De onde as informações de contexto são obtidas
- Se a chamada falhará se o gerenciador de filas estiver em um estado de quiesce
- Se agrupamento ou segmentação é permitido
- Geração de um novo identificador de mensagem e identificador de correlação
- A ordem na qual as mensagens e os segmentos são colocados em uma fila
- Se nomes de filas locais devem ser resolvidos

Se deixar o campo *Options* configurado para o valor padrão (MQPMO_NONE), a mensagem para a qual put foi efetuado tem informações de contexto padrão associadas a ela.

Além disso, a maneira que a chamada opera com pontos de sincronização é determinada pela plataforma. O padrão de controle do ponto de sincronização é yes no z/OS; para outras plataformas, é no.

Contexto
Indica o nome do identificador de fila do qual deseja que as informações de contexto sejam copiadas (se solicitado no campo *Options*).

Para obter uma introdução ao contexto da mensagem, consulte "Contexto da mensagem" na página 39. Para obter informações sobre como usar a estrutura MQPMO para controlar as informações de contexto em uma mensagem, consulte "Controlando informações de contexto" na página 235.

ResolvedQName
Contém o nome (após a resolução de qualquer nome de alias) da fila que foi aberta para receber a mensagem. Esse é um campo de saída.

ResolvedQMgrName
Contém o nome (após a resolução de qualquer nome alternativo) do gerenciador de filas que possui a fila em *ResolvedQName*. Esse é um campo de saída.

O MQPMO também pode acomodar campos necessários para listas de distribuição (consulte "Listas de distribuição" na página 238). Se desejar usar esse recurso, a Versão 2 da estrutura MQPMO será usada. Inclui os campos a seguir:

RecsPresent
Esse campo contém o número de filas na lista de distribuição; ou seja, o número de Put Message Records (MQPMR) e os Response Records (MQRR) correspondentes presentes.

O valor inserido pode ser o mesmo que o número de Object Records fornecidos na chamada MQOPEN. No entanto, se o valor for menor que o número de Object Records fornecido na chamada MQOPEN ou se você não fornecer Put Message Records, os valores das filas não definidos serão obtidos dos valores padrão fornecidos pelo descritor de mensagens. Além disso, se o valor for maior que o número de Object Records fornecido, os Put Message Records em excesso serão ignorados.

É recomendável executar um dos seguintes:

- Se desejar receber um relatório ou resposta de cada destino, insira o mesmo valor que aparece na estrutura MQOR e use MQPMRs que contém campos *MsgId*. Inicialize esses campos *MsgId* para zeros ou especifique MQPMO_NEW_MSG_ID.

  Quando tiver colocado a mensagem na fila, os valores de *MsgId* que o gerenciador de filas criou serão disponibilizados nos MQPMRs; será possível usá-los para identificar qual destino está associado a cada relatório ou resposta.
- Se não deseja receber relatórios ou respostas, escolha uma das opções a seguir:

1. Se desejar identificar destinos que falham imediatamente, pode ser que queira inserir o mesmo valor no campo *RecsPresent* como aparece na estrutura MQOR e fornecer MQRRs para identificar esses destinos. Não especifique nenhum MQPMRs.
2. Se não desejar identificar destinos com falha, insira zero no campo *RecsPresent* e não forneça MQPMRs nem MQRRs.

Nota: Se estiver usando MQPUT1, o número de Response Record Pointers e Response Record Offsets deve ser zero.

Para obter uma descrição completa de Put Message Records (MQPMR) e Response Records (MQRR), consulte MQPMR e MQRR.

PutMsgRecFields
Isso indica quais campos estão presentes em cada Put Message Record (MQPMR). Para obter uma lista desses campos, consulte "Usando a estrutura MQPMR" na página 242.

PutMsgRecOffset* e *PutMsgRecPtr
Ponteiros (geralmente em C) e deslocamentos (geralmente em COBOL) são usados para direcionar os Put Message Records (consulte "Usando a estrutura MQPMR" na página 242 para obter uma visão geral da estrutura MQPMR).

Use o campo *PutMsgRecPtr* para especificar um ponteiro para o primeiro Put Message Record ou o campo *PutMsgRecOffset* para especificar o deslocamento do primeiro Put Message Record. Esse é o deslocamento do início de MQPMO. Dependendo do campo *PutMsgRecFields*, insira um valor não nulo para *PutMsgRecOffset* ou *PutMsgRecPtr*.

ResponseRecOffset e ResponseRecPtr
Você também usa ponteiros e deslocamentos para direcionar Response Records (consulte "Usando a estrutura MQRR" na página 241 para obter informações adicionais sobre Response Records).

Use o campo *ResponseRecPtr* para especificar um ponteiro para o Response Record ou o campo *ResponseRecOffset* para especificar o deslocamento do primeiro Response Record. Esse é o deslocamento do início da estrutura MQPMO. Insira um valor não nulo para *ResponseRecOffset* ou *ResponseRecPtr*.

Nota: Se estiver usando MQPUT1 para colocar mensagens em uma lista de distribuição, *ResponseRecPtr* deve ser nulo ou zero e *ResponseRecOffset* deve ser zero.

A Versão 3 da estrutura MQPMO inclui adicionalmente os campos a seguir:

OriginalMsgHandle
O uso que é possível fazer desse campo depende do valor do campo *Action*. Se estiver efetuando put de uma nova mensagem com propriedades de mensagem associadas, configure esse campo para o identificador de mensagem criado anteriormente e ative as propriedades. Se estiver encaminhando, respondendo ou gerando um relatório em resposta a uma mensagem anteriormente recuperada, esse campo contém o identificador dessa mensagem.

NewMsgHandle
Se especificar um *NewMsgHandle*, quaisquer propriedades associadas ao identificados substituem as propriedades associadas a *OriginalMsgHandle*. Para obter mais informações, consulte Action (MQLONG).

Ação
Use esse campo para especificar o tipo de put que está sendo executado. Valores possíveis e seus significados são os seguintes:

MQACTP_NEW
Esta é uma nova mensagem não relacionada a qualquer outra.

MQACTP_FORWARD
Esta mensagem foi recuperada anteriormente e agora está sendo encaminhada.

MQACTP_REPLY
Esta mensagem é uma resposta a uma mensagem recuperada anteriormente.

MQACTP_REPORT
Esta mensagem é um relatório gerado como resultado de uma mensagem recuperada anteriormente.

Para obter mais informações, consulte Action (MQLONG).

PubLevel
Se esta mensagem for uma publicação, será possível definir esse campo para determinar quais as assinaturas a recebem. Somente assinaturas com um *SubLevel* menor ou igual a esse valor receberão essa publicação. O valor padrão é 9, que é o nível mais alto e significa que assinaturas com qualquer *SubLevel* podem receber esta publicação.

Os dados em sua mensagem

Forneça o endereço do buffer que contém seus dados no parâmetro *Buffer* da chamada MQPUT. É possível incluir qualquer coisa nos dados em suas mensagens. A quantia de dados nas mensagens, no entanto, afeta o desempenho do aplicativo que as está processando.

O tamanho máximo dos dados é determinado por:

- O atributo *MaxMsgLength* do gerenciador de filas
- O atributo *MaxMsgLength* da fila na qual você está colocando a mensagem
- O tamanho de qualquer cabeçalho da mensagem incluído pelo WebSphere MQ (incluindo o cabeçalho de devoluções MQDLH e o cabeçalho da lista de distribuição, MQDH)

O atributo *MaxMsgLength* do gerenciador de filas retém o tamanho da mensagem que o gerenciador de filas pode processar. Isso tem um padrão de 100 MB para todos os produtos WebSphere MQ em V6 ou superior

Para determinar o valor desse atributo, use a chamada MQINQ no objeto do gerenciador de filas. Para mensagens grandes, é possível mudar esse valor.

O atributo *MaxMsgLength* de uma fila determina o tamanho máximo da mensagem que é possível colocar na fila. Se tentar colocar uma mensagem com um tamanho maior do que o valor desse atributo, a chamada MQPUT falhará. Se estiver colocando uma mensagem em uma fila remota, o tamanho máximo de mensagem que é possível colocar com sucesso é determinado pelo atributo *MaxMsgLength* da fila remota, de quaisquer filas de transmissão intermediárias nas quais a mensagem é colocada ao longo da rota para seu destino e dos canais usados.

Para uma operação MQPUT, o tamanho da mensagem deve ser menor que ou igual ao atributo *MaxMsgLength* da fila e do gerenciador de filas. Os valores desses atributos são independentes, mas é recomendado configurar o *MaxMsgLength* da fila para um valor menor ou igual ao do gerenciador de filas.

WebSphere MQ inclui informações do cabeçalho em mensagens nas seguintes circunstâncias:

- Quando você coloca uma mensagem em uma fila remota, o WebSphere MQ inclui uma estrutura de cabeçalho de transmissão (MQXQH) na mensagem. Essa estrutura inclui o nome da fila de destino e seu gerenciador de filas proprietário.
- Se o WebSphere MQ não puder entregar uma mensagem para uma fila remota, ele tentará colocar a mensagem na fila de mensagens não entregues (não entregues). Ela inclui uma estrutura MQDLH na mensagem. Essa estrutura inclui o nome da fila de destino e a razão pela qual a mensagem foi colocada na fila de mensagens não entregues.
- Se desejar enviar uma mensagem para várias filas de destino, o WebSphere MQ incluirá um cabeçalho MQDH na mensagem Isso descreve os dados que estão presentes em uma mensagem, pertencente a uma lista de distribuição, em uma fila de transmissão. Considere isso ao escolher um valor ideal para o comprimento máximo da mensagem.
- Se a mensagem for um segmento ou uma mensagem em um grupo, WebSphere MQ poderá incluir um MQMDE.

Essas estruturas são descritas em MQDH e MQMDE.

Se suas mensagens forem do tamanho máximo permitido para essas filas, a adição desses cabeçalhos significa que as operações put falham porque agora as mensagens são muito grandes. Para reduzir a possibilidade das operações put com falha:

- Torne o tamanho de suas mensagens menor do que o atributo *MaxMsgLength* das filas de transmissão e de mensagens não entregues. Permita pelo menos o valor da constante MQ_MSG_HEADER_LENGTH (mais para listas de distribuição grandes).
- Certifique-se de que o atributo *MaxMsgLength* da fila de mensagens não entregues esteja configurado para o mesmo que o *MaxMsgLength* do gerenciador de filas que possui a fila de mensagens não entregues.

Os atributos do gerenciador de filas e as constantes de enfileiramento de mensagens estão descritos em Atributos do gerenciador de filas.

Efetuando put de mensagens: usando identificadores de mensagens

Dois identificadores de mensagens estão disponíveis na estrutura MQPMO, *OriginalMsgHandle* e *NewMsgHandle*. O relacionamento entre esses identificadores de mensagens é definido pelo valor do campo *Action* de MQPMO.

Para obter detalhes integrais, consulte Action (MQLONG). Um identificador de mensagem não é necessariamente requerido para efetuar put de uma mensagem. Seu propósito é associar propriedades a uma mensagem, portanto, é necessário somente se você estiver usando as propriedades de mensagem.

Colocando mensagens em uma fila remota

Quando você deseja colocar uma mensagem em uma fila remota (ou seja, uma fila pertencente a um gerenciador de filas diferente do que aquele ao qual seu aplicativo está conectado) em vez de uma fila local, a única consideração adicional é como especificar o nome da fila quando você a abre.Isso é descrito no “Abrindo filas remotas” na página 227. Não há mudança em como você usa a chamada MQPUT ou MQPUT1 para uma fila local.

Para obter mais informações sobre o uso de filas remotas e de transmissão, consulte WebSphere MQ técnicas de mensagens distribuídas .

Configurando propriedades de uma mensagem

Chamada MQSETMP para cada propriedade que você deseja configurar. Ao efetuar put da mensagem, configure o identificador de mensagem e os campos de ação da estrutura MQPMO.

Para associar propriedades a uma mensagem, a mensagem deve ter um identificador de mensagem. Crie um identificador de mensagem usando a chamada de função MQCRTMH. Chame MQSETMP especificando esse identificador de mensagem para cada propriedade que deseja configurar. Um programa de amostra, amqsstma.c, é fornecido para ilustrar o uso de MQSETMP.

Se essa for uma nova mensagem, ao colocá-la em uma fila, usando MQPUT ou MQPUT1, configure o campo OriginalMsgHandle em MQPMO para o valor desse identificador de mensagem e configure o campo Action de MQPMO para MQACTP_NEW (esse é o valor padrão).

Se esta for uma mensagem anteriormente recuperada e agora você está encaminhado ou respondendo a mesma ou enviando um relatório em resposta a ela, coloque o identificador de mensagem original no campo OriginalMsgHandle de MQPMO e o novo identificador de mensagem no campo NewMsgHandle. Configure o campo Action para MQACTP_FORWARD, MQACTP_REPLY ou MQACTP_REPORT, conforme apropriado.

Se tiver propriedades em um cabeçalho MQRFH2 de uma mensagem anteriormente recuperada, será possível convertê-las para propriedades do identificador de mensagem usando a chamada MQBUFMH.

Se estiver colocando sua mensagem em uma fila em um gerenciador de filas em um nível anterior ao WebSphere MQ Versão 7.0, que não pode processar propriedades de mensagem, é possível configurar o parâmetro PropertyControl na definição de canal para especificar como as propriedades devem ser tratadas.

Controlando informações de contexto

Quando você usa a chamada MQPUT ou MQPUT1 para colocar uma mensagem em uma fila, é possível especificar se o gerenciador de filas deve incluir algumas informações de contexto padrão para o descritor de mensagens. Aplicativos que possuem o nível apropriado de autoridade podem incluir informações de contexto adicionais. É possível usar o campo de opções na estrutura MQPMO para controlar informações de contexto.

Para controlar informações de contexto, use o campo *Options* na estrutura MQPMO.

Se não, o gerenciador de filas grava informações de contexto que podem já estar no descritor de mensagens com as informações de identidade e contexto que foi gerado para a mensagem. Isso é o mesmo que especificar a opção MQPMO_DEFAULT_CONTEXT.Talvez você queira essas informações de contexto padrão ao criar uma nova mensagem (por exemplo, ao processar a entrada do usuário a partir de uma tela de consulta).

Se você não deseja informações de contexto associadas a sua mensagem, use a opção MQPMO_NO_CONTEXT. Ao colocar uma mensagem sem contexto, todas as verificações de autoridade feitas pelo IBM WebSphere MQ são feitas usando um ID de usuário em branco. Um ID de usuário em branco não pode ter a autoridade explícita designada aos recursos do IBM WebSphere MQ, mas é tratado como um membro do grupo especial 'nobody'. Para obter mais detalhes sobre o grupo especial nobody, consulte Informações de referência de interface de serviços instaláveis.

Se você não deseja informações de contexto associadas a sua mensagem, use a opção MQPMO_NO_CONTEXT.

As seções a seguir deste tópico explicam o uso de contexto de identidade, o contexto do usuário e todo o contexto.

- “Transmitindo contexto de identidade” na página 235
- “Transmitindo contexto de usuário” na página 236
- “Transmitindo todo o contexto” na página 236
- “Configurando contexto de identidade” na página 236
- “Configurando contexto do usuário” na página 236
- “Configurando todo o contexto” na página 236

Transmitindo contexto de identidade

Em geral, os programas devem transmitir informações de contexto de identidade de mensagem para mensagem em torno de um aplicativo até que os dados atinjam seu destino final.

Os programas devem mudar as informações de contexto de origem cada vez que eles mudarem os dados. No entanto, os aplicativos que desejam mudar ou configurar qualquer informação de contexto devem ter o nível apropriado de autoridade. O gerenciador de filas verifica essa autoridade quando os aplicativos abrem as filas; eles devem ter autoridade para usar as opções de contexto apropriadas para a chamada MQOPEN.

Se seu aplicativo obtiver uma mensagem, processar os dados da mensagem e, em seguida, colocar os dados mudados em outra mensagem (possivelmente para processamento por outro aplicativo), o aplicativo deverá transmitir as informações do contexto de identidade da mensagem original para a nova mensagem. É possível permitir que o gerenciador de filas crie as informações de contexto de origem.

Para salvar as informações de contexto da mensagem original, use a opção MQOO_SAVE_ALL_CONTEXT ao abrir a fila para obter a mensagem. Isso está em adição a quaisquer outras opções que você usar com a chamada MQOPEN. Observe, no entanto, que você não pode salvar informações de contexto se você só procurar a mensagem.

Quando você cria a segunda mensagem:

- Abra a fila usando a opção MQOO_PASS_IDENTITY_CONTEXT (além da opção MQOO_OUTPUT).

- No campo *Context* da estrutura de opções de mensagem put, forneça o identificador da fila a partir da qual você salvou as informações de contexto.
- No campo *Options* da estrutura de opções de mensagem put, especifique a opção MQPMO_PASS_IDENTITY_CONTEXT.

Transmitindo contexto de usuário

Não é possível optar por transmitir apenas contexto do usuário. Para transmitir o contexto do usuário ao colocar uma mensagem, especifique MQPMO_PASS_ALL_CONTEXT. Quaisquer propriedades no contexto do usuário são transmitidas da mesma maneira que o contexto de origem.

Quando um MQPUT ou MQPUT1 ocorre e o contexto está sendo transmitido, todas as propriedades no contexto do usuário são transmitidas a partir da mensagem recuperada para a mensagem colocada. Quaisquer propriedades de contexto do usuário que o aplicativo put tenha alterado são colocadas com seus valores originais. Quaisquer propriedades de contexto do usuário que o aplicativo put excluiu são restauradas na mensagem colocada. Quaisquer propriedades de contexto do usuário que o aplicativo put incluiu na mensagem são retidas.

Transmitindo todo o contexto

Se o seu aplicativo obtiver uma mensagem e colocar os dados da mensagem (inalterada) em outra mensagem, o aplicativo deverá transmitir todas (identidade, origem e usuário) as informações de contexto da mensagem original para a nova mensagem. Um exemplo de um aplicativo que pode fazer isso é um transportador de mensagem que move mensagens de uma fila para outra.

Siga o mesmo procedimento que para transmitir contexto de identidade, exceto que você usa a opção MQOPEN MQOO_PASS_ALL_CONTEXT e a opção de mensagem colocada MQPMO_PASS_ALL_CONTEXT.

Configurando contexto de identidade

Se desejar configurar as informações de contexto de identidade para uma mensagem:

- Abra a fila usando a opção MQOO_SET_IDENTITY_CONTEXT.
- Coloque a mensagem na fila, especificando a opção MQPMO_SET_IDENTITY_CONTEXT. No descritor de mensagens, especifique quaisquer informações de contexto de identidade que você requerer.

Nota: Quando você configura alguns (mas não todos) dos campos de contexto de identidade usando as opções MQOO_SET_IDENTITY_CONTEXT e MQPMO_SET_IDENTITY_CONTEXT, é importante perceber que o gerenciador de filas não configura nenhum um dos outros campos.

Para modificar qualquer uma das opções de contexto da mensagem, deve-se ter as autorizações apropriadas para emitir a chamada. Por exemplo, para usar MQOO_SET_IDENTITY_CONTEXT ou MQPMO_SET_IDENTITY_CONTEXT, deve-se ter a permissão `+setid`.

Configurando contexto do usuário

Para configurar uma propriedade no contexto do usuário, configure o campo Contexto do descritor de propriedade de mensagem (MQPD) para MQPD_USER_CONTEXT quando você fizer a chamada MQSETMP.

Você não precisa de nenhuma autoridade especial para configurar uma propriedade no contexto do usuário. O contexto do usuário não tem as opções de contexto MQOO_SET_* ou MQPMO_SET_*.

Configurando todo o contexto

Se você desejar configurar ambas as informações de contexto de origem e de identidade para uma mensagem:

1. Abra a fila usando a opção MQOO_SET_ALL_CONTEXT.

2. Coloque a mensagem na fila, especificando a opção MQPMO_SET_ALL_CONTEXT. No descritor de mensagens, especifique quaisquer informações de contexto de origem e de identidade de que você precisa.

A autoridade apropriada é necessária para cada tipo de configuração de contexto.

Conceitos relacionados

"Contexto da mensagem" na página 39
As informações de *Contexto da Mensagem* permitem que o aplicativo recupere a mensagem para descobrir sobre o originador da mensagem.

Referências relacionadas

"Opções de MQOPEN relacionadas ao contexto da mensagem" na página 225
Se deseja conseguir associar as informações a uma mensagem ao colocá-la em uma fila, deve-se usar uma das opções de contexto da mensagem ao abrir a fila.

Colocando uma mensagem em uma fila usando a chamada MQPUT1

Use a chamada MQPUT1 quando desejar fechar a fila imediatamente após ter colocado uma única mensagem nela. Por exemplo, um aplicativo do servidor provavelmente usará a chamada MQPUT1 quando estiver enviando uma resposta para cada uma das filas diferentes.

MQPUT1 é funcionalmente equivalente a chamar MQOPEN seguida de MQPUT, seguida de MQCLOSE. A única diferença na sintaxe para as chamadas MQPUT e MQPUT1 é que, para MQPUT, você especifica uma manipulação de objetos, enquanto que, para MQPUT1, especifica uma estrutura do descritor de objeto (MQOD) conforme definido em MQOPEN (consulte "Identificando objetos (a estrutura MQOD)" na página 220). Isso acontece porque você precisa fornecer informações para a chamada MQPUT1 sobre a fila que ela precisa abrir, enquanto que a ao chamar MQPUT, a fila já deve estar aberta.

Como entrada para a chamada MQPUT1, deve-se fornecer:

- Uma manipulação de conexões.
- Uma descrição do objeto que deseja abrir. Isso no formato de uma estrutura do descritor de objeto (MQOD).
- Uma descrição da mensagem que você deseja colocar na fila. Isso no formato de uma estrutura do descritor de mensagens (MQMD).
- Informações de controle no formato de uma estrutura de opções put-message (MQPMO).
- O comprimento dos dados contidos dentro da mensagem (MQLONG).
- O endereço dos dados da mensagem.

A saída de MQPUT1 é:

- Um código de conclusão
- Um código de razão

Se a chamada for concluída com sucesso, ela também retornará a estrutura de suas opções e a estrutura de ser descritor de mensagens. A chamada modifica a estrutura de suas opções para mostrar o nome da fila e o gerenciador de filas para o qual a mensagem foi enviada. Se você solicitar que o gerenciador de filas gere um valor exclusivo para o identificador da mensagem que está colocando (especificando zero binário no campo *MsgId* da estrutura MQMD), a chamada inserirá o valor no campo *MsgId* antes de retornar essa estrutura para você.

Nota: Não é possível usar MQPUT1 com um nome de fila modelo; no entanto, quando uma fila modelo tiver sido aberta, será possível emitir um MQPUT1 para uma fila dinâmica.

Os seis parâmetros de entrada para MQPUT1 são:

Hconn
Essa é uma manipulação de conexões. Para aplicativos CICS , é possível especificar a constante MQHC_DEF_HCONN (que tem o valor zero) ou usar o identificador de conexão retornado pela chamada MQCONN ou MQCONNX. Para outros programas, sempre use a manipulação de conexões retornada pela chamada MQCONN ou MQCONNX.

ObjDesc
Esta é uma estrutura do descritor de objeto (MQOD).

Nos campos *ObjectName* e *ObjectQMgrName*, dê o nome da fila no qual você deseja colocar uma mensagem e o nome do gerenciador de filas que possui essa fila.

O campo *DynamicQName* é ignorado para a chamada MQPUT1 porque ela não pode usar filas modelo.

Use o campo *AlternateUserId* se desejar denominar um identificador de usuário alternativo que deve ser usado para testar a autoridade para abrir a fila.

MsgDesc
Essa é uma estrutura de descritor de mensagens (MQMD). Como com a chamada MQPUT, use esta estrutura para definir a mensagem que está colocando na fila.

PutMsgOpts
Esta é uma estrutura de opções put-message (MQPMO). Use-a como o faria para a chamada MQPUT (consulte "Especificando opções usando a estrutura MQPMO" na página 230).

Quando o campo *Options* estiver configurado para zero, o gerenciador de filas usa o seu próprio ID do usuário ao executar testes de autoridade para acessar a fila. Além disso, o gerenciador de filas ignora qualquer identificador de usuário alternativo fornecido no campo *AlternateUserId* da estrutura MQOD.

BufferLength
Este é o comprimento da mensagem.

Buffer
Essa é a área de buffer que contém o texto de sua mensagem.

Ao usar clusters, MQPUT1 opera como se MQOO_BIND_NOT_FIXED estivesse em vigor. Os aplicativos devem usar os campos resolvidos na estrutura MQPMO em vez da estrutura MQOD para determinar para onde a mensagem foi enviada. Consulte Configurando um cluster de gerenciador de filas para obter mais informações.

Há uma descrição da chamada MQPUT1 em MQPUT1.

Listas de distribuição

Não suportado no WebSphere MQ para z/OS. As listas de distribuição permitem que você coloque uma mensagem para diversos destinos em uma única chamada MQPUT ou MQPUT1.Uma chamada única MQOPEN pode abrir diversas filas e uma única chamada MQPUT pode, então, colocar uma mensagem em cada uma dessas filas. Algumas informações genéricas de estruturas MQI usadas para este processo podem ser substituídas pelas informações específicas relativas aos destinos individuais incluídos na lista de distribuição.

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.

Quando uma chamada MQOPEN é emitida, informações genéricas são obtidas do Descritor de objeto (MQOD). Se você especificar MQOD_VERSION_2 no campo *Version* e um valor maior que zero no campo *RecsPresent*, o *Hobj* poderá ser definido como uma manipulação de uma lista (de uma ou mais filas) em vez daquela de uma fila. Neste caso, informações específicas são fornecidas por meio dos registros de objeto (MQORs), que fornecem detalhes de destino (ou seja, *ObjectName* e *ObjectQMgrName*).

A manipulação de objeto (*Hobj*) é transmitida para a chamada MQPUT, permitindo que você coloque em uma lista em vez de em uma única fila.

Quando uma mensagem é colocada nas filas (MQPUT), informações genéricas são obtidas a partir da estrutura de Put Message Option (MQPMO) e o Message Descriptor (MQMD). Informações específicas são concedidas sob a forma de Put Message Records (MQPMRs).

Os Response Records (MQRR) podem receber um código de conclusão e código de razão específicos para cada fila de destino.

Figura 29 na página 239 mostra como as listas de distribuição funcionam.

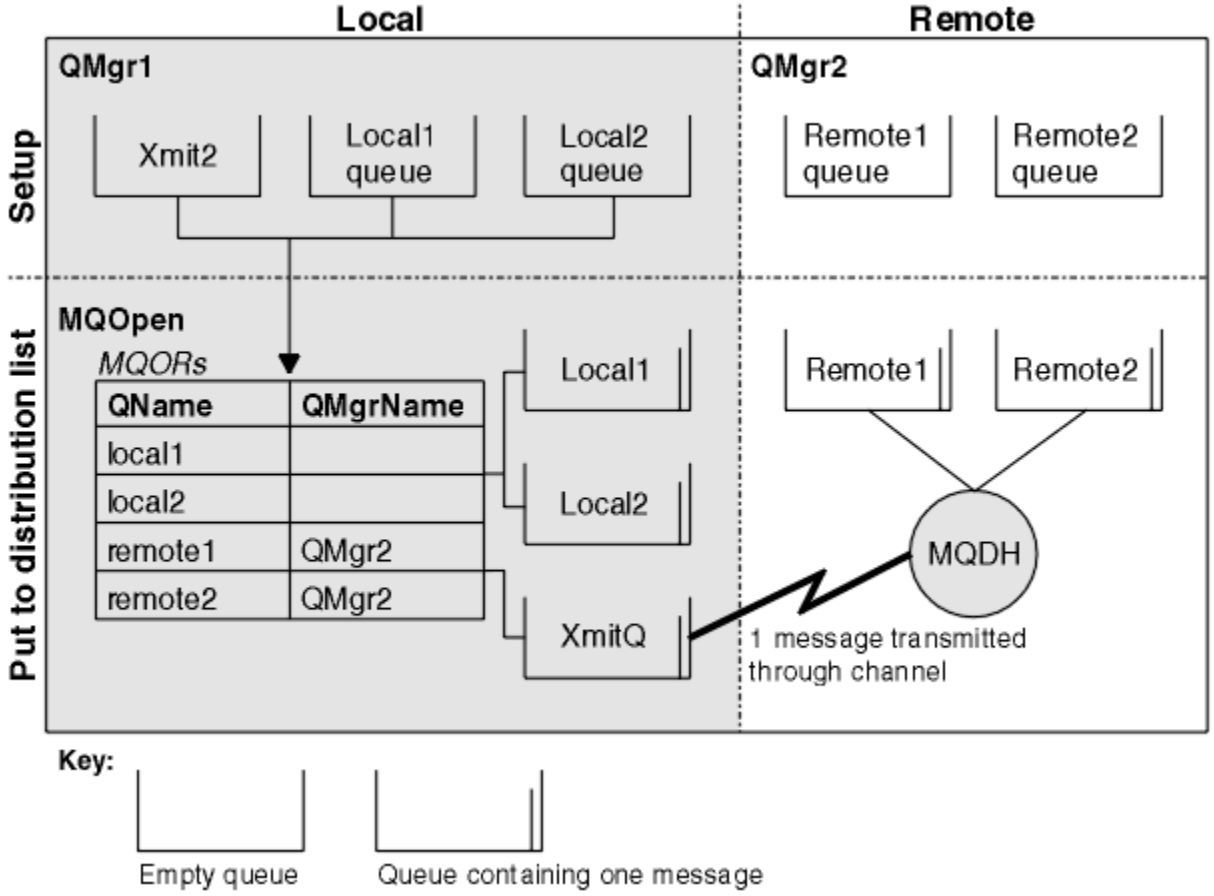


Figura 29. Como as listas de distribuição funcionam

Abrindo listas de distribuição

Use a chamada MQOPEN para abrir uma lista de distribuição e use as opções da chamada para especificar o que você deseja fazer com a lista.

Como entrada para MQOPEN, deve-se fornecer:

- Uma manipulação de conexões (consulte "Colocando mensagens em uma fila" na página 228 para obter uma descrição)
- Informações genéricas na estrutura Objeto Descriptor (MQOD)
- O nome de cada fila que você deseja abrir, usando a estrutura Object Record (MQOR)

A saída de MQOPEN é:

- Uma manipulação de objetos que representa seu acesso à lista de distribuição
- Um código de conclusão genérico
- Um código de razão genérico
- Response Records (opcionais), contendo um código de conclusão e de razão para cada destino

Usando a estrutura MQOD

Use a estrutura MQOD para identificar as filas que deseja abrir.

Para definir uma lista de distribuição, deve-se especificar MQOD_VERSION_2 no campo *Version*, um valor maior que zero no campo *RecsPresent* e MQOT_Q no campo *ObjectType* campo. Consulte MQOD para obter uma descrição de todos os campos da estrutura MQOD.

Usando a estrutura MQOR

Forneça uma estrutura MQOR para cada destino.

A estrutura contém a fila de destino e nomes do gerenciador de filas. Os campos *ObjectName* e *ObjectQMgrName* no MQOD não são usados para listas de distribuição. Deve haver um ou mais registros de objeto. Se *ObjectQMgrName* for deixado em branco, o gerenciador de filas locais será usado. Consulte ObjectName e ObjectQMgrName para obter informações adicionais sobre esses campos.

É possível especificar as filas de destino de duas maneiras:

- Usando o campo de deslocamento *ObjectRecOffset*.

  Nesse caso, o aplicativo deve declarar sua própria estrutura contendo uma estrutura MQOD, seguida pela matriz de registros MQOR (com quantos elementos de matriz forem necessários) e configurar *ObjectRecOffset* para o deslocamento do primeiro elemento na matriz a partir do início do MQOD. Assegure que esse deslocamento esteja correto.

  O uso de recursos integrados fornecidos pela linguagem de programação é recomendado, se estiverem disponíveis em todos os ambientes nos quais o aplicativo é executado. O código a seguir ilustra essa técnica para a linguagem de programação COBOL:

```
01  MY-OPEN-DATA.
    02 MY-MQOD.
       COPY CMQODV.
    02 MY-MQOR-TABLE OCCURS 100 TIMES.
       COPY CMQORV.
    MOVE LENGTH OF MY-MQOD TO MQOD-OBJECTRECOFFSET.
```

  Como alternativa, use a constante MQOD_CURRENT_LENGTH, se a linguagem de programação não suportar os recursos internos necessários em todos os ambientes em questão. O código a seguir ilustra essa técnica:

```
01  MY-MQ-CONSTANTS.
    COPY CMQV.
01  MY-OPEN-DATA.
    02 MY-MQOD.
       COPY CMQODV.
    02 MY-MQOR-TABLE OCCURS 100 TIMES.
       COPY CMQORV.
    MOVE MQOD-CURRENT-LENGTH TO MQOD-OBJECTRECOFFSET.
```

  No entanto, isso funcionará corretamente somente se a estrutura MQOD e a matriz de registros MQOR forem contíguas; se o compilador inserir bytes para ignorar entre a matriz MQOD e MQOR, eles devem ser incluídos no valor armazenado em *ObjectRecOffset*.

  Usar *ObjectRecOffset* é recomendado para linguagens de programação que não suportam o tipo de dados do ponteiro ou que implementam o tipo de dados do ponteiro sem portabilidade para diferentes ambientes (por exemplo, a linguagem de programação COBOL).

- Usando o campo do ponteiro *ObjectRecPtr*

  Nesse caso, o aplicativo pode declarar a matriz de estruturas MQOR separadamente da estrutura MQOD e configurar *ObjectRecPtr* para o endereço da matriz. O código a seguir ilustra essa técnica para a linguagem de programação C:

```
MQOD MyMqod;
MQOR MyMqor[100];
MyMqod.ObjectRecPtr = MyMqor;
```

Usar *ObjectRecPtr* é recomendado para linguagens de programação que suportam o tipo de dados do ponteiro de uma maneira que tenha portabilidade para os diferentes ambientes (por exemplo, a linguagem de programação C).

Qualquer que seja a técnica escolhida, deve-se usar um dos *ObjectRecOffset* e *ObjectRecPtr*; a chamada falhará com o código de razão MQRC_OBJECT_RECORDS_ERROR se ambos forem zero ou ambos forem diferentes de zero

Usando a estrutura MQRR

Essas estruturas são específicas do destino; cada Response Record contém um campo *CompCode* e *Reason* para cada fila de uma lista de distribuição. Deve-se usar essa estrutura para permitir que você distingua onde residem os problemas.

Por exemplo, se você receber um código de razão MQRC_MULTIPLE_REASONS e sua lista de distribuição contiver cinco filas de destino, não saberá a quais filas os problemas se aplicam se não usar essa estrutura. No entanto, se tiver um código de conclusão e um código de razão para cada destino, será possível localizar os erros mais facilmente.

Consulte MQRR para obter informações adicionais sobre a estrutura MQRR.

Figura 30 na página 241 mostra como é possível abrir uma lista de distribuição em C.

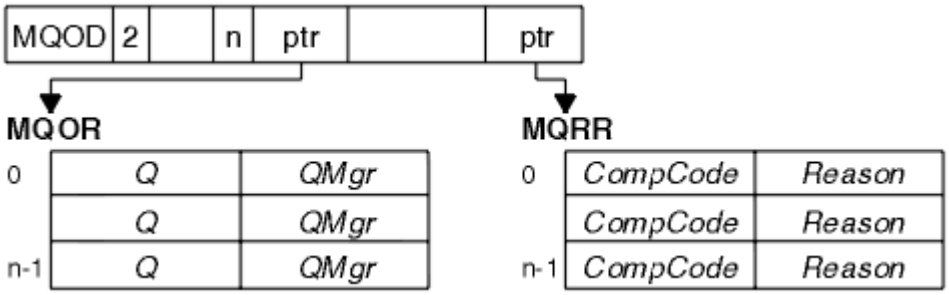


Figura 30. Abrindo uma lista de distribuição em C

Figura 31 na página 241 mostra como é possível abrir uma lista de distribuição em COBOL.

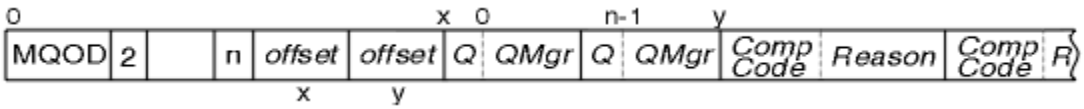


Figura 31. Abrindo uma lista de distribuição em COBOL

Usando as opções de MQOPEN

É possível especificar as opções a seguir ao abrir uma lista de distribuição:

- MQOO_OUTPUT
- MQOO_FAIL_IF_QUIESCING (opcional)
- MQOO_ALTERNATE_USER_AUTHORITY (opcional)
- MQOO_*_CONTEXT (opcional)

Consulte "Abrindo e fechando objetos" na página 218 para obter uma descrição dessas opções.

Colocando mensagens em uma lista de distribuição

Para colocar mensagens em uma lista de distribuição, é possível usar MQPUT ou MQPUT1.

Como entrada, deve-se fornecer:

- Uma manipulação de conexões (consulte "Colocando mensagens em uma fila" na página 228 para obter uma descrição).

- Uma manipulação de objetos. Se uma lista de distribuição for aberta usando MQOPEN, *Hobj* permite somente colocar na lista.
- Uma estrutura do descritor de mensagens (MQMD). Consulte MQMD para obter uma descrição dessa estrutura.
- Informações de controle na forma de uma estrutura da opção put-message (MQPMO). Consulte "Especificando opções usando a estrutura MQPMO" na página 230 para obter informações sobre como preencher os campos da estrutura MQPMO.
- informações de controle na forma de Put Message Records (MQPMR).
- O comprimento dos dados contidos dentro da mensagem (MQLONG).
- Os dados da mensagem em si.

A saída é:

- Um código de conclusão
- Um código de razão
- Response Records (opcional)

Usando a estrutura MQPMR

Esta estrutura é opcional e fornece informações específicas do destino para alguns campos que você pode desejar identificar de forma diferente daqueles já identificados no MQMD.

Para obter uma descrição desses campos, consulte MQPMR.

O conteúdo de cada registro depende das informações fornecidas no campo *PutMsgRecFields* do MQPMO. Por exemplo, no programa de amostra AMQSPTL0.C (consulte "O programa de amostra Distribution List" na página 129 para obter uma descrição) que mostra o uso de listas de distribuição, a amostra optar por fornecer valores para *MsgId* e *CorrelId* no MQPMR. Essa seção do programa de amostra é semelhante à seguinte:

```
typedef struct
{
MQBYTE24 MsgId;
MQBYTE24 CorrelId;
} PutMsgRec;
...
/**********************
MQLONG PutMsgRecFields=MQPMRF_MSG_ID | MQPMRF_CORREL_ID;
```

Isso sugere que *MsgId* e *CorrelId* são fornecidos para cada destino de uma lista de distribuição. Put Message Records são fornecidos como uma matriz.

Figura 32 na página 242 mostra como é possível colocar uma mensagem em uma lista de distribuição em C.

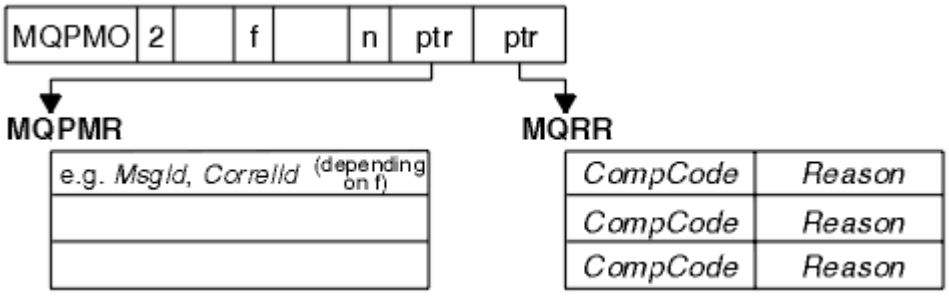


Figura 32. Colocando uma mensagem em uma lista de distribuição em C

Figura 33 na página 243 mostra como é possível colocar uma mensagem em uma lista de distribuição em COBOL.

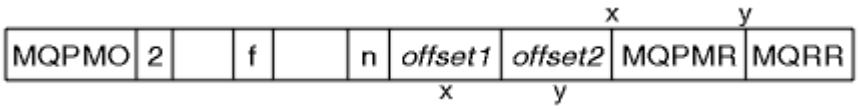


Figura 33. Colocando uma mensagem em uma lista de distribuição em COBOL

Usando MQPUT1

Se você estiver usando MQPUT1, considere os pontos a seguir:

1. Os valores dos campos *ResponseRecOffset* e *ResponseRecPtr* devem ser nulos ou zero.
2. O Response Records, se necessário, deve ser direcionado a partir do MQOD.

Alguns casos em que as chamadas put falham

Se determinados atributos de uma fila forem alterados usando a opção FORCE em um comando durante o intervalo entre a emissão de um MQOPEN e uma chamada MQGET, a chamada MQGET falhará e retornará o código de razão MQRC_OBJECT_CHANGED.

O gerenciador de filas marca a manipulação de objetos como não sendo mais válida. Isso também acontece se as mudanças forem feitas enquanto uma chamada MQPUT1 estiver sendo processada ou se as mudanças se aplicarem a qualquer fila para a qual o nome da fila é resolvido. Os atributos que afetam a manipulação dessa forma são listados na descrição da chamada MQOPEN no MQOPEN. Se sua chamada retornar o código de razão MQRC_OBJECT_CHANGED, feche a fila, reabra-a e, em seguida, tente colocar uma mensagem novamente.

Se as operações put forem inibidas para uma fila na qual você está tentando colocar mensagens (ou qualquer fila para a qual o nome da fila é resolvido), a chamada MQPUT ou MQPUT1 falhará e retornará o código de razão MQRC_PUT_INHIBITED. É possível colocar uma mensagem com êxito se você tentar a chamada posteriormente, se o design do aplicativo for tal que outros programas mudem os atributos de filas regularmente.

Além disso, se a fila na qual você está tentando colocar a mensagem estiver completa, a chamada MQPUT ou MQPUT1 falhará e retornará MQRC_Q_FULL.

Se uma fila dinâmica (temporária ou permanente) foi excluída, chamadas MQPUT que usam uma manipulação de objetos adquirida anteriormente falharão e retornarão o código de razão MQRC_Q_DELETED. Nesta situação, é uma boa prática fechar a manipulação de objetos, pois não é mais útil para você.

No caso de listas de distribuição, diversos códigos de conclusão e códigos de razão podem ocorrer em uma única solicitação. Eles não podem ser manipulados usando somente os campos de saída *CompCode* e *Reason* em MQOPEN e MQPUT.

Quando você usa listas de distribuição para colocar mensagens em múltiplos destinos, os Registros de resposta contêm o *CompCode* e o *Reason* específicos para cada destino. Se você receber um código de conclusão de MQCC_FAILED, nenhuma mensagem será colocada em nenhuma fila de destino com êxito. Se o código de conclusão for MQCC_WARNING, a mensagem será colocada com êxito em uma ou mais das filas de destino. Se você receber um código de retorno de MQRC_MULTIPLE_REASONS, os códigos de razão não serão todos iguais para cada destino. Portanto, é recomendado usar a estrutura MQRR para que seja possível determinar qual fila ou quais filas causaram um erro e as razões para cada.

Obtendo mensagens de uma fila

Use estas informações para aprender como obter mensagens de uma fila.

É possível obter mensagens de uma fila de duas maneiras:

1. É possível remover uma mensagem da fila para que outros programas não possam mais vê-la.
2. É possível copiar uma mensagem, deixando a mensagem original na fila. Isso é conhecido como *procura*. É possível remover a mensagem quando a tiver procurado.

Em ambos os casos, você usa a chamada MQGET, mas primeiro seu aplicativo deve ser conectado ao gerenciador de filas, e deve-se usar a chamada MQOPEN para abrir a fila (para entrada, procura ou ambos). Essas operações estão descritas em "Conectando-se e desconectando-se de um gerenciador de filas" na página 209 e "Abrindo e fechando objetos" na página 218.

Quando tiver aberto a fila, será possível usar a chamada MQGET repetidamente para procurar ou remover mensagens na mesma fila. Chame MQCLOSE quando tiver terminado de obter todas as mensagens desejadas da fila.

Use os links a seguir para descobrir mais sobre como obter mensagens a partir de uma fila:

- "Obtendo mensagens de uma fila usando a chamada MQGET" na página 244
- "A ordem em que as mensagens são recuperadas de uma fila" na página 248
- "Obtendo uma mensagem específica" na página 260
- "Melhorando o desempenho de mensagens não persistentes" na página 261
- "Manipulando mensagens com mais de 4 MB de comprimento" na página 266
- "Esperando mensagens" na página 271
-
- "Ignorando restauração" na página 272
- "Conversão de Dados do Aplicativo" na página 274
- "Procurando mensagens em uma fila" na página 275
- "Alguns casos em que a chamada MQGET falha" na página 281

Conceitos relacionados

"Visão geral da Message Queue Interface" na página 198
Aprenda sobre os componentes de Message Queue Interface (MQI).

"Conectando-se e desconectando-se de um gerenciador de filas" na página 209
Para usar WebSphere MQ serviços de programação, um programa deve ter uma conexão com um gerenciador de filas. Use essas informações para aprender como conectar-se e desconectar-se de um gerenciador de filas.

"Abrindo e fechando objetos" na página 218
Estas informações fornecem um insight sobre como abrir e fechar objetos do WebSphere MQ .

"Colocando mensagens em uma fila" na página 228
Use estas informações para aprender como colocar mensagens em uma fila.

"Consultando e configurando atributos de objeto" na página 324
Atributos são as propriedades que definem as características de um objeto WebSphere MQ .

"Confirmando e fazendo backup de unidades de trabalho" na página 327
Essas informações descrevem como confirmar e restaurar quaisquer operações de GET e PUT que ocorreram em uma unidade de trabalho.

"Iniciando aplicativos IBM WebSphere MQ usando acionadores" na página 333
Aprenda sobre acionadores e como iniciar aplicativos IBM WebSphere MQ usando acionadores.

"Trabalhando com MQI e clusters" na página 351
Há opções especiais em chamadas e códigos de retorno relacionadas ao armazenamento em cluster.

Obtendo mensagens de uma fila usando a chamada MQGET

A chamada MQGET recebe uma mensagem de uma fila local aberta. Ela não pode obter uma mensagem de uma fila em outro sistema.

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

- Uma manipulação de conexões.
- Um identificador de fila.

- Uma descrição da mensagem que você deseja obter da fila. Ela está na forma de uma estrutura do descritor de mensagens (MQMD).
- Informações de controle na forma de uma estrutura Get Message Options (MQGMO).
- O tamanho do buffer que você designou para manter a mensagem (MQLONG).
- O endereço do armazenamento no qual colocar a mensagem.

A saída de MQGET é:

- Um código de razão
- Um código de conclusão
- A mensagem na área de buffers que você especificou, se a chamada for concluída com êxito
- Sua estrutura de opções, modificada para mostrar o nome da fila da qual a mensagem foi recuperada
- Sua estrutura do descritor de mensagem, com o conteúdo dos campos modificado para descrever a mensagem que foi recuperada
- O comprimento da mensagem (MQLONG)

Existe uma descrição da chamada MQGET em MQGET.

As seções a seguir descrevem as informações que devem ser fornecidas como entrada para a chamada MQGET.

- "Especificando manipulações de conexões" na página 245
- "Descrevendo mensagens usando a estrutura MQMD e a chamada MQGET" na página 245
- "Especificando opções MQGET usando a estrutura MQGMO" na página 246
- "Especificando o tamanho da área de buffer" na página 248

Especificando manipulações de conexões

Para CICS em z/OS aplicativos, é possível especificar a constante MQHC_DEF_HCONN (que tem o valor zero), ou usar o identificador de conexão retornado pela chamada MQCONN ou MQCONNX. Para outros aplicativos, use sempre a manipulação de conexões retornada pela chamada MQCONN ou MQCONNX.

Use o identificador de filas (*Hobj*) que é retornado quando você chama MQOPEN.

Descrevendo mensagens usando a estrutura MQMD e a chamada MQGET

Para identificar a mensagem que você deseja obter de uma fila, use a estrutura do descritor de mensagens (MQMD).

Este é um parâmetro de entrada/saída para a chamada MQGET. Há uma introdução para as propriedades de mensagens que o MQMD descreve no "Mensagens IBM WebSphere MQ" na página 9 e há uma descrição da própria estrutura no MQMD.

Se você souber qual mensagem deseja obter da fila, consulte "Obtendo uma mensagem específica" na página 260.

Se você não especificar uma mensagem determinada, MQGET recuperará a *primeira* mensagem na fila. O "A ordem em que as mensagens são recuperadas de uma fila" na página 248 descreve como a prioridade de uma mensagem, o atributo *MsgDeliverySequence* da fila e a opção MQGMO_LOGICAL_ORDER determinam a ordem das mensagens na fila.

Nota: Se desejar usar MQGET mais de uma vez (por exemplo, para percorrer as mensagens na fila), deve-se configurar os campos *MsgId* e *CorrelId* dessa estrutura como nulo após cada chamada. Isso limpa estes campos dos identificadores da mensagem que foi recuperada.

No entanto, se você desejar agrupar suas mensagens, o *GroupId* deverá ser o mesmo para mensagens no mesmo grupo, para que a chamada procure uma mensagem que possui os mesmos identificadores que a mensagem anterior para compor todo o grupo.

Especificando opções MQGET usando a estrutura MQGMO

A estrutura MQGMO é uma variável de entrada/saída para transmitir opções para a chamada MQGET. As seções a seguir ajudam a concluir alguns dos campos desta estrutura.

Há uma descrição da estrutura MQGMO em MQGMO.

StrucId
StrucId é um campo de 4 caracteres usado para identificar a estrutura como uma estrutura de opções get-message. Sempre especifique MQGMO_STRUC_ID.

Version
*Version*descreve o número da versão da estrutura. MQGMO_VERSION_1 é o padrão. Se desejar usar os campos Versão 2 ou recuperar mensagens em ordem lógica, especifique MQGMO_VERSION_2. Se desejar usar os campos Versão 3 ou recuperar mensagens em ordem lógica, especifique MQGMO_VERSION_3. MQGMO_CURRENT_VERSION configura seu aplicativo para usar o nível mais recente.

Options
No seu código, é possível selecionar as opções em qualquer ordem; cada opção é representada por um bit no campo *Options*.

O campo *Options* controla:

- Se a chamada MQGET aguarda que uma mensagem chegue na fila antes de ela ser concluída (consulte "Esperando mensagens" na página 271)
- Se a operação get está incluída em uma unidade de trabalho.
- Se uma mensagem não persistente é recuperada fora do ponto de sincronização, permitindo um sistema de mensagens rápido
- No WebSphere MQ para z/OS, se a mensagem recuperada está marcada como ignorando a restauração (consulte "Ignorando restauração" na página 272)
- Se a mensagem foi removida da fila ou simplesmente procurada
- Se uma mensagem deve ser selecionada usando um cursor de navegação ou outros critérios de seleção
- Se a chamada é bem-sucedida mesmo que a mensagem seja maior que o seu buffer
- No WebSphere MQ for z/OS, se a chamada deve ser concluída. Esta opção também configura um sinal para indicar que você deseja ser notificado quando uma mensagem chegar
- Se a chamada falhará se o gerenciador de filas estiver em um estado de quiesce
- No WebSphere MQ para z/OS, se a chamada falhará se a conexão estiver em um estado quiesce
- Se a conversão de dados de mensagens do aplicativo é necessária (consulte "Conversão de Dados do Aplicativo" na página 274)
- A ordem na qual as mensagens e (exceto para WebSphere MQ para z/OS) segmentos são recuperados de uma fila
- Exceto no WebSphere MQ para z/OS, se somente mensagens lógicas concluídas forem recuperáveis
- Se as mensagens em um grupo podem ser recuperadas apenas quando *todas* as mensagens no grupo estiverem disponíveis
- Exceto no WebSphere MQ para z/OS, se segmentos em uma mensagem lógica podem ser recuperados somente quando *todos* os segmentos na mensagem lógica estiverem disponíveis

Se você deixar o campo *Options* configurado como o valor padrão (MQGMO_NO_WAIT), a chamada MQGET operará dessa maneira:

- Se não houver nenhuma mensagem correspondente aos seus critérios de seleção na fila, a chamada não aguardará que uma mensagem chegue, mas será concluída imediatamente. Além disso, no WebSphere MQ para z/OS, a chamada não configura um sinal solicitando notificação quando tal mensagem chega.
- A maneira como a chamada opera com pontos de sincronização é determinada pela plataforma:

Plataforma	Sob controle do ponto de sincronização
IBM i	NÃO
Sistemas UNIX and Linux	NÃO
z/OS	Sim
Sistemas Windows	NÃO

- No WebSphere MQ para z/OS, a mensagem recuperada não é marcada como ignorando a restauração
- A mensagem selecionada é removida da fila (não procurada).
- Nenhuma conversão de dados de mensagens do aplicativo é necessária.
- A chamada falhará se a mensagem for mais longa que seu buffer.

WaitInterval
O campo *WaitInterval* especifica o tempo máximo (em milissegundos) que a chamada MQGET aguarda uma mensagem chegar na fila quando você usa a opção MQGMO_WAIT. Se nenhuma mensagem chegar dentro do tempo especificado em *WaitInterval*, a chamada será concluída e retornará um código de razão mostrando que não havia nenhuma mensagem que correspondesse aos seus critérios de seleção na fila.

No WebSphere MQ para z/OS, se você usar a opção MQGMO_SET_SIGNAL, o campo *WaitInterval* especificará o tempo para o qual o sinal será configurado

Para obter informações adicionais sobre essas opções, consulte “Esperando mensagens” na página 271 .

Signal1
Signal1 é suportado no WebSphere MQ para z/OS e MQSeries para HP Integrity NonStop Server apenas.

Se você usar a opção MQGMO_SET_SIGNAL para solicitar que seu aplicativo seja notificado quando uma mensagem adequada chegar, especifique o tipo de sinal no campo *Signal1* . No WebSphere MQ em todas as outras plataformas, o campo *Signal1* é reservado e seu valor não é significativo.

Signal2
O campo *Signal2* é reservado em todas as plataformas e seu valor não é significativo.

ResolvedQName
ResolvedQName é um campo de saída no qual o gerenciador de filas retorna o nome da fila (após resolução de qualquer alias) da qual a mensagem foi recuperada.

MatchOptions
MatchOptions controla os critérios de seleção para MQGET.

GroupStatus
GroupStatus indica se a mensagem que você recuperou está em um grupo.

SegmentStatus
SegmentStatus indica se o item que você recuperou é um segmento de uma mensagem lógica.

Segmentation
Segmentation indica se a segmentação é permitida para a mensagem recuperada.

MsgToken

MsgToken identifica exclusivamente uma mensagem.

ReturnedLength
ReturnedLength é um campo de saída no qual o gerenciador de filas retorna o comprimento de dados da mensagem retornados (em bytes).

MsgHandle
O identificador para uma mensagem que deve ser preenchida com as propriedades da mensagem que está sendo recuperada da fila. O identificador foi criado anteriormente por uma chamada MQCRTMH. Todas as propriedades já associadas ao identificador são limpas antes de recuperar uma mensagem.

Especificando o tamanho da área de buffer

No parâmetro *BufferLength* da chamada MQGET, especifique o tamanho da área de buffer para conter os dados de mensagens que você recuperar. Você decide o tamanho que isso deve ter de três maneiras:

1. Você já pode saber qual comprimento de mensagens esperar desse programa. Em caso afirmativo, especifique um buffer deste tamanho.

   No entanto, é possível usar a opção MQGMO_ACCEPT_TRUNCATED_MSG na estrutura MQGMO se desejar que a chamada MQGET seja concluída mesmo que a mensagem seja muito grande para o buffer. Nesse caso:

   - O buffer é preenchido com o máximo da mensagem que ele pode manter
   - A chamada retorna um código de conclusão de aviso
   - A mensagem é removida da fila (descartando o restante da mensagem) ou o cursor de navegação é avançado (se você estiver navegando na fila)
   - O comprimento real da mensagem é retornado em *DataLength*

   Sem esta opção, a chamada ainda é concluída com um aviso, mas não remove a mensagem da fila (ou avança o cursor de navegação).
2. Faça uma estimativa de um tamanho para o buffer (ou mesmo especifique um tamanho de zero bytes) e *não* use a opção MQGMO_ACCEPT_TRUNCATED_MSG. Se a chamada MQGET falhar (por exemplo, porque o buffer é muito pequeno), o comprimento da mensagem será retornado no parâmetro *DataLength* da chamada. (O buffer ainda é preenchido com o máximo da mensagem que ele pode manter, mas o processamento da chamada não é concluído.) Armazene o *MsgId* desta mensagem e, em seguida, repita a chamada MQGET, especificando uma área de buffer do tamanho correto e o *MsgId* que você anotou da primeira chamada.

   Se seu programa estiver atendendo uma fila que também está sendo atendida por outros programas, um desses outros programas poderá remover a mensagem desejada antes que seu programa possa emitir outra chamada MQGET. Seu programa poderia perder tempo procurando por uma mensagem que não existe mais. Para evitar isso, primeiro procure na fila até localizar a mensagem desejada, especificando um *BufferLength* igual a zero e usando a opção MQGMO_ACCEPT_TRUNCATED_MSG. Isso posiciona o cursor de navegação sob a mensagem que você deseja. É possível, então, recuperar a mensagem chamando MQGET novamente, especificando a opção MQGMO_MSG_UNDER_CURSOR. Se outro programa remover a mensagem entre suas chamadas de navegação e remoção, seu segundo MQGET falhará imediatamente (sem procurar na fila inteira), porque não há nenhuma mensagem sob o cursor de navegação.
3. O atributo *MaxMsgLength queue* determina o comprimento máximo de mensagens aceitas por essa fila; o atributo *MaxMsgLength queue manager* determina o comprimento máximo de mensagens aceitas para esse gerenciador de filas. Se você não souber qual comprimento de mensagem esperar, poderá pesquisar sobre o atributo *MaxMsgLength* (usando a chamada MQINQ) e, em seguida, especificar um buffer desse tamanho.

   Tente tornar o tamanho do buffer o mais próximo possível do tamanho da mensagem real para evitar o desempenho reduzido.

   Para obter informações adicionais sobre o atributo *MaxMsgLength*, consulte "Aumentar o comprimento máximo da mensagem" na página 266.

A ordem em que as mensagens são recuperadas de uma fila

É possível controlar a ordem na qual se recuperam as mensagens de uma fila. Esta seção analisa as opções.

Priority

Um programa pode designar uma prioridade a uma mensagem quando ela coloca a mensagem em uma fila (consulte “Prioridades de mensagens” na página 18). As mensagens de prioridade igual são armazenadas em uma fila na ordem de chegada, não a ordem na qual elas são confirmadas.

O gerenciador de filas mantém as filas em sequência estrita FIFO (primeiro a entrar, primeiro a sair) ou em FIFO em sequência de prioridade. Isso depende da configuração do atributo *MsgDeliverySequence* da fila.Quando uma mensagem chega em uma fila, ela é inserida imediatamente após a última mensagem com a mesma prioridade.

Programas podem obter a primeira mensagem de uma fila ou eles podem obter uma mensagem específica de uma fila, ignorando a prioridade dessas mensagens. Por exemplo, um programa pode desejar processar a resposta a uma determinada mensagem que enviou anteriormente. Para obter mais informações, consulte “Obtendo uma mensagem específica” na página 260.

Se um aplicativo coloca uma sequência de mensagens em uma fila, outro aplicativo pode recuperar essas mensagens na mesma ordem em que foram colocadas, desde que:

- Todas as mensagens tenham a mesma prioridade
- As mensagens foram todas colocadas na mesma unidade de trabalho ou todas colocadas fora de uma unidade de trabalho
- A fila é local para o aplicativo de colocação

Se essas condições não forem atendidas e os aplicativos dependerem das mensagens serem recuperadas em uma determinada ordem, os aplicativos devem incluir informações de sequenciamento ou nos dados da mensagem ou estabelecer um meio de reconhecer o recebimento de uma mensagem antes que a próxima seja enviada.

Ordenação lógica e física

As mensagens em filas podem ocorrer (dentro de cada nível de prioridade) em ordem *física* ou *lógica*.

Ordem física é a ordem na qual as mensagens chegam a uma fila. Ordem lógica é quando todas as mensagens e segmentos em um grupo estão em sua sequência lógica, próximos uns dos outros, na posição determinada pela posição física do primeiro item pertencente ao grupo.

Para obter uma descrição dos grupos, mensagens e segmentos, consulte “Grupos de mensagens” na página 36. Essas ordens físicas e lógicas podem diferir, pois:

- Os grupos podem chegar a um destino em momentos semelhantes a partir de diferentes aplicativos e, portanto, perder uma ordem física distinta.
- Mesmo dentro de um único grupo, as mensagens podem ficar fora de ordem por causa de roteamento ou atraso de algumas das mensagens no grupo.

Por exemplo, a ordem lógica poderá ser parecida com a da Figura Figura 34 na página 250:

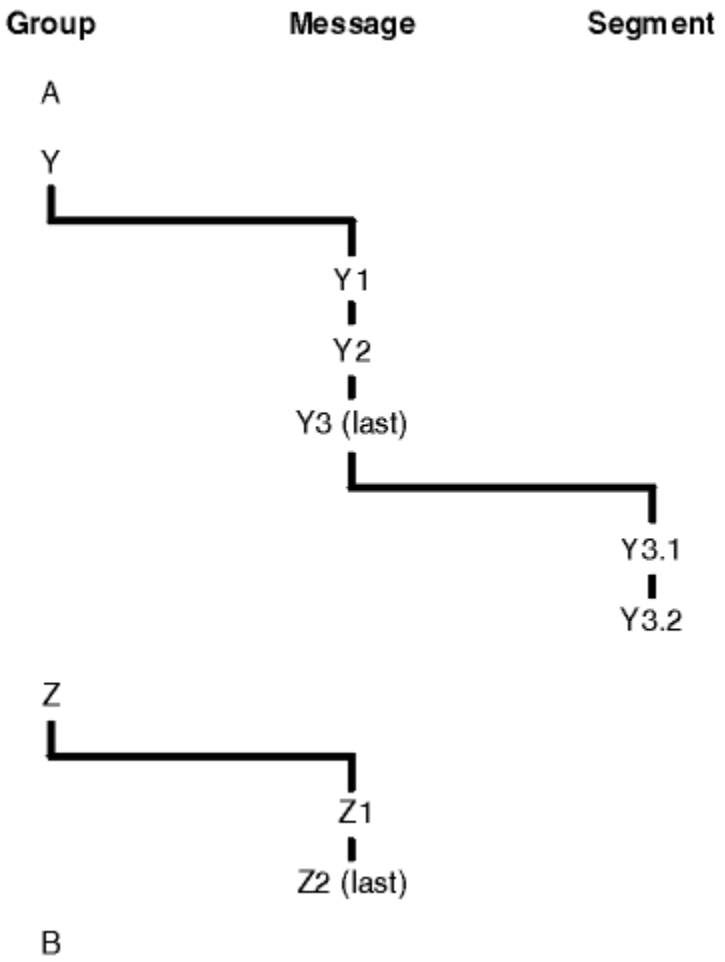


Figura 34. Ordem lógica em uma fila

Essas mensagens ocorreriam na ordem lógica a seguir em uma fila:

1. Mensagem A (não em um grupo)
2. Mensagem lógica 1 do grupo Y
3. Mensagem lógica 2 do grupo Y
4. Segmento 1 da (última) mensagem lógica 3 do grupo Y
5. (Último) segmento 2 da (última) mensagem lógica 3 do grupo Y
6. Mensagem lógica 1 do grupo Z
7. (Última) mensagem lógica 2 do grupo Z
8. Mensagem B (não em um grupo)

A ordem física, no entanto, pode ser totalmente diferente. A posição física do *primeiro* item dentro de cada grupo determina a posição lógica do grupo inteiro. Por exemplo, se os grupos Y e Z chegarem em momentos semelhantes e a mensagem 2 do grupo Z alcançar a mensagem 1 do mesmo grupo, a ordem física seria semelhante à da Figura Figura 35 na página 251:

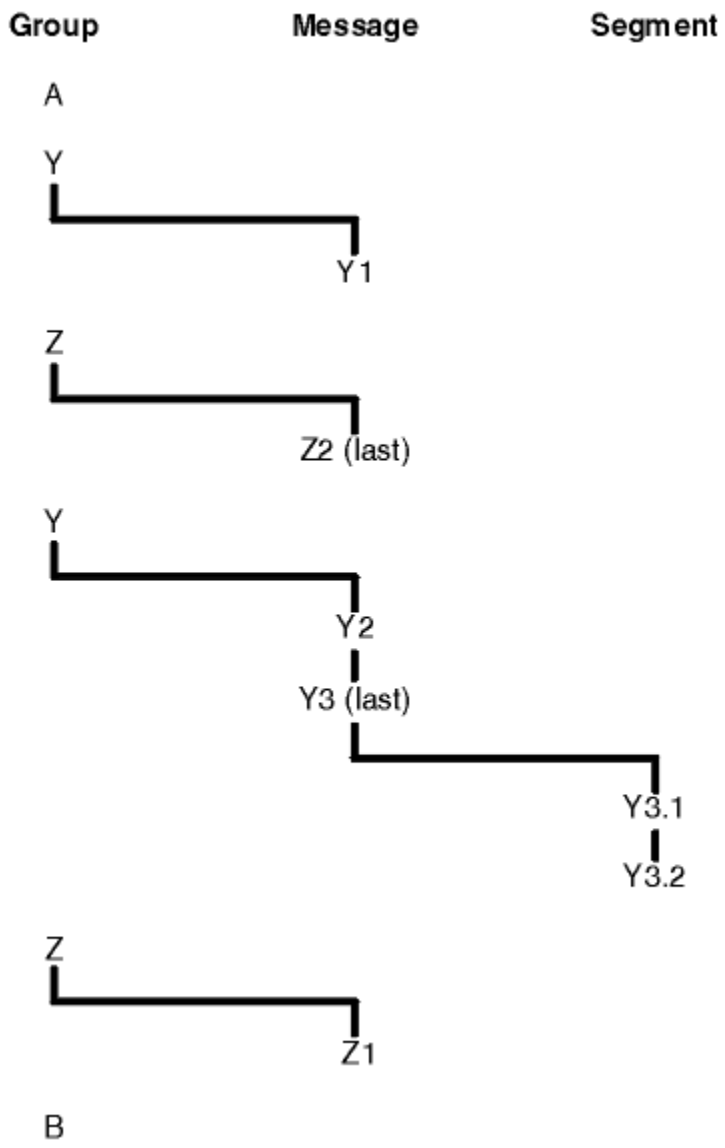


Figura 35. Ordem física em uma fila

Essas mensagens ocorrem na seguinte ordem física na fila:

1. Mensagem A (não em um grupo)
2. Mensagem lógica 1 do grupo Y
3. Mensagem lógica 2 do grupo Z
4. Mensagem lógica 2 do grupo Y
5. Segmento 1 da (última) mensagem lógica 3 do grupo Y
6. (Último) segmento 2 da (última) mensagem lógica 3 do grupo Y
7. Mensagem lógica 1 do grupo Z
8. Mensagem B (não em um grupo)

Nota: No IBM WebSphere MQ for z/OS, a ordem física de mensagens na fila não será garantida se a fila for indexada por GROUPID.

Ao obter mensagens, é possível especificar MQGMO_LOGICAL_ORDER para recuperar mensagens em ordem lógica em vez da ordem física.

Se você emitir uma chamada MQGET com MQGMO_BROWSE_FIRST e MQGMO_LOGICAL_ORDER, as chamadas MQGET com MQGMO_BROWSE_NEXT subsequentes também deverão especificar

MQGMO_LOGICAL_ORDER. De modo inverso, se MQGET com MQGMO_BROWSE_FIRST não especifica MQGMO_LOGICAL_ORDER, os seguintes MQGETs com MQGMO_BROWSE_NEXT também não precisarão.

As informações de grupo e segmento que o gerenciador de filas retém para chamadas MQGET que pesquisam mensagens na fila são separadas do grupo e informações de segmento que o gerenciador de filas retém para chamadas MQGET que removem da fila. Quando você especifica MQGMO_BROWSE_FIRST, o gerenciador de filas ignora as informações de grupo e segmento que devem ser pesquisadas e varre a fila se não houver nenhum grupo atual e nenhuma mensagem lógica atual.

Nota: Não use uma chamada MQGET para pesquisar *além do fim* de um grupo de mensagens (ou mensagem lógica não em um grupo) sem especificar MQGMO_LOGICAL_ORDER. Por exemplo, se a última mensagem no grupo *precede* a primeira mensagem no grupo na fila, usando MQGMO_BROWSE_NEXT para pesquisar além do final do grupo, especificando MQMO_MATCH_MSG_SEQ_NUMBER com o *MsgSeqNumber* configurado como 1 (para localizar a primeira mensagem do próximo grupo) retorna novamente a primeira mensagem no grupo já pesquisado. Isso pode acontecer imediatamente ou um número de chamadas MQGET posterior (se houver grupos de intervenção).

Evite a possibilidade de um loop infinito abrindo a fila *duas vezes* para pesquisar:

- Use o primeiro identificador para pesquisar apenas a primeira mensagem em cada grupo.
- Use o segundo identificador para pesquisar apenas as mensagens em um grupo específico.
- Use as opções MQMO_* para mover o segundo cursor de navegação para a posição do primeiro cursor de navegação, antes de pesquisar as mensagens no grupo.
- Não use a pesquisa MQGMO_BROWSE_NEXT além do fim de um grupo.

Para obter informações adicionais sobre isso, consulte MQGET, MQMD e Regras para validar opções de MQI.

Para a maioria dos aplicativos, você provavelmente escolherá ordem lógica ou física ao pesquisar. No entanto, se você desejar alternar entre esses modos, lembre-se de que quando você emite pela primeira vez uma pesquisa com MQGMO_LOGICAL_ORDER, sua posição dentro da sequência lógica é estabelecida.

Se o primeiro item dentro do grupo não estiver presente neste momento, o grupo que você está não é considerado parte da sequência lógica.

Depois que o cursor de navegação estiver em um grupo, ele poderá continuar dentro do mesmo grupo, mesmo se a primeira mensagem for removida. Inicialmente, no entanto, não é possível fazer movimentos em um grupo usando MQGMO_LOGICAL_ORDER, no qual o primeiro item não está presente.

MQPMO_LOGICAL_ORDER

A opção MQPMO instrui o gerenciador de filas sobre como o aplicativo coloca mensagens em grupos e segmentos de mensagens lógicas. Ela só pode ser especificada na chamada MQPUT. Ela não é válida na chamada MQPUT1.

Se MQPMO_LOGICAL_ORDER é especificado, ele indica que o aplicativo utiliza chamadas MQPUT sucessivas para:

1. Colocar os segmentos em cada mensagem lógica na ordem crescente de deslocamento de segmento, iniciando a partir de 0, sem lacunas.
2. Colocar todos os segmentos em uma mensagem lógica antes de colocar os segmentos na próxima mensagem lógica.
3. Colocar as mensagens lógicas em cada grupo de mensagens na ordem crescente de número de sequência da mensagem, iniciando a partir de 1, sem lacunas.IBM WebSphere MQ incrementa o número de sequência da mensagem automaticamente
4. Colocar todas as mensagens lógicas em um grupo de mensagens antes de colocar mensagens lógicas no próximo grupo de mensagens.

Como o aplicativo informou o gerenciador de filas como ele coloca mensagens em grupos e segmentos de mensagens lógicas, o aplicativo não precisa manter e atualizar as informações do grupo e do segmento sobre cada chamada MQPUT, pois o gerenciador de filas mantém e atualiza essas informações. Especificamente, isso significa que o aplicativo não precisa configurar os campos

GroupId, *MsgSeqNumber* e *Offset* no MQMD, porque o gerenciador de filas configura esses campos para os valores apropriados. O aplicativo deve configurar apenas o campo *MsgFlags* no MQMD, para indicar quando as mensagens pertencem a grupos ou são segmentos de mensagens lógicas e para indicar a última mensagem em um grupo ou último segmento de uma mensagem lógica.

Depois de um grupo de mensagens ou mensagem lógica ser iniciado, chamadas MQPUT subsequentes devem especificar os sinalizadores MQMF_* apropriados em *MsgFlags* no MQMD. Se o aplicativo tentar colocar uma mensagem que não está em um grupo quando há um grupo de mensagens não terminado ou colocar uma mensagem que não é um segmento quando houver uma mensagem lógica não terminada, a chamada falhará com o código de razão MQRC_INCOMPLETE_GROUP ou MQRC_INCOMPLETE_MSG, conforme apropriado. No entanto, o gerenciador de filas retém as informações sobre o grupo de mensagens atual ou de mensagens lógicas atual e o aplicativo pode finalizá-las enviando uma mensagem (possivelmente sem dados da mensagem do aplicativo) especificando MQMF_LAST_MSG_IN_GROUP ou MQMF_LAST_SEGMENT conforme apropriado, antes de emitir novamente a chamada MQPUT para colocar a mensagem que não está no grupo ou que não é um segmento.

Figura 35 na página 251 mostra as combinações de opções e sinalizações que são válidas e os valores dos campos *GroupId*, *MsgSeqNumber*e *Offset* que o gerenciador de filas usa em cada caso. Combinações de opções e sinalizadores que não são mostrados na tabela não são válidos. As colunas na tabela possuem os seguintes significados; Qualquer um significa Sim ou Não:

LOG ORD
Se a opção MQPMO_LOGICAL_ORDER é especificada na chamada.

MIG
Se a opção MQMF_MSG_IN_GROUP ou MQMF_LAST_MSG_IN_GROUP é especificada na chamada.

SEG
Se a opção MQMF_SEGMENT ou MQMF_LAST_SEGMENT é especificada na chamada.

SEG OK
Se a opção MQMF_SEGMENTATION_ALLOWED é especificada na chamada.

Cur grp
Se um grupo de mensagens atual existe antes da chamada.

Cur log msg
Se uma mensagem lógica atual existe antes da chamada.

Outras colunas
Mostram os valores que o gerenciador de filas usa. Anterior indica o valor usado para o campo na mensagem anterior para o manipulador de filas.

Tabela 36. Opções MQPUT relacionadas às mensagens em grupos e segmentos de mensagens lógicas

Opções que você especifica				**Grupo e status log-msg antes da chamada**		**Valores que o gerenciador de filas usa**		
LOG ORD	**MIG**	**SEG**	**SEG OK**	**Cur grp**	**Cur log msg**	***GroupId***	***MsgSeqNumber***	***Offset***
Sim	NÃO	NÃO	NÃO	NÃO	NÃO	MQGI_NONE	1	0
Sim	NÃO	NÃO	Sim	NÃO	NÃO	Nova ID de grupo	1	0
Sim	NÃO	Sim	Qualquer um	NÃO	NÃO	Nova ID de grupo	1	0

Tabela 36. Opções MQPUT relacionadas às mensagens em grupos e segmentos de mensagens lógicas (continuação)

Opções que você especifica				**Grupo e status log-msg antes da chamada**		**Valores que o gerenciador de filas usa**		
LOG ORD	**MIG**	**SEG**	**SEG OK**	**Cur grp**	**Cur log msg**	***GroupId***	***MsgSeqNumber***	***Offset***
Sim	NÃO	Sim	Qualquer um	NÃO	Sim	ID de grupo anterior	1	Deslocamento anterior + comprimento do segmento anterior
Sim	Sim	Qualquer um	Qualquer um	NÃO	NÃO	Nova ID de grupo	1	0
Sim	Sim	Qualquer um	Qualquer um	Sim	NÃO	ID de grupo anterior	Número da sequência anterior + 1	0
Sim	Sim	Sim	Qualquer um	Sim	Sim	ID de grupo anterior	Número de sequência anterior	Deslocamento anterior + comprimento do segmento anterior
NÃO	NÃO	NÃO	NÃO	Qualquer um	Qualquer um	MQGI_NONE	1	0
NÃO	NÃO	NÃO	Sim	Qualquer um	Qualquer um	Nova ID de grupo se MQGI_NONE, valor else no campo	1	0
NÃO	NÃO	Sim	Qualquer um	Qualquer um	Qualquer um	Nova ID de grupo se MQGI_NONE, valor else no campo	1	Valor no campo
NÃO	Sim	NÃO	Qualquer um	Qualquer um	Qualquer um	Nova ID de grupo se MQGI_NONE, valor else no campo	Valor no campo	0
NÃO	Sim	Sim	Qualquer um	Qualquer um	Qualquer um	Nova ID de grupo se MQGI_NONE, valor else no campo	Valor no campo	Valor no campo

Nota:

- MQPMO_LOGICAL_ORDER não é válido na chamada MQPUT1.
- Para o campo *MsgId*, o gerenciador de filas gerará um novo identificador de mensagem, se MQPMO_NEW_MSG_ID ou MQMI_NONE for especificado e usará o valor no campo, caso contrário.
- Para o campo *CorrelId*, o gerenciador de filas gera um novo identificador de correlação, se MQPMO_NEW_CORREL_ID for especificado e usa o valor no campo, caso contrário.

Quando você especificar MQPMO_LOGICAL_ORDER, o gerenciador de filas exigirá que todas as mensagens em um grupo e os segmentos de uma mensagem lógica sejam colocados com o mesmo valor no campo *Persistence* no MQMD, ou seja, todos devem ser persistentes ou todos devem ser não persistentes. Se essa condição não for satisfeita, a chamada MQPUT falhará com o código de razão MQRC_INCONSISTENT_PERSISTENCE.

A opção MQPMO_LOGICAL_ORDER afeta as unidades de trabalho conforme segue:

- Se a primeira mensagem física em um grupo ou mensagem lógica for colocada em uma unidade de trabalho, todas as outras mensagens físicas no grupo ou mensagens lógicas devem ser colocadas em uma unidade de trabalho, se o mesmo manipulador de filas for usado. No entanto, elas não precisam ser colocadas na mesma unidade de trabalho, permitindo que um grupo de mensagens ou mensagem lógica que consiste em várias mensagens físicas seja dividida em duas ou mais unidades de trabalho consecutivas para o manipulador de filas.
- Se a primeira mensagem física em um grupo ou mensagem lógica não for colocada em uma unidade de trabalho, nenhuma das outras mensagens físicas no grupo ou mensagem lógica poderá ser colocada em uma unidade de trabalho se o mesmo manipulador de filas for usado.

Se estas condições não forem satisfeitas, a chamada MQPUT falhará com o código de razão MQRC_INCONSISTENT_UOW.

Quando MQPMO_LOGICAL_ORDER é especificado, o MQMD fornecido na chamada MQPUT não deve ser menor que MQMD_VERSION_2. Se essa condição não for satisfeita, a chamada falhará com o código de razão MQRC_WRONG_MD_VERSION.

Se MQPMO_LOGICAL_ORDER não for especificado, as mensagens em grupos e segmentos de mensagens lógicas poderão ser colocados em qualquer ordem e não é necessário colocar grupos de mensagens completas ou mensagens lógicas completas. É responsabilidade do aplicativo assegurar que os campos *GroupId*, *MsgSeqNumber*, *Offset*e *MsgFlags* tenham valores apropriados..

Use essa técnica para reiniciar um grupo de mensagens ou mensagem lógica no meio, após ocorrer uma falha do sistema. Quando o sistema é reiniciado, o aplicativo pode configurar os campos *GroupId*, *MsgSeqNumber*, *Offset*, *MsgFlags* e *Persistence* para os valores apropriados e, em seguida, emitir a chamada MQPUT com MQPMO_SYNCPOINT ou MQPMO_NO_SYNCPOINT configurado como necessário, mas sem especificar MQPMO_LOGICAL_ORDER. Se esta chamada for bem-sucedida, o gerenciador de filas retém as informações sobre o grupo e o segmento e chamadas MQPUT subsequentes usando esse manipulador de filas poderão especificar MQPMO_LOGICAL_ORDER como normal.

As informações de grupo e segmento que o manipulador de filas retém para a chamada MQPUT são separadas das informações de grupo e segmento que ele retém para a chamada MQGET.

Para qualquer manipulador de filas, o aplicativo pode misturar chamadas MQPUT que especificam MQPMO_LOGICAL_ORDER com chamadas MQPUT que não especificam, mas observe os seguintes pontos:

- Se MQPMO_LOGICAL_ORDER não for especificado, cada chamada MQPUT bem-sucedida faz com que o gerenciador de filas configure as informações de grupo e segmento para o manipulador de filas com os valores especificados pelo aplicativo, substituindo as informações de grupo e segmento existentes retidas pelo gerenciador de filas para o manipulador de filas.
- Se MQPMO_LOGICAL_ORDER não for especificado, a chamada não falhará se houver um grupo de mensagens atuais ou mensagens lógicas. A chamada pode ter êxito com um código de conclusão MQCC_WARNING.Tabela 37 na página 256 mostra os vários casos que podem surgir. Nesses casos, se o código de conclusão não for MQCC_OK, o código de razão é um dos seguintes (conforme apropriado):
  - MQRC_INCOMPLETE_GROUP
  - MQRC_INCOMPLETE_MSG
  - MQRC_INCONSISTENT_PERSISTENCE
  - MQRC_INCONSISTENT_UOW

Nota: O gerenciador de filas não verifica as informações de grupo e segmento para a chamada MQPUT1.

Tabela 37. Resultado quando a chamada MQPUT ou MQCLOSE não é consistente com as informações de grupo e segmento

A chamada atual é	A chamada anterior era MQPUT com MQPMO_LOGICAL_ORDER	A chamada anterior era MQPUT sem MQPMO_LOGICAL_ORDER
MQPUT com MQPMO_LOGICAL_ORDER	MQCC_FAILED	MQCC_FAILED
MQPUT sem MQPMO_LOGICAL_ORDER	MQCC_WARNING	MQCC_OK
MQCLOSE com um grupo ou mensagem lógica não terminada	MQCC_WARNING	MQCC_OK

Para aplicativos que colocam mensagens e segmentos em ordem lógica, especifique MQPMO_LOGICAL_ORDER, pois é a opção mais simples de usar. Esta opção livra o aplicativo da necessidade de gerenciar as informações de grupo e segmento, pois o gerenciador de filas gerencia essa informação. No entanto, os aplicativos especializados podem precisar de mais controle do que aquele fornecido pela opção MQPMO_LOGICAL_ORDER, que pode ser obtido não especificando essa opção; se você fizer isso, deverá assegurar que os campos*GroupId*, *MsgSeqNumber*, *Offset*e *MsgFlags* em MQMD sejam configurados corretamente, antes de cada chamada MQPUT ou MQPUT1 .

Por exemplo, um aplicativo que deseja redirecionar mensagens físicas que recebe, sem considerar a finalidade dessas mensagens nos grupos ou segmentos de mensagens lógicas, não deve especificar MQPMO_LOGICAL_ORDER por duas razões:

- Se as mensagens são recuperadas e colocadas em ordem, especificar MQPMO_LOGICAL_ORDER designa um novo identificador de grupo para as mensagens, que pode tornar difícil ou impossível para o originador das mensagens correlacionar quaisquer mensagens de resposta ou de relatório que resultem do grupo de mensagens.
- Em uma rede complexa com vários caminhos entre os gerenciadores de filas de envio e recebimento, as mensagens físicas podem chegar fora de ordem. Ao não especificar MQPMO_LOGICAL_ORDER e MQGMO_LOGICAL_ORDER na chamada MQGET, o aplicativo de redirecionamento pode recuperar e encaminhar cada mensagem física assim que ela chegar, sem aguardar a próxima na ordem lógica chegar.

Os aplicativos que geram mensagens de relatório para mensagens em grupos ou segmentos de mensagens lógicas também não devem especificar MQPMO_LOGICAL_ORDER ao colocar a mensagem de relatório.

MQPMO_LOGICAL_ORDER pode ser especificado com qualquer uma das outras opções MQPMO_*.

Colocando grupos ordenados de forma lógica em uma fila em cluster (MQOO_BIND_ON_GROUP)

A opção MQOO_BIND_ON_OPEN assegura que todas as mensagens deste aplicativo e, portanto, todos os grupos, sejam roteadas para uma única instância. A desvantagem é que o tráfego do aplicativo não tem a carga balanceada nas várias instâncias de uma fila de clusters. Para ativar o balanceamento de carga de trabalho enquanto mantém os grupos de mensagens intactos, deve-se configurar as opções a seguir:

- A chamada MQPUT deve especificar MQPMO_LOGICAL_ORDER
- A chamada MQOPEN deve especificar uma das duas opções a seguir:
  - MQOO_BIND_ON_GROUP
  - MQOO_BIND_AS_Q_DEF e a definição de fila deve especificar DEFBIND(GROUP)

O balanceamento de carga de trabalho é, então, direcionado *entre grupos* de mensagens sem precisar de um MQCLOSE e MQOPEN da fila. *Entre grupos* significa que MQMF_MSG_IN_GROUP é definido no MQMD(v2) ou MQMDE, e não há grupo parcialmente concluído em andamento. Quando um grupo está em andamento, o gerenciador de filas resolvido e o nome da fila na manipulação de objetos são reutilizados.

Se a mensagem anterior foi MO_LOGICAL_ORDER e/ou MQMF_MSG_IN_IN_GROUP foi definida mas a mensagem atual não faz parte do grupo, então a chamada PUT falhará com MQRC_INCOMPLETE_GROUP.

Se um MQPUT individual não especifica MQPMO_LOGICAL_ORDER e nenhum grupo atual está ativo, então o balanceamento de carga de trabalho é direcionado para essa mensagem (como se a chamada MQOPEN tivesse especificado MQOO_BIND_NOT_FIXED).

A não realocação ocorre para mensagens vinculadas a um destino usando MQOO_BIND_ON_GROUP. Para obter mais informações sobre a realocação, consulte "Grupos de mensagens" na página 36.

Agrupando mensagens lógicas

Há duas razões principais para usar mensagens lógicas em um grupo:

- Talvez seja necessário processar as mensagens em uma ordem específica.
- Talvez seja necessário processar cada mensagem em um grupo de forma relacionada.

Em qualquer caso, recupere o grupo inteiro com a mesma instância do aplicativo de obtenção.

Por exemplo, suponha que o grupo consista em quatro mensagens lógicas. O aplicativo de colocação será semelhante ao seguinte:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP

MQCMIT
```

O aplicativo de obtenção especifica a opção MQGMO_ALL_MSGS_AVAILABLE para a primeira mensagem no grupo. Isso assegura que o processamento não seja iniciado até que todas as mensagens no grupo tenham chegado. A opção MQGMO_ALL_MSGS_AVAILABLE é ignorada para mensagens subsequentes dentro do grupo.

Quando a primeira mensagem lógica do grupo é recuperada, é possível usar MQGMO_LOGICAL_ORDER para assegurar que as mensagens lógicas restantes do grupo sejam recuperadas em ordem.

Assim, o aplicativo de obtenção é semelhante ao seguinte:

```
/* Wait for the first message in a group, or a message not in a group */
GMO.Options = MQGMO_SYNCPOINT | MQGMO_WAIT
            | MQGMO_ALL_MSGS_AVAILABLE | MQGMO_LOGICAL_ORDER
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Process each remaining message in the group */
   ...

MQCMIT
```

Para obter exemplos adicionais de como agrupar mensagens, consulte "Segmentação de mensagens lógicas do aplicativo" na página 268 e "Colocando e obtendo um grupo que abrange unidades de trabalho" na página 257.

Para obter informações sobre como permitir que um aplicativo solicite que um grupo de mensagens sejam todas alocadas para a mesma instância de destino para filas de clusters, consulte DefBind.

Colocando e obtendo um grupo que abrange unidades de trabalho

No caso anterior, mensagens ou segmentos não podem começar a deixar o nó (se seu destino for remoto) ou começar a ser recuperados até que o grupo inteiro tenha sido colocado e a unidade de trabalho estiver

confirmada. Isto pode não ser o desejado se levar muito tempo para colocar o grupo inteiro ou se o espaço da fila estiver limitado no nó. Para superar isso, coloque o grupo em várias unidades de trabalho.

Se o grupo for colocado em diversas unidades de trabalho, é possível que parte do grupo confirme mesmo quando o aplicativo de colocação falhar. O aplicativo deve, portanto, salvar informações de status, confirmada em cada unidade de trabalho, que ele poderá usar após um reinício para continuar um grupo incompleto. O local mais simples para registrar essas informações é em uma fila STATUS. Se um grupo completo tiver sido colocado com êxito, a fila STATUS estará vazia.

Se a segmentação estiver envolvida, a lógica será semelhante. Nesse caso, o StatusInfo deve incluir o *Offset*

Aqui está um exemplo de colocação do grupo em diversas unidades de trabalho:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

/* First UOW */

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
StatusInfo = GroupId,MsgSeqNumber from MQMD
MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
MQCMIT

/* Next and subsequent UOWs */
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
StatusInfo = GroupId,MsgSeqNumber from MQMD
MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
MQCMIT

/* Last UOW */
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
MQCMIT
```

Se todas as unidades de trabalho forem confirmadas, o grupo inteiro foi colocado com êxito e a fila STATUS está vazia. Se não, o grupo deve ser continuado do ponto indicado pelas informações de status. MQPMO_LOGICAL_ORDER não pode ser usado para a primeira colocação, mas pode posteriormente.

O processamento de reinicialização é semelhante a este:

```
MQGET (StatusInfo from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
if (Reason == MQRC_NO_MSG_AVAILABLE)
   /* Proceed to normal processing */
   ...

else
   /* Group was terminated prematurely */
   Set GroupId, MsgSeqNumber in MQMD to values from Status message
   PMO.Options = MQPMO_SYNCPOINT
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP

   /* Now normal processing is resumed.
      Assume this is not the last message */
   PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
   StatusInfo = GroupId,MsgSeqNumber from MQMD
   MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
   MQCMIT
```

No aplicativo de obtenção, você talvez queira iniciar o processamento das mensagens em um grupo antes que o grupo inteiro tenha chegado. Isso melhora tempos de resposta nas mensagens dentro do grupo e também significa que o armazenamento não é requerido para o grupo inteiro. Para realizar os benefícios,

use diversas unidades de trabalho para cada grupo de mensagens. Por razões de recuperação, deve-se recuperar cada mensagem em uma unidade de trabalho.

Como com o aplicativo de colocação correspondente, isto requer que informações de status sejam registradas em algum lugar automaticamente conforme cada unidade de trabalho é confirmada. Novamente, o local mais simples para registrar estas informações estão em uma fila STATUS. Se um grupo completo foi processado com êxito, a fila STATUS estará vazia.

Nota: Para unidades de trabalho intermediárias, é possível evitar as chamadas MQGET a partir da fila STATUS especificando que cada chamada MQPUT para a fila de status é um segmento de uma mensagem (ou seja, configurando a sinalização MQMF_SEGMENT) em vez de colocar uma mensagem nova completa para cada unidade de trabalho. Na última unidade de trabalho, um segmento final é colocado na fila de status especificando MQMF_LAST_SEGMENT e, em seguida, as informações de status são limpas com um MQGET especificando MQGMO_COMPLETE_MSG.

Durante o processamento de reinicialização, em vez de usar um único MQGET para obter uma mensagem de status possível, navegue na fila de status com MQGMO_LOGICAL_ORDER até que você atinja o último segmento (ou seja, até que nenhum segmentos adicionais sejam retornados). Na primeira unidade de trabalho após a reinicialização, especifique também o deslocamento explicitamente ao colocar o segmento de status.

No exemplo a seguir, consideramos apenas as mensagens em um grupo, assumindo que buffer do aplicativo é sempre grande o suficiente para conter a mensagem inteira, seja ela segmentada ou não. Portanto, MQGMO_COMPLETE_MSG é especificado em cada MQGET. Os mesmos princípios se aplicam se a segmentação estiver envolvida (nesse caso, StatusInfo deve incluir o *Offset*).

Para simplificar, assumimos que um máximo de 4 mensagens são recuperadas dentro de uma única UOW:

```
msgs = 0    /* Counts messages retrieved within UOW */
/* Should be no status message at this point */

/* Retrieve remaining messages in the group */
do while ( GroupStatus == MQGS_MSG_IN_GROUP )

   /* Process up to 4 messages in the group */
   GMO.Options = MQGMO_SYNCPOINT | MQGMO_WAIT
               | MQGMO_LOGICAL_ORDER
   do while ( (GroupStatus == MQGS_MSG_IN_GROUP) && (msgs < 4) )
      MQGET
      msgs = msgs + 1
      /* Process this message */
      ...
   /* end while

   /* Have retrieved last message or 4 messages  */
   /* Update status message if not last in group */
   MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
   if ( GroupStatus == MQGS_MSG_IN_GROUP )
      StatusInfo = GroupId,MsgSeqNumber from MQMD
      MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
   MQCMIT
   msgs = 0
/* end while

if ( msgs > 0 )
   /* Come here if there was only 1 message in the group */
   MQCMIT
```

Se todas as unidades de trabalho tiverem sido confirmadas, o grupo inteiro foi recuperado com êxito e a fila STATUS está vazia. Se não, o grupo deve ser continuado do ponto indicado pelas informações de status. MQGMO_LOGICAL_ORDER não pode ser usado para o primeiro recuperar, mas pode posteriormente.

O processamento de reinicialização é semelhante a este:

```
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
if (Reason == MQRC_NO_MSG_AVAILABLE)
   /* Proceed to normal processing */
```

```
      ...

   else
      /* Group was terminated prematurely */
      /* The next message on the group must be retrieved by matching
         the sequence number and group id with those retrieved from the
         status information. */
      GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT | MQGMO_WAIT
      MQGET GMO.MatchOptions = MQMO_MATCH_GROUP_ID | MQMO_MATCH_MSG_SEQ_NUMBER,
            MQMD.GroupId      = value from Status message,
            MQMD.MsgSeqNumber = value from Status message plus 1
      msgs = 1
      /* Process this message */
      ...

      /* Now normal processing is resumed */
      /* Retrieve remaining messages in the group */
      do while ( GroupStatus == MQGS_MSG_IN_GROUP )

         /* Process up to 4 messages in the group */
         GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT | MQGMO_WAIT
                     | MQGMO_LOGICAL_ORDER
         do while ( (GroupStatus == MQGS_MSG_IN_GROUP) && (msgs < 4) )
            MQGET
            msgs = msgs + 1
            /* Process this message */
            ...

         /* Have retrieved last message or 4 messages  */
         /* Update status message if not last in group */
         MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
         if ( GroupStatus == MQGS_MSG_IN_GROUP )
            StatusInfo = GroupId,MsgSeqNumber from MQMD
            MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
         MQCMIT
         msgs = 0
```

Obtendo uma mensagem específica

Existem diversas formas de obter uma mensagem específica de uma fila. São elas: selecionando em MsgId e CorrelId, selecionando em GroupId, MsgSeqNumber e Offset e selecionando em MsgToken. É possível também usar uma sequência de seleção ao abrir a fila.

Para obter uma mensagem específica de uma fila, use os campos *MsgId* e *CorrelId* da estrutura MQMD. No entanto, os aplicativos podem configurar explicitamente esses campos, portanto, os valores que você especifica podem não identificar uma mensagem exclusiva. O Tabela 38 na página 260 mostra qual mensagem é recuperada para as configurações possíveis desses campos. Esses campos serão ignorados na entrada se você especificar MQGMO_MSG_UNDER_CURSOR no parâmetro *GetMsgOpts* da chamada MQGET.

Tabela 38. Usando identificadores de mensagem e de correlação

Para recuperar...	***MsgId***	***CorrelId***
Primeira mensagem na fila	MQMI_NONE	MQCI_NONE
Primeira mensagem que corresponde a *MsgId*	Diferente de zero	MQCI_NONE
Primeira mensagem que corresponde a *CorrelId*	MQMI_NONE	Diferente de zero
Primeira mensagem que corresponde a *MsgId* e *CorrelId*	Diferente de zero	Diferente de zero

Em cada caso, *primeiro* significa a primeira mensagem que satisfaz os critérios de seleção (a menos que MQGMO_BROWSE_NEXT seja especificado, quando significa a *próxima* mensagem na sequência que satisfaz os critérios de seleção).

No retorno, a chamada MQGET configura os campos *MsgId* e *CorrelId* como os identificadores de mensagem e de correlação da mensagem retornada, se houver.

Se você configurar o campo *Version* da estrutura MQMD como 2, poderá usar os campos *GroupId*, *MsgSeqNumber* e *Offset*.Tabela 39 na página 261 mostra qual mensagem é recuperada para as configurações possíveis desses campos

Tabela 39. Usando o identificador de grupo

Para recuperar...	**Opções de correspondência**
Primeira mensagem na fila	MQMO_NONE
Primeira mensagem que corresponde a *MsgId*	MQMO_MATCH_MSG_ID
Primeira mensagem que corresponde a *CorrelId*	MQMO_MATCH_CORREL_ID
Primeira mensagem que corresponde a *GroupId*	MQMO_MATCH_GROUP_ID
Primeira mensagem que corresponde a *MsgSeqNumber*	MQMO_MATCH_MSG_SEQ_NUMBER
Primeira mensagem que corresponde a *MsgToken*	MQMO_MATCH_MSG_TOKEN
Primeira mensagem que corresponde a *Offset*	MQMO_MATCH_OFFSET

Notes:

1. MQMO_MATCH_XXX implica que o campo *XXX* na estrutura MQMD é configurado como o valor a ser correspondido.
2. Os sinalizadores MQMO podem ser usados em combinação. Por exemplo, MQMO_MATCH_GROUP_ID, MQMO_MATCH_MSG_SEQ_NUMBER e MQMO_MATCH_OFFSET podem ser usados juntos para fornecer o segmento identificado pelos campos *GroupId*, *MsgSeqNumber* e *Offset*.
3. Se especificar MQGMO_LOGICAL_ORDER, a mensagem que você está tentando recuperar será afetada porque a opção depende das informações de estado controladas para o manipulador de filas. Para obter informações sobre isso, consulte "Ordenação lógica e física" na página 249 e Opções.

A chamada MQGET, geralmente, recupera a primeira mensagem de uma fila. Se você especificar uma determinada mensagem ao usar a chamada MQGET, o gerenciador de filas deverá procurar na fila até localizar essa mensagem. Isso pode afetar o desempenho do seu aplicativo.

Se você estiver usando a Versão 2 ou posterior da estrutura MQGMO e não especificar os sinalizadores MQMO_MATCH_MSG_ID ou MQMO_MATCH_CORREL_ID, não será necessário reconfigurar os campos *MsgId* ou *CorrelId* entre MQGETs.

É possível obter uma mensagem específica de uma fila especificando seu MsgToken e o MatchOption MQMO_MATCH_MSG_TOKEN na estrutura MQGMO. O MsgToken é retornado pela chamada MQPUT que originalmente colocou essa mensagem na fila ou por operações MQGET anteriores e permanece constante, a menos que o gerenciador de filas seja reiniciado.

Se você estiver interessado em apenas um subconjunto de mensagens na fila, poderá especificar quais mensagens deseja processar usando uma sequência de seleção com a chamada MQOPEN ou MQSUB. MQGET, então, recupera a próxima mensagem que satisfaz essa sequência de seleção. Para obter informações adicionais sobre sequências de seleção, consulte "Seletores" na página 23.

Melhorando o desempenho de mensagens não persistentes

Quando um cliente requer uma mensagem de um servidor, ele envia uma solicitação ao servidor. Ele envia uma solicitação separada para cada uma das mensagens que consome. Para melhorar o desempenho de um cliente que consome mensagens não persistentes evitando a necessidade de enviar essas mensagens de solicitação, um cliente pode ser configurado para usar *leia mais adiante*. Leia mais adiante permite que mensagens sejam enviadas a um cliente sem que um aplicativo precise solicitá-las.

Quando a opção leia mais adiante estiver ativada, as mensagens são enviadas para um buffer de memória no cliente chamado de buffer de *leia mais adiante*. O cliente terá um buffer de leia mais adiante para cada fila que tiver aberto com leia mais adiante ativado. As mensagens no buffer de leia mais adiante não são

persistidas. O cliente atualiza periodicamente o servidor com informações sobre a quantia de dados que consumiu.

Quando você chama MQOPEN com MQOO_READ_AHEAD, o cliente do WebSphere MQ só permite a leitura antecipada se determinadas condições forem atendidas Essas condições incluem:

- Tanto o cliente quanto o gerenciador de filas remotas deve estar em WebSphere MQ Versão 7 ou posterior.
- O aplicativo cliente deve ser compilado e vinculado às bibliotecas do cliente MQI do WebSphere MQ encadeado.
- O canal do cliente deve estar utilizando o protocolo TCP/IP
- O canal deve ter a configuração de SharingConversations diferente de zero (SHARECNV) em ambas as definições de canal, do cliente e do servidor.

Usar leia mais adiante pode melhorar o desempenho ao consumir mensagens não persistentes a partir de um aplicativo cliente. Essa melhoria de desempenho está disponível para os aplicativos MQI e JMS. Aplicativos clientes que usam MQGET ou consumo assíncrono se beneficiarão das melhorias de desempenho ao consumir mensagens não persistentes.

Nem todos os designs de aplicativo cliente são adequados para usar leia mais adiante, pois nem todas as opções são suportadas para uso com leia mais adiante e algumas opções precisam ser consistentes entre chamadas MQGET quando leia mais adiante está ativado. Se um cliente alterar seus critérios de seleção entre chamadas MQGET, as mensagens que estão sendo armazenadas no buffer de leia mais adiante permanecerão conectadas ao buffer de leia mais adiante do cliente.

Se uma lista não processada de mensagens conectadas com os critérios de seleção anteriores não for mais necessária, um intervalo de limpeza configurável pode ser configurado no cliente para limpar automaticamente essas mensagens do cliente. O intervalo de limpeza é uma de um grupo de opções de ajuste de leia mais adiante determinadas pelo cliente. É possível ajustar essas opções para atender seus requisitos.

Se um aplicativo cliente for reiniciado, as mensagens no buffer de leia mais adiante poderão ser perdidas. Por outro lado, uma mensagem que foi movida para um buffer de leia mais adiante poderia, então, ser excluída da fila subjacente; isso não resulta em sua remoção do buffer, portanto, uma chamada MQGET usando leia mais adiante pode retornar uma mensagem que não existe mais.

Leia mais adiante é executado somente para ligações com o cliente. O atributo é ignorado para todas as outras ligações.

Leia mais adiante não tem efeito sobre o acionamento. Nenhuma mensagem do acionador é gerada quando o cliente executa leia mais adiante de uma mensagem. Leia mais adiante não gera informações de contabilidade e estatísticas quando está ativado.

Usando leia mais adiante com sistema de mensagens de assinatura de publicação

Quando um aplicativo de assinatura especifica uma fila de destino para a qual publicações são enviadas, o valor de DEFREADA da fila especificada é usado como o valor padrão de leia mais adiante.

Quando um aplicativo de assinatura solicita que o WebSphere MQ gerencie o destino para o qual as publicações são enviadas, uma fila gerenciada é criada como uma fila dinâmica baseada em uma fila modelo predefinida. É o valor de DEFREADA da fila modelo que é usado como o valor padrão de leia mais adiante. As filas modelo padrão SYSTEM.DURABLE.PUBLICATIONS.MODEL ou SYSTEM.NONDURABLE.PUBLICATIONS.MODEL são usadas, a menos que uma fila modelo seja definido para este ou um tópico pai.

Conceitos relacionados

“Ajustando o desempenho para mensagens não persistentes no AIX” na página 265
Se você estiver usando o AIX V5.3 ou posterior, considere configurar o parâmetro de ajuste para usar o desempenho integral para mensagens não persistentes.

Tarefas relacionadas

“Ativando e desativando leia mais adiante” na página 264

Por padrão leia mais adiante está desativado. É possível ativar leia mais adiante no nível da fila ou do aplicativo.

Referências relacionadas

"Opções de MQGET e leia mais adiante" na página 263
Nem todas as opções MQGET são suportadas quando leia mais adiante está ativada; algumas opções precisam ser consistentes entre chamadas MQGET.

Opções de MQGET e leia mais adiante

Nem todas as opções MQGET são suportadas quando leia mais adiante está ativada; algumas opções precisam ser consistentes entre chamadas MQGET.

Quando você chama MQOPEN com MQOO_READ_AHEAD, o cliente do WebSphere MQ só permite a leitura antecipada se determinadas condições forem atendidas Essas condições incluem:

- Tanto o cliente quanto o gerenciador de filas remotas deve estar em WebSphere MQ Versão 7 ou posterior.
- O aplicativo cliente deve ser compilado e vinculado às bibliotecas do cliente MQI do WebSphere MQ encadeado.
- O canal do cliente deve estar utilizando o protocolo TCP/IP
- O canal deve ter a configuração de SharingConversations diferente de zero (SHARECNV) em ambas as definições de canal, do cliente e do servidor.

A tabela a seguir indica quais opções são suportadas para uso com leia mais adiante e se elas podem ser alteradas entre chamadas MQGET.

Tabela 40. Opções de MQGET e leia mais adiante

	Permitido quando a leitura antecipada está ativada e pode ser alterada entre chamadas MQGET[5]	**Permitido quando a leitura antecipada está ativada e não pode ser alterada entre chamadas MQGET[1]**	**Opções MQGET que não são permitidas quando a leitura antecipada estiver ativada[2]**
Valores de MQGET MQMD	MsgId[3] CorrelId[3]	Encoding CodedCharSetId	
Opções de MQGET MQGMO	• MQGMO_NO_WAIT • MQGMO_BROWSE_MESSAGE _UNDER_CURSOR • MQGMO_BROWSE_FIRST • MQGMO_BROWSE_NEXT • MQGMO_FAIL_IF_QUIESCING	• MQGMO_SYNCPOINT_IF _PERSISTENT • MQGMO_NO_SYNCPOINT • MQGMO_ACCEPT_TRUNCATED_MSG • MQGMO_CONVERT	• MQGMO_SET_SIGNAL • MQGMO_SYNCPOINT • MQGMO_MARK_SKIP _BACKOUT • MQGMO_MSG_UNDER _CURSOR[4] • MQGMO_LOCK • MQGMO_UNLOCK • MQGMO_LOGICAL_ORDER • MQGMO_COMPLETE_MSG • MQGMO_ALL_MSGS_AVAILABLE • MQGMO_ALL_SEGMENTS_ AVAILABLE

Notes:

1. Se essas opções forem mudadas entre chamadas MQGET, um código de razão MQRC_OPTIONS_CHANGED será retornado.
2. Se estas opções forem especificadas na primeira chamada MQGET, a leitura antecipada é desativada. Se essas opções forem especificadas em uma chamada MQGET subsequente, um código de razão MQRC_OPTIONS_ERROR será retornado.
3. Se um aplicativo cliente alterar os valores de MsgId e CorrelId entre chamadas MQGET, mensagens com os valores anteriores poderão já ter sido enviadas ao cliente e permanecerão no buffer de leita mais adiante do cliente até serem consumidas (ou limpas automaticamente).
4. MQGMO_MSG_UNDER_CURSOR não é possível com a leitura antecipada. Leia mais adiante está desativada quando MQOO_BROWSE e uma das opções MQOO_INPUT_SHARED ou MQOO_INPUT_EXCLUSIVE são especificadas ao abrir a fila.
5. Quando a leitura antecipada está ativada, a primeira MQGET determina se mensagens devem ser procuradas ou obtidas de uma fila. Se o aplicativo cliente usar então MQGET com opções mudadas,

como uma tentativa de procura após um get inicial ou tentar efetuar get após uma procura inicial, um código de razão MQRC_OPTIONS_CHANGED será retornado.

Se um cliente alterar seus critérios de seleção entre chamadas MQGET, as mensagens que estão sendo armazenadas no buffer de leia mais adiante que correspondem aos critérios de seleção iniciais não serão consumidas pelo aplicativo cliente e permanecerão conectadas ao buffer de leia mais adiante do cliente. Em situações em que o buffer de leia mais adiante do cliente contém muitas mensagens conectadas, os benefícios associados a leia mais adiante serão perdidos e uma solicitação separada para o servidor será necessária para cada mensagem consumida. Para determinar se leia mais adiante está sendo usada de forma eficiente, é possível usar o parâmetro do status da conexão, READA.

Leia mais adiante pode ser inibida quando solicitado por um aplicativo devido às opções incompatíveis especificadas na primeira chamada MQGET. Nessa situação, o status da conexão mostra leia mais adiante como sendo inibida.

Se, devido a essas restrições em MQGET, você decidir que o design de um aplicativo cliente não é adequado para leia mais adiante, especifique a opção MQOO_READ_AHEAD_NO de MQOPEN. Como alternativa, configure o valor de leia mais adiante padrão da fila que está sendo aberta alterado para NO ou DISABLED.

Ativando e desativando leia mais adiante

Por padrão leia mais adiante está desativado. É possível ativar leia mais adiante no nível da fila ou do aplicativo.

Sobre esta tarefa

Quando você chama MQOPEN com MQOO_READ_AHEAD, o cliente do WebSphere MQ só permite a leitura antecipada se determinadas condições forem atendidas Essas condições incluem:

- Tanto o cliente quanto o gerenciador de filas remotas deve estar em WebSphere MQ Versão 7 ou posterior.
- O aplicativo cliente deve ser compilado e vinculado às bibliotecas do cliente MQI do WebSphere MQ encadeado.
- O canal do cliente deve estar utilizando o protocolo TCP/IP
- O canal deve ter a configuração de SharingConversations diferente de zero (SHARECNV) em ambas as definições de canal, do cliente e do servidor.

Para ativar leia mais adiante:

- Para configurar leia mais adiante no nível da fila, configure o atributo da fila, DEFREADA para YES.
- Para configurar leia mais adiante no nível do aplicativo:
  - para usar leia mais adiante sempre que possível, use a opção MQOO_READ_AHEAD na chamada de função MQOPEN. Não é possível para o aplicativo cliente usar leia mais adiante se o atributo da fila DEFREADA tiver sido configurado para DISABLED.
  - para usar leia mais adiante somente quando leia mais adiante estiver ativado em uma fila, use a opção MQOO_READ_AHEAD_AS_Q_DEF na chamada de função MQOPEN.

Se um design do aplicativo cliente não for adequado para leia mais adiante, é possível desativá-lo:

- no nível da fila, configurando o atributo da fila DEFREADA para NO, se não desejar que leia mais adiante seja usado, a menos que seja solicitado por um aplicativo cliente ou DISABLED, se não desejar que leia mais adiante seja usado independentemente de se leia mais adiante é necessário para um aplicativo cliente.
- no nível do aplicativo, usando a opção MQOO_NO_READ_AHEAD na chamada de função MQOPEN.

Duas opções MQCLOSE permitem configurar o que acontece com as mensagens que estão sendo armazenadas no buffer de leia mais adiante se a fila for fechada.

- Use MQCO_IMMEDIATE para descartar as mensagens no buffer de leia mais adiante.

- Use MQCO_QUIESCE para assegurar que as mensagens no buffer de leia mais adiante sejam consumidas pelo aplicativo antes da fila ser fechada. Quando MQCLOSE com MQCO_QUIESCE é emitido e há mensagens restantes no buffer de leia mais adiante, MQRC_READ_AHEAD_MSGS retorna com MQCC_WARNING.

Ajustando o desempenho para mensagens não persistentes no AIX

Se você estiver usando o AIX V5.3 ou posterior, considere configurar o parâmetro de ajuste para usar o desempenho integral para mensagens não persistentes.

Para configurar o parâmetro de ajuste de modo que ele entre em vigor imediatamente, emita o comando a seguir como um usuário raiz:

```
/usr/sbin/ioo -o j2_nPagesPerWriteBehindCluster=0
```

Para configurar o parâmetro de ajuste de modo que ele entra em vigor imediatamente e persista sobre as reinicializações, emita o comando a seguir como um usuário raiz:

```
/usr/sbin/ioo -p -o j2_nPagesPerWriteBehindCluster=0
```

Normalmente, as mensagens não persistentes são mantidas apenas na memória, mas há circunstâncias em que o AIX pode planejar que as mensagens não persistentes sejam gravadas no disco. As mensagens planejadas para serem gravadas em disco estão indisponíveis para MQGET até que a gravação do disco esteja concluída. O comando de ajuste sugerido varia esse limite; em vez de planejar que as mensagens sejam gravadas no disco quando 16 kilobytes de dados são enfileirados, a gravação em disco ocorre apenas quando o armazenamento real na máquina se torna quase cheio. Esta é uma alteração global e pode afetar outros componentes de software.

No AIX, ao usar aplicativos multiencadeados e principalmente ao executar em máquinas com múltiplos processadores, é altamente recomendável configurar AIXTHREAD_SCOPE=S no ID mqm `.profile` ou configurar AIXTHREAD_SCOPE=S no ambiente antes de iniciar o aplicativo, para obter um desempenho melhor e um planejamento mais sólido. Por exemplo:

```
export AIXTHREAD_SCOPE=S
```

Configurar AIXTHREAD_SCOPE=S significa que os encadeamentos de usuário criados com atributos padrão são colocados no escopo de contenção de todo o sistema. Se um encadeamento de usuários for criado com o escopo de contenção para todo o sistema, ele será ligado a um encadeamento kernel e planejado pelo kernel. O encadeamento kernel subjacente não é compartilhado com nenhum encadeamento de usuário.

Descritores de Arquivos

Ao executar um processo de encadeamento múltiplo, como o processo do agente, você pode alcançar o limite flexível para descritores de arquivos. Esse limite fornece o IBM WebSphere MQ código de razão `MQRC_UNEXPECTED_ERROR (2195)` e, se houver descritores de arquivo suficientes, um arquivo IBM WebSphere MQ FFST™

Para evitar esse problema, é possível aumentar o limite de processo para o número de descritores de arquivo. Para isso, altere o atributo `nofiles` em `/etc/security/limits` para 10.000 para o ID do usuário mqm ou na sub-rotina padrão.

Limites de recursos do sistema

Defina o limite de recursos do sistema para segmento de dados e segmento de pilha como ilimitado utilizando os seguintes comandos em um prompt de comandos:

```
ulimit -d unlimited
ulimit -s unlimited
```

Manipulando mensagens com mais de 4 MB de comprimento

As mensagens podem ser muito grandes para o aplicativo, fila ou gerenciador de filas. Dependendo do ambiente, o WebSphere MQ fornece várias maneiras de lidar com mensagens maiores que 4 MB.

É possível aumentar o atributo *MaxMsgLength* até 100 MB em todos os sistemas WebSphere MQ em V6 ou posterior. Configure esse valor para refletir o tamanho das mensagens usando a fila. No WebSphere MQ sistemas diferentes de WebSphere MQ para z/OS, também é possível:

1. Usar mensagens segmentadas. (As mensagens podem ser segmentadas pelo aplicativo ou pelo gerenciador de filas.)
2. Usar mensagens de referência.

Cada uma dessas abordagens é descrita no restante desta seção.

Aumentar o comprimento máximo da mensagem

O atributo *MaxMsgLength* do gerenciador de filas define o comprimento máximo de uma mensagem que pode ser manipulada por um gerenciador de filas. De forma semelhante, o atributo da fila *MaxMsgLength* é o comprimento máximo de uma mensagem que pode ser manipulada por uma fila. O comprimento máximo da mensagem *padrão* suportado depende do ambiente no qual você está trabalhando.

Se você estiver manipulando mensagens grandes, será possível alterar esses atributos de forma independente. É possível configurar o valor do atributo do gerenciador de filas no intervalo de 32768 bytes a 100 MB; é possível o valor do atributo da fila no intervalo de 0 a 100 MB.

Após mudar um ou ambos os atributos *MaxMsgLength*, reinicie seus aplicativos e canais para assegurar que as mudanças entrem em vigor.

Quando estas mudanças forem feitas, o comprimento da mensagem deve ser menor ou igual aos atributos *MaxMsgLength* da fila e do gerenciador de filas. No entanto, mensagens existentes podem ser mais longas do que qualquer um desses dois atributos.

Se a mensagem for muito grande para a fila, MQRC_MSG_TOO_BIG_FOR_Q será retornado.
De forma semelhante, se a mensagem for muito grande para o gerenciador de filas, MQRC_MSG_TOO_BIG_FOR_Q_MGR será retornado.

Esse método de manipulação de mensagens grandes é fácil e conveniente. No entanto, considere os fatores a seguir antes de usá-lo:

- A uniformidade entre os gerenciadores de filas é reduzida. O tamanho máximo de dados da mensagem é determinado por *MaxMsgLength* para cada fila (incluindo filas de transmissão) em que a mensagem será colocada. Esse valor geralmente é padronizado para o *MaxMsgLength* do gerenciador de filas, principalmente, para filas de transmissão. Isso dificulta prever se uma mensagem é muito grande quando for viajar para um gerenciador de filas remotas.
- O uso de recursos do sistema é aumentado. Por exemplo, os aplicativos precisam de buffers maiores e, em algumas plataformas, poderá haver aumento no uso de armazenamento compartilhado. O armazenamento de fila deve ser afetado somente se realmente necessário para mensagens maiores.
- Lote do canal é afetado. Uma grande mensagem ainda conta como apenas uma mensagem para a contagem de lotes, mas precisa de mais tempo para transmissão, aumentando assim os tempos de resposta para outras mensagens.

Segmentação de mensagem

Use estas informações para aprender sobre a segmentação de mensagens.

Nota: Não suportado em IBM WebSphere MQ para z/OS ou por aplicativos usando classes IBM WebSphere MQ para JMS.

Aumentar o comprimento máximo da mensagem, conforme explicado no tópico "Aumentar o comprimento máximo da mensagem" na página 266 tem algumas implicações negativas. Além disso, ainda pode resultar na mensagem ser muito grande para a fila ou para o gerenciador de filas. Nesses

casos, é possível segmentar uma mensagem. Para obter informações sobre segmentos, consulte "Grupos de mensagens" na página 36.

As próximas seções verificam usos comuns para segmentação de mensagens. Para put e get destrutivo, supõe-se que as chamadas MQPUT ou MQGET *sempre* operam dentro de uma unidade de trabalho. Sempre considere usar essa técnica para reduzir a possibilidade de grupos incompletos estarem presentes na rede. Single-phase commit pelo gerenciador de filas é assumido, mas outras técnicas de coordenação são igualmente válidas.

Além disso, nos aplicativos de get, supõe-se que se vários servidores estiverem processando a mesma fila, cada servidor executa um código semelhante, de forma que um servidor nunca deixe de localizar uma mensagem ou um segmento que espera que esteja lá (porque havia especificado MQGMO_ALL_MSGS_AVAILABLE ou MQGMO_ALL_SEGMENTS_AVAILABLE anteriormente).

Efetuando put e get de uma mensagem segmentada que se estende por unidades de trabalho

É possível efetuar put e get de uma mensagem segmentada que se estende por uma unidade de trabalho de maneira semelhante a "Colocando e obtendo um grupo que abrange unidades de trabalho" na página 257.

Não é possível, no entanto, efetuar put e get de mensagens segmentadas em uma unidade de trabalho global.

Segmentação e remontagem pelo gerenciador de filas
Esse é o cenário mais simples, no qual um aplicativo coloca uma mensagem a ser recuperada por outro. A mensagem pode ser grande: não muito grande para que o aplicativo put ou get manipule em um único buffer, mas muito grande para o gerenciador de filas ou uma fila na qual a mensagem deve ser colocada.

As únicas mudanças necessárias para esses aplicativos são para que o aplicativo put autorize o gerenciador de filas a executar a segmentação se necessário:

```
PMO.Options = (existing options)
MD.MsgFlags = MQMF_SEGMENTATION_ALLOWED
memcpy(MD.GroupId, MQGI_NONE, MQ_GROUP_ID_LENGTH)
MQPUT
```

e para o aplicativo get solicite ao gerenciador de filas que remonte a mensagem se ela foi segmentada:

```
GMO.Options = MQGMO_COMPLETE_MSG | (existing options)
MQGET
```

Neste cenário mais simples, o aplicativo deve reconfigurar o campo GroupId para MQGI_NONE antes da chamada MQPUT, para que o gerenciador de filas possa gerar um identificador de grupo exclusivo para cada mensagem. Se isso não for feito, mensagens não relacionadas poderão ter o mesmo identificador de grupo, que pode, subsequentemente, levar ao processamento incorreto.

O buffer do aplicativo deve ser grande o suficiente para conter a mensagem remontada (a menos que você inclua a opção MQGMO_ACCEPT_TRUNCATED_MSG).

Se o atributo MAXMSGLEN de uma fila tiver de ser modificado para acomodar a segmentação de mensagens, considere:

- O segmento mínimo da mensagem suportado em uma fila local é 16 bytes.
- Para uma fila de transmissão, MAXMSGLEN também deve incluir o espaço necessário para os cabeçalhos. Considere usar um valor pelo menos 4000 bytes maior que o comprimento máximo esperado de dados do usuário em qualquer segmento de mensagem que poderia ser colocado em uma fila de transmissão.

Se a conversão de dados for necessária, o aplicativo get pode ter que fazer isso especificando MQGMO_CONVERT. Isso deve ser simples porque a saída de conversão de dados é apresentada com a mensagem completa. Não tente converter os dados em um canal do emissor se a mensagem estiver

segmentada e o formato dos dados forem de tal forma que a saída de conversão de dados não puder fazer a conversão em dados incompletos.

Segmentação do aplicativo
A segmentação do aplicativo é usada quando a segmentação do gerenciador de filas não é adequada ou quando aplicativos requerem conversão de dados com limites de segmentos específicos.

A segmentação do aplicativo é usada por duas razões principais:

1. A segmentação do gerenciador de filas sozinha não é adequada porque a mensagem é muito grande para ser manipulada em um único buffer pelos aplicativos.
2. A conversão de dados deve ser executada pelos canais emissores e o formato é tal que o aplicativo de put deve estipular onde devem ser os limites dos segmentos para que a conversão de um segmento individual seja possível.

No entanto, se a conversão de dados não for um problema ou se o aplicativo de get sempre usar MQGMO_COMPLETE_MSG, a segmentação do gerenciador de filas também poderá ser permitida especificando MQMF_SEGMENTATION_ALLOWED. Em nosso exemplo, o aplicativo segmenta a mensagem em quatro segmentos:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_SEGMENT

MQCMIT
```

Se não usar MQPMO_LOGICAL_ORDER, o aplicativo deverá configurar *Offset* e o comprimento de cada segmento. Neste caso, o estado lógico não é mantido automaticamente.

O aplicativo de get não pode garantir ter um buffer grande o suficiente para conter qualquer mensagem remontada. Deve, portanto, estar preparado para processar os segmentos individualmente.

Para mensagens que são segmentadas, esse aplicativo não deseja iniciar o processamento de um segmento até que todos os segmentos que constituem a mensagem lógica estejam presentes. MQGMO_ALL_SEGMENTS_AVAILABLE é, portanto, especificado para o primeiro segmento. Se você especificar MQGMO_LOGICAL_ORDER e houver uma mensagem lógica atual, MQGMO_ALL_SEGMENTS_AVAILABLE será ignorado.

Após o primeiro segmento de uma mensagem lógica ter sido recuperado, use MQGMO_LOGICAL_ORDER para assegurar que os segmentos restantes da mensagem lógica serão recuperados em ordem.

As mensagens dentro de grupos diferentes não são consideradas. Se essas mensagens ocorrerem, elas serão processadas na ordem em que o primeiro segmento de cada mensagem ocorre na fila.

```
GMO.Options = MQGMO_SYNCPOINT | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_SEGMENTS_AVAILABLE | MQGMO_WAIT
do while ( SegmentStatus == MQSS_SEGMENT )
   MQGET
   /* Process each remaining segment of the logical message */
   ...

MQCMIT
```

Segmentação de mensagens lógicas do aplicativo
As mensagens devem ser mantidas em ordem lógica em um grupo e algumas ou todas elas podem ser tão grandes que requerem segmentação de aplicativos.

Em nosso exemplo, um grupo de quatro mensagens lógicas devem ter put efetuado. Todas, exceto a terceira mensagem, são grandes e precisam de segmentação, que é executada pelo aplicativo de put:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT
```

```
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_LAST_SEGMENT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_LAST_SEGMENT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP

MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_LAST_SEGMENT

MQCMIT
```

No aplicativo de get, MQGMO_ALL_MSGS_AVAILABLE é especificado no primeiro MQGET. Isso significa que nenhuma mensagem ou segmento de um grupo será recuperado até que todo o grupo esteja disponível. Quando a primeira mensagem física de um grupo tiver sido recuperada, MQGMO_LOGICAL_ORDER é usado para assegurar que os segmentos e as mensagens do grupo sejam recuperadas em ordem:

```
GMO.Options = MQGMO_SYNCPOINT | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_MSGS_AVAILABLE | MQGMO_WAIT

do while ( (GroupStatus   != MQGS_LAST_MSG_IN_GROUP) ||
           (SegmentStatus != MQGS_LAST_SEGMENT) )
   MQGET
   /* Process a segment or complete logical message.  Use the GroupStatus
      and SegmentStatus information to see what has been returned */
   ...

MQCMIT
```

Nota: Se você especificar MQGMO_LOGICAL_ORDER e houver um grupo atual, MQGMO_ALL_MSGS_AVAILABLE será ignorado.

Mensagens de referência

Use estas informações para saber mais sobre as mensagens de referência.

Nota: Não suportado no WebSphere MQ para z/OS..

Este método permite que um objeto grande seja transferido de um nó para outro sem armazenar o objeto nas filas do WebSphere MQ nos nós de origem ou de destino Isso é especialmente benéfico quando os dados existem em outro formato, por exemplo, para aplicativos de e-mail.

Para fazer isso, você especifica uma saída de mensagem em ambas as extremidades de um canal. Para obter informações sobre como fazer isso, consulte “Programas de saída de mensagem do canal” na página 417.

WebSphere MQ define o formato de um cabeçalho da mensagem de referência (MQRMH). Consulte MQRMH para obter uma descrição. Ele é reconhecido com um nome de formato definido e pode ser seguido pelos dados reais.

Para iniciar a transferência de um objeto grande, um aplicativo pode colocar uma mensagem que consiste em um cabeçalho de mensagem de referência sem dados após ele. Quando essa mensagem sai do nó, a saída de mensagem recupera o objeto de maneira apropriada e anexa-o à mensagem de referência. Em seguida, retorna a mensagem (agora maior do que antes) ao Agente do canal de mensagens de envio para transmissão ao MCA de recebimento.

Outra saída de mensagem é configurada no MCA de recebimento. Quando essa saída de mensagem recebe uma dessas mensagens, ela cria o objeto usando os dados do objeto que foram anexados e passa adiante a mensagem de referência *sem* isso. A mensagem de referência agora pode ser recebida por um aplicativo e esse aplicativo sabe que o objeto (ou pelo menos a parte dele representada por essa mensagem de referência) foi criado neste nó.

A quantia máxima de dados do objeto que uma saída de mensagem de envio pode anexar à mensagem de referência é limitada pelo comprimento máximo da mensagem negociado para o canal. A saída

pode retornar somente uma única mensagem para o MCA para cada mensagem passada, portanto, o aplicativo de put pode colocar várias mensagens para fazer com que um objeto seja transferido. Cada mensagem deve identificar o comprimento *lógico* e o deslocamento do objeto que deve ser anexado a ele. No entanto, nos casos em que não é possível saber o tamanho total do objeto ou o tamanho máximo permitido pelo canal, projetar a saída de mensagem de envio para que o aplicativo de put apenas coloque uma única mensagem e a saída em si coloque a próxima mensagem na fila de transmissão quando tiver anexado o quanto de dados puder à mensagem que foi passada a ela.

Antes de usar esse método de lidar com mensagens grandes, considere os pontos a seguir:

- O MCA e a saída de mensagem são executados sob um ID do usuário WebSphere MQ . A saída de mensagem (e, portanto, o ID do usuário) precisa acessar o objeto para recuperá-lo na extremidade de envio ou criá-lo na extremidade de recebimento; isso pode ser viável somente em casos em que o objeto está acessível de forma global. Isso levanta um problema de segurança.
- Se a mensagem de referência com dados em massa anexados a ela tiver que percorrer vários gerenciadores de filas antes de atingir seu destino, os dados em massa *estão* presentes nas filas do WebSphere MQ nos nós intervenientes. No entanto, nenhum suporte ou saídas especiais precisam ser fornecidos nesses casos.
- Projetar sua saída de mensagem é dificultado se for permitido novo roteamento ou enfileiramento de filas não entregues. Nesses casos, as partes do objeto podem chegar fora de ordem.
- Quando uma mensagem de referência chega ao seu destino, a saída de mensagem de recebimento cria o objeto. No entanto, isso não é sincronizado com a unidade de trabalho do MCA, portanto, se o lote for restaurado, outra mensagem de referência que contém essa mesma parte do objeto chegará em um lote posterior e a saída de mensagem pode tentar recriar a mesma parte do objeto. Se o objeto for, por exemplo, uma série de atualizações do banco de dados, isso pode ser inaceitável. Nesse caso, a saída de mensagem deve manter um log do qual as atualizações foram aplicadas; isso pode requerer o uso de uma fila do WebSphere MQ .
- Dependendo das características do tipo de objeto, as saídas de mensagem e os aplicativos podem precisar cooperar para manter contagens de uso, de forma que o objeto possa ser excluído quando não for mais necessário. Um identificador de instância também pode ser necessário; um campo é fornecido para isso no cabeçalho da mensagem de referência (consulte MQRMH.
- Se uma mensagem de referência for colocada como uma lista de distribuição, o objeto deverá ser recuperável para cada lista de distribuição ou destino individual resultante nesse nó. Pode ser necessário manter contagens de uso. Além disso, considere a possibilidade de que um nó pode ser o nó final para alguns dos destinos na lista, mas um nó intermediário para outros.
- Dados em massa não são geralmente convertidos. Isso ocorre porque a conversão ocorre *antes* que a saída de mensagem seja chamada. Por essa razão, a conversão não deve ser solicitada no canal emissor original. Se a mensagem de referência passar por um nó intermediário, os dados em massa serão convertidos quando enviados do nó intermediário, se solicitado.
- Mensagens de referência não podem ser segmentadas.

Usando as estruturas MQRMH e MQMD

Consulte MQRMH e MQMD para obter uma descrição dos campos no cabeçalho da mensagem de referência e no descritor de mensagens.

Na estrutura do MQMD, configure o campo *Format* para MQFMT_REF_MSG_HEADER. O formato MQHREF, quando solicitado em MQGET, é convertido automaticamente pelo WebSphere MQ juntamente com quaisquer dados em massa a seguir.

Aqui está um exemplo do uso dos campos *DataLogicalOffset* e *DataLogicalLength* de MQRMH:

Um aplicativo de put pode colocar uma mensagem de referência com:

- Nenhum dado físico
- *DataLogicalLength* = 0 (esta mensagem representa o objeto inteiro)
- *DataLogicalOffset* = 0.

Supondo que o objeto tenha 70.000 bytes de comprimento, a saída de mensagem de envio envia os primeiros 40.000 bytes juntamente com o canal em uma mensagem de referência que contém:

- 40.000 bytes de dados físicos após o MQRMH
- *DataLogicalLength* = 40000
- *DataLogicalOffset* = 0 (a partir do início do objeto).

Em seguida, coloca uma outra mensagem na fila de transmissão contendo:

- Nenhum dado físico
- *DataLogicalLength* = 0 (até o final do objeto). Você poderia especificar um valor de 30.000 aqui.
- *DataLogicalOffset* = 40000 (a partir deste ponto).

Quando essa saída de mensagem for vista pela saída de mensagem de envio, os 30.000 bytes de dados restantes são anexados e os campos são configurados para:

- 30.000 bytes de dados físicos após o MQRMH
- *DataLogicalLength* = 30000
- *DataLogicalOffset* = 40000 (a partir deste ponto).

A sinalização MQRMHF_LAST também é configurada.

Para obter uma descrição dos programas de amostra fornecidos para o uso de mensagens de referência, consulte “Programas de amostra para plataformas distribuídas” na página 98.

Esperando mensagens

Se desejar que um programa espere até que uma mensagem chegue em uma fila, especifique a opção MQGMO_WAIT no campo *Options* da estrutura MQGMO.

Use o campo *WaitInterval* da estrutura MQGMO para especificaro tempo máximo (em milissegundos) que deseja que uma chamada MQGET espere a chegada de uma mensagem em uma fila.

Se a mensagem não chegar dentro desse tempo, a chamada MQGET será concluída com o código de razão MQRC_NO_MSG_AVAILABLE.

É possível especificar um intervalo de espera ilimitado usando a constante MQWI_UNLIMITED no campo *WaitInterval*. No entanto, eventos fora de seu controle poderiam fazer seu programa esperar muito tempo, portanto, use essa constante com cuidado. Os aplicativos IMS não devem especificar um intervalo de espera ilimitado porque isso impediria a finalização do sistema IMS . (Quando o IMS é finalizado, ele requer que todas as regiões dependentes sejam encerradas.). Em vez disso, os aplicativos IMS podem especificar um intervalo de espera finito; então, se a chamada for concluída sem recuperar uma mensagem após esse intervalo, emita outra chamada MQGET com a opção de espera.

Nota: Se mais de um programa estiver esperando na mesma fila compartilhada para *remover* uma mensagem, somente um programa será ativado por uma mensagem que chega. No entanto, se mais de um programa estiver esperando para procurar uma mensagem, todos os programas poderão ser ativados. Para obter mais informações, consulte a descrição do campo *Options* da estrutura MQGMO em MQGMO.

Se o estado da fila ou do gerenciador de filas mudar antes que o intervalo de espera expire, ocorrem as ações a seguir:

- Se o gerenciador de filas entrar no estado quiesce e você usou a opção MQGMO_FAIL_IF_QUIESCING, a espera será cancelada e a chamada MQGET será concluída com o código de razão MQRC_Q_MGR_QUIESCING. Sem essa opção, a chamada permanece em espera.
- Se o gerenciador de filas for forçado a parar ou for cancelado, a chamada MQGET será concluída com o código de razão MQRC_Q_MGR_STOPPING ou MQRC_CONNECTION_BROKEN.
- Se os atributos da fila (ou uma fila para a qual o nome da fila é resolvido) forem mudados de forma que solicitações get agora são inibidas, a espera será cancelada e a chamada MQGET será concluída com o código de razão MQRC_GET_INHIBITED.

- Se os atributos da fila (ou uma fila para a qual o nome da fila é resolvido) forem mudados de tal forma que a opção FORCE seja necessária, a espera será cancelada e a chamada MQGET será concluída com o código de razão MQRC_OBJECT_CHANGED.

Para obter mais informações sobre as circunstâncias nas quais essas ações ocorrem, consulte MQGMO

Ignorando restauração

É possível evitar que um programa de aplicativo entre em um loop *MQGET-error-backout* especificando a opção **MQGMO_MARK_SKIP_BACKOUT** na chamada MQGET.

Nota: Suportado apenas no WebSphere MQ para o z/OS

Como parte de uma unidade de trabalho, um programa de aplicativo pode emitir uma ou mais chamadas MQGET para obter mensagens de uma fila. Se o programa de aplicativo detectar um erro, ele pode restaurar a unidade de trabalho. Isso restaura todos os recursos atualizados durante essa unidade de trabalho para o estado em que estavam antes da unidade de trabalho ser iniciada e restabelece as mensagens recuperadas pelas chamadas MQGET.

Após serem restabelecidas, essas mensagens estarão disponíveis para chamadas MQGET subsequentes emitidas pelo programa de aplicativo. Em muitos casos, isso não causa um problema para o programa de aplicativo. No entanto, em casos em que o erro que leva à restauração não pode ser contornado, ter a mensagem restabelecida na fila poderá fazer o programa de aplicativo entrar em um loop *MQGET-error-backout*.

Para evitar esse problema, especifique a opção MQGMO_MARK_SKIP_BACKOUT na chamada MQGET. Isso marca a solicitação MQGET como não estando envolvida na restauração iniciada pelo aplicativo; ou seja, ela não deve ser restaurada. O uso dessa opção significa que quando uma restauração ocorrer, atualizações em outros recursos são recuperadas conforme necessário, mas a mensagem marcada será tratada como se tivesse sido recuperada sob uma nova unidade de trabalho.

O programa de aplicativo deve emitir uma chamada WebSphere MQ para confirmar a nova unidade de trabalho ou para voltar a nova unidade de trabalho. Por exemplo, o programa pode executar a manipulação de exceção, como informar o originador que a mensagem foi descartada e confirmar a unidade de trabalho removendo a mensagem da fila. Se a nova unidade de trabalho for restaurada (por qualquer razão) a mensagem será restabelecida na fila.

Dentro de uma unidade de trabalho, pode haver apenas uma solicitação MQGET marcada para ignorar restauração; no entanto, pode haver várias outras mensagens que não estão marcadas para ignorar restauração. Após uma mensagem ser marcada para ignorar restauração, quaisquer chamadas MQGET adicionais dentro da unidade de trabalho que especifiquem MQGMO_MARK_SKIP_BACKOUT falharão com o código de razão MQRC_SECOND_MARK_NOT_ALLOWED.

Nota:

1. A mensagem marcada ignora a restauração somente se a unidade de trabalho que a contém for finalizada por uma solicitação de aplicativo para restaurá-la. Se a unidade de trabalho for restaurada por qualquer outra razão, a mensagem será restaurada na fila da mesma maneira que seria se não estivesse marcada para ignorar a restauração.
2. Ignorar a restauração não é suportado nos procedimentos armazenados do DB2 que participam de unidades de trabalho controladas pelo RRS Por exemplo, uma chamada MQGET com a opção MQGMO_MARK_SKIP_BACKOUT falhará com o código de razão MQRC_OPTION_ENVIRONMENT_ERROR.

Figura 36 na página 273 ilustra uma sequência típica de etapas que um programa de aplicativo pode conter quando uma solicitação MQGET é necessária para ignorar a restauração.

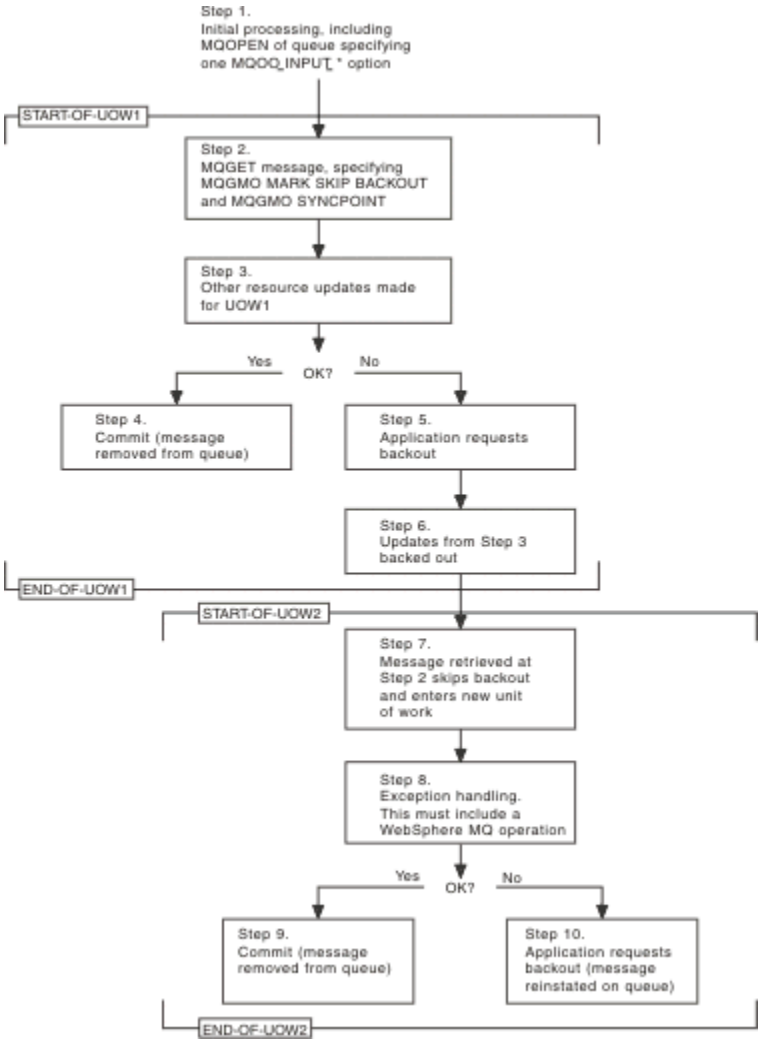


Figura 36. Ignorando a restauração usando MQGMO_MARK_SKIP_BACKOUT

As etapas em Figura 36 na página 273 são:

Etapa 1
O processamento inicial ocorre dentro da transação, incluindo uma chamada MQOPEN para abrir a fila (especificando uma das opções MQOO_INPUT_* para obter mensagens da fila na Etapa 2).

Etapa 2
MQGET é chamado, com MQGMO_SYNCPOINT e MQGMO_MARK_SKIP_BACKOUT. MQGMO_SYNCPOINT é necessário porque MQGET deve estar em uma unidade de trabalho para MQGMO_MARK_SKIP_BACKOUT ser efetivo. Em Figura 36 na página 273, essa unidade de trabalho é referida como UOW1.

Etapa 3
Outras atualizações de recursos são feitas como parte de UOW1. Elas podem incluir outras chamadas MQGET (emitidas sem MQGMO_MARK_SKIP_BACKOUT).

Etapa 4
Todas as atualizações das Etapas 2 e 3 concluídas conforme necessário. O programa de aplicativo confirma as atualizações e a UOW1 é finalizada. A mensagem recuperada na Etapa 2 é removida da fila.

Etapa 5
Algumas das atualizações das Etapas 2 e 3 não são concluídas conforme necessário. O programa de aplicativo solicita que as atualizações feitas durante essas etapas sejam restauradas.

Etapa 6
As atualizações feitas na Etapa 3 são restauradas.

Etapa 7
A solicitação MQGET feita na Etapa 2 ignora a restauração e se torna parte de uma nova unidade de trabalho, UOW2.

Etapa 8
A UOW2 executa a manipulação de exceção em resposta à restauração da UOW1. (Por exemplo, uma chamada MQPUT para outra fila, indicando que ocorreu um problema que causou a restauração da UOW1.)

Etapa 9
A Etapa 8 é concluída conforme necessário, o programa de aplicativo confirma a atividade e a UOW2 é finalizada. Como a solicitação MQGET faz parte da UOW2 (consulte a Etapa 7), essa confirmação faz com que a mensagem seja removida da fila.

Etapa 10
A Etapa 8 não é concluída conforme necessário e o programa de aplicativo restaura a UOW2. Como a solicitação get message faz parte da UOW2 (consulte a Etapa 7), ela também é restaurada e restabelecida na fila. Agora, esta está disponível para novas chamadas MQGET emitidas por este ou outro programa de aplicativo (da mesma maneira que qualquer outra mensagem na fila).

Conversão de Dados do Aplicativo

Quando necessário, MCAs convertem o descritor de mensagens e os dados de cabeçalho no conjunto de caracteres e na codificação necessários. Qualquer uma das extremidades do link (ou seja, o MCA local ou o MCA remoto) pode fazer a conversão.

Quando um aplicativo coloca mensagens em uma fila, o gerenciador de filas local inclui informações de controle nos descritores de mensagens para facilitar o controle das mensagens quando elas são processadas pelos gerenciadores de filas e MCAs. Dependendo do ambiente, os campos de dados do cabeçalho da mensagem são criados no conjunto de caracteres e na codificação do sistema local.

Ao mover mensagens entre sistemas, você, às vezes, precisa converter os dados do aplicativo no conjunto de caracteres e na codificação requeridos pelo sistema de recebimento. Isso pode ser feito a partir de programas de aplicativos no sistema de recebimento ou pelos MCAs no sistema de envio. Se a conversão de dados for suportada no sistema de recebimento, use os programas de aplicativo para converter os dados do aplicativo, em vez de depender de a conversão já ter ocorrido no sistema de envio.

Os dados do aplicativo são convertidos em um programa de aplicativo quando você especifica a opção MQGMO_CONVERT no campo *Options* da estrutura MQGMO transmitida a uma chamada MQGET e *todas* as opções a seguir são verdadeiras:

- Os campos *CodedCharSetId* ou *Encoding* configurados na estrutura MQMD associada à mensagem na fila são diferentes dos campos *CodedCharSetId* ou *Encoding* configurados na estrutura MQMD especificada na chamada MQGET.
- O campo *Format* na estrutura MQMD associada à mensagem não for MQFMT_NONE.
- O *BufferLength* especificado na chamada MQGET não for zero.
- O comprimento dos dados da mensagem não for zero.
- O gerenciador de filas suporta a conversão entre os campos *CodedCharSetId* e *Encoding* especificados nas estruturas MQMD associadas à mensagem e à chamada MQGET. Consulte CodedCharSetId e Codificação para obter detalhes sobre os identificadores do conjunto de caracteres codificados e as codificações de máquina suportados.
- O gerenciador de filas suporta a conversão do formato da mensagem. Se o campo *Format* da estrutura MQMD associada à mensagem for um dos formatos integrados, o gerenciador de filas poderá converter a mensagem. Se o *Format* não for um dos formatos integrados, será necessário gravar uma saída de conversão de dados para converter a mensagem.

Se o MCA de envio for para converter os dados, especifique a palavra-chave CONVERT(YES) na definição de cada emissor ou canal do servidor para o qual a conversão é necessária. Se a conversão de dados falhar, a mensagem será enviada para o DLQ no gerenciador de filas de envio e o campo *Feedback* da estrutura MQDLH indicará a razão. Se a mensagem não puder ser colocada no DLQ, o canal será fechado

e a mensagem não convertida permanecerá na fila de transmissão. A conversão de dados nos aplicativos em vez de nos MCAs de envio evita esta situação.

Como regra, os dados na mensagem que são descritos como *dados de caracteres* pelo formato integrado ou saída de conversão de dados são convertidos do conjunto de caracteres codificados usado pela mensagem para esse solicitado e os campos *numéricos* são convertidos para a codificação solicitada.

Para obter detalhes adicionais das convenções de processamento de conversão usadas ao converter os formatos integrados e para obter informações sobre como gravar suas próprias saídas de conversão de dados, consulte "Escrevendo saídas de conversão de dados" na página 421. Consulte também Idiomas nacionais e Codificações da máquina para obter informações sobre as tabelas de suporte ao idioma e sobre as codificações de máquina suportadas.

Conversão de caracteres de nova linha EBCDIC

Se precisar assegurar que os dados que você envia de uma plataforma EBCDIC para um ASCII são idênticos aos dados que você recebe de volta, deve-se controlar a conversão de caracteres de nova linha EBCDIC.

É possível fazer isso usando um comutador dependente de plataforma que força o WebSphere MQ a usar as tabelas de conversão não modificadas, mas você deve estar ciente do comportamento inconsistente que pode resultar.

O problema surge porque o caractere de nova linha EBCDIC não é convertido de forma consistente em plataformas ou tabelas de conversão. Como resultado, se os dados forem exibidos em uma plataforma ASCII, a formatação pode estar incorreta. Isso dificultaria, por exemplo, administrar um sistema IBM i remotamente a partir de uma plataforma ASCII usando RUNMQSC.

Consulte Conversão de dados para obter informações adicionais sobre como converter dados de formato EBCDIC no formato ASCII.

Procurando mensagens em uma fila

Use estas informações para descobrir sobre como procurar mensagens em uma fila usando a chamada MQGET.

Para usar a chamada MQGET para procurar as mensagens em uma fila:

1. Chame o MQOPEN para abrir a fila para navegar, especificando a opção MQOO_BROWSE.
2. Para navegar para a primeira mensagem na fila, chame MQGET com a opção MQGMO_BROWSE_FIRST. Para localizar a mensagem que você deseja, chame MQGET repetidamente com a opção MQGMO_BROWSE_NEXT para percorrer várias mensagens.

   Deve-se configurar os campos *MsgId* e *CorrelId* da estrutura MQMD para nulo após cada chamada MQGET para ver todas as mensagens.
3. Chame MQCLOSE para fechar a fila.

O cursor de navegação

Ao abrir (MQOPEN) uma fila para procura, a chamada estabelece um cursor de procura para ser usado com chamadas MQGET que usem uma das opções de procura. É possível considerar o cursor de procura como um ponteiro lógico posicionado antes da primeira mensagem na fila.

É possível haver mais de um cursor de procura ativo (a partir de um único programa) emitindo várias solicitações MQOPEN para a mesma fila.

Ao chamar MQGET para procurar, use uma das opções a seguir em sua estrutura MQGMO:

MQGMO_BROWSE_FIRST
Obtém uma cópia da primeira mensagem que satisfaz as condições especificadas em sua estrutura MQMD.

MQGMO_BROWSE_NEXT
Obtém uma cópia da próxima mensagem que satisfaz as condições especificadas em sua estrutura MQMD.

MQGMO_BROWSE_MSG_UNDER_CURSOR
Obtém uma cópia da mensagem atualmente apontada pelo cursor, ou seja, aquela que foi recuperada por último usando a opção MQGMO_BROWSE_FIRST ou MQGMO_BROWSE_NEXT.

Em todos os casos, a mensagem permanece na fila.

Ao abrir uma fila, o cursor de procura é posicionado logicamente antes da primeira mensagem na fila. Isso significa que se você fizer sua chamada MQGET imediatamente após a sua chamada MQOPEN, será possível usar a opção MQGMO_BROWSE_NEXT para procurar a primeira mensagem; não é necessário usar a opção MQGMO_BROWSE_FIRST.

A ordem na qual as mensagens são copiadas da fila é determinada pelo atributo *MsgDeliverySequence* da fila. (Para obter mais informações, consulte "A ordem em que as mensagens são recuperadas de uma fila" na página 248.)

- "Filas na sequência FIFO (primeira a entrar, primeira a sair)" na página 276
- "Filas em sequência de prioridade" na página 276
- "Mensagens não confirmadas" na página 276
- "Mudança na sequência da fila" na página 277
- "Usando o índice da fila" na página 277

Filas na sequência FIFO (primeira a entrar, primeira a sair)

A primeira mensagem em uma fila nessa sequência é a mensagem que está na fila há mais tempo.

Use MQGMO_BROWSE_NEXT para ler as mensagens sequencialmente na fila. Você verá quaisquer mensagens colocadas na fila enquanto estiver procurando, pois uma fila nessa sequência tem mensagens colocadas no final. Quando o cursor reconhece que alcançou o fim da fila, o cursor de procura permanece onde está e retorna com MQRC_NO_MSG_AVAILABLE. É possível então deixá-lo lá esperando mensagens adicionais ou reconfigurá-lo para o início da fila com uma chamada MQGMO_BROWSE_FIRST.

Filas em sequência de prioridade

A primeira mensagem em uma fila nessa sequência é a mensagem que está na fila há mais tempo e que tem a prioridade mais alta no momento em que a chamada MQOPEN é emitida.

Use MQGMO_BROWSE_NEXT para ler as mensagens na fila.

O cursor de procura aponta para a próxima mensagem, trabalhando a partir da prioridade da primeira mensagem para terminar com a mensagem com a prioridade mais baixa. Ele procura quaisquer mensagens colocadas na fila durante esse tempo, contanto que elas tenham prioridade igual a ou menor que a mensagem identificada pelo cursor de procura atual.

Quaisquer mensagens colocadas na fila com prioridade mais alta poderão ser procuradas somente da seguinte forma:

- Abrindo a fila para procurar novamente, quando um novo cursor de procura é estabelecido
- Usando a opção MQGMO_BROWSE_FIRST

Mensagens não confirmadas

Uma mensagem não confirmada nunca está visível para uma procura; o cursor de procura a ignora.

As mensagens dentro de uma unidade de trabalho não podem ser procuradas até a unidade de trabalho ser confirmada. As mensagens não mudam sua posição na fila quando confirmadas, portanto, mensagens não confirmadas ignoradas não serão vistas, mesmo quando *estiverem* confirmadas, a menos que você use a opção MQGMO_BROWSE_FIRST e trabalha ao longo da fila novamente.

Mudança na sequência da fila

Se a sequência de entrega de mensagem for mudada de prioridade para FIFO enquanto houver mensagens na fila, a ordem das mensagens que já estão na fila não será mudada. As mensagens incluídas na fila posteriormente, usam a prioridade padrão da fila.

Usando o índice da fila

Ao procurar em uma fila indexada que contenha somente mensagens de uma única prioridade (persistentes, não persistentes ou ambas), o gerenciador de filas usa o índice para procurar quando determinadas formas de procura são usadas.

Nota: Suportado apenas no WebSphere MQ para o z/OS

Qualquer uma das formas de procura a seguir é usada quando uma fila indexada contém somente mensagens de prioridade única:

1. Se a fila estiver indexada por MSGID, as solicitações de procura que passam um MSGID na estrutura MQMD são processadas usando o índice para localizar a mensagem de destino.
2. Se a fila for indexada por CORRELID, solicitações de procura que passam um CORRELID na estrutura MQMD são processadas usando o índice para localizar a mensagem de destino.
3. Se a fila for indexada por GROUPID, solicitações de procura que passam um GROUPID na estrutura MQMD são processadas usando o índice para localizar a mensagem de destino.

Se a solicitação de procura não passar um MSGID, CORRELID ou GROUPID na estrutura MQMD, a fila será indexada e uma mensagem será retornada, a entrada de índice para a mensagem deve ser localizada e as informações da mesma usadas para atualizar o cursor de procura. Se você usar uma ampla seleção de valores de índice, isso não inclui processamento extra significativo à solicitação de procura.

Procurando mensagens quando o comprimento da mensagem é desconhecido

Para procurar uma mensagem quando não souber o tamanho da mensagem e se não desejar usar os campos *MsgId*, *CorrelId* ou *GroupId* para localizar a mensagem, será possível usar a opção MQGMO_BROWSE_MSG_UNDER_CURSOR:

1. Emita uma chamada MQGET com:
   - A opção MQGMO_BROWSE_FIRST ou MQGMO_BROWSE_NEXT
   - A opção MQGMO_ACCEPT_TRUNCATED_MSG
   - Buffer de comprimento zero

   Nota: Se outro programa provavelmente obtiver a mesma mensagem, considere usar a opção MQGMO_LOCK também. MQRC_TRUNCATED_MSG_ACCEPTED deve ser retornado.
2. Use *DataLength* retornado para alocar o armazenamento necessário.
3. Emita uma chamada MQGET com a MQGMO_BROWSE_MSG_UNDER_CURSOR.

A mensagem apontada é a última que foi recuperada; o cursor não terá se movido. É possível optar por bloquear a mensagem usando a opção MQGMO_LOCK ou desbloquear uma mensagem bloqueada usando a opção MQGMO_UNLOCK.

A chamada falha se nenhuma MQGET com as opções MQGMO_BROWSE_FIRST ou MQGMO_BROWSE_NEXT tiver sido emitida com sucesso desde a abertura da fila.

Removendo uma mensagem que você procurou

É possível remover da fila uma mensagem que já procurou desde que tenha aberto a fila para remover mensagens, assim como para procura. (Deve-se especificar uma das opções MQOO_INPUT_*, assim como a opção MQOO_BROWSE, em sua chamada MQOPEN.)

Para remover a mensagem, chame MQGET novamente, mas no campo *Options* da estrutura MQGMO, especifique MQGMO_MSG_UNDER_CURSOR.Neste caso, a chamada MQGET ignora os campos *MsgId*, *CorrelId*e *GroupId* da estrutura MQMD.

No tempo entre suas etapas de procura e remoção, outro programa pode ter removido mensagens da fila, inclusive a mensagem sob seu cursor de procura. Nesse caso, sua chamada MQGET retorna um código de razão para informar que a mensagem não está disponível.

Procurando mensagens em ordem lógica

"Ordenação lógica e física" na página 249 explica a diferença entre a ordem lógica e física de mensagens em uma fila. Essa distinção é importante principalmente ao procurar em uma fila, porque, em geral, as mensagens não estão sendo excluídas e operações de procura não começam necessariamente no início da fila.

Se um aplicativo procurar nas diversas mensagens de um grupo (usando ordem lógica), é importante que a ordem lógica deve ser seguida para chegar ao início do próximo grupo, porque a última mensagem de um grupo pode ocorrer fisicamente *após* a primeira mensagem do próximo grupo. A opção MQGMO_LOGICAL_ORDER assegura que a ordem lógica seja seguida ao fazer a varredura de uma fila.

Use MQGMO_ALL_MSGS_AVAILABLE (ou MQGMO_ALL_SEGMENTS_AVAILABLE) com cuidado para operações de procura. Considere o caso de mensagens lógicas com MQGMO_ALL_MSGS_AVAILABLE. O efeito disso é que uma mensagem lógica estar disponível somente se todas as mensagens restantes no grupo também estiverem presentes. Se não estiverem, a mensagem será saltada. Isso pode significar que quando as mensagens ausentes chegarem subsequentemente, elas não serão observadas por uma operação browse-next.

Por exemplo, se as mensagens lógicas a seguir estiverem presentes,

```
Logical message 1 (not last) of group 123
Logical message 1 (not last) of group 456
Logical message 2 (last)     of group 456
```

e uma função de procura for emitido com MQGMO_ALL_MSGS_AVAILABLE, a primeira mensagem lógica do grupo 456 será retornada primeiro, deixando o cursor de procura nessa mensagem lógica. Se a segunda (última) mensagem do grupo 123 chegar agora:

```
Logical message 1 (not last) of group 123
Logical message 2 (last)     of group 123
Logical message 1 (not last) of group 456  <=== browse cursor
Logical message 2 (last)     of group 456
```

e a mesma função browse-next for emitida, não será percebido que o grupo 123 agora está completo, pois a primeira mensagem desse grupo está *antes* do cursor de procura.

Em alguns casos (por exemplo, se mensagens forem recuperadas destrutivamente quando o grupo estiver presente em sua totalidade), é possível usar MQGMO_ALL_MSGS_AVAILABLE juntamente com MQGMO_BROWSE_FIRST. Caso contrário, deve-se repetir a varredura de procura para observar as novas mensagens que chegaram que passaram despercebidas; simplesmente emitindo MQGMO_WAIT junto com MQGMO_BROWSE_NEXT e MQGMO_ALL_MSGS_AVAILABLE não as leva em consideração. (Isso também acontece com mensagens de prioridade mais alta que podem chegar após a varredura de mensagens ser concluída.)

As próximas seções verificam exemplos de procura que lidam com mensagens não segmentadas; as mensagens segmentadas seguem princípios semelhantes.

Procurando mensagens em grupos
Neste exemplo, o aplicativo procura em cada mensagem na fila, em ordem lógica.

Mensagens na fila podem estar agrupadas. Para mensagens agrupadas, o aplicativo não deseja iniciar o processamento de qualquer grupo até que todas as mensagens dele tenham chegado. MQGMO_ALL_MSGS_AVAILABLE, portanto, é especificado para a primeira mensagem no grupo; para mensagens subsequentes no grupo, essa opção é desnecessária.

MQGMO_WAIT é usado neste exemplo. No entanto, embora a espera possa ser satisfeita se um novo grupo chegar, pelas razões em "Procurando mensagens em ordem lógica" na página 278, ela não será satisfeita e o cursor de procura já tiver passado a primeira mensagem lógica de um grupo e as mensagens

restantes chegarem agora. No entanto, esperar um intervalo adequado assegura que o aplicativo não entre em loop constantemente enquanto espera novas mensagens ou segmentos.

MQGMO_LOGICAL_ORDER é usado de forma geral para assegurar que a varredura esteja em ordem lógica. Isso contrasta com o exemplo MQGET destrutivo, em que como cada grupo está sendo removido, MQGMO_LOGICAL_ORDER não é usado ao procurar a primeira (ou única) mensagem em um grupo.

Supõe-se que o buffer do aplicativo sempre seja grande o suficiente para conter a mensagem inteira, independentemente de se a mensagem foi segmentada ou não. Portanto, MQGMO_COMPLETE_MSG é especificado em cada MQGET.

Segue um exemplo de como procurar mensagens lógicas em um grupo:

```
/* Browse the first message in a group, or a message not in a group */
GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_MSGS_AVAILABLE | MQGMO_WAIT
MQGET GMO.MatchOptions = MQMO_MATCH_MSG_SEQ_NUMBER, MD.MsgSeqNumber = 1
/* Examine first or only message */
 ...

GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Examine each remaining message in the group */
   ...
```

O grupo é repetido até MQRC_NO_MSG_AVAILABLE ser retornado.

Procurando e recuperando de forma destrutiva
Neste exemplo, o aplicativo procura cada uma das mensagens lógicas dentro de um grupo, antes de decidir se recupera esse grupo de forma destrutiva.

A primeira parte deste exemplo é semelhante à anterior. No entanto, nesse caso, tendo navegado em um grupo inteiro, decidimos voltar e recuperá-lo de forma destrutiva.

Conforme cada grupo é removido neste exemplo, MQGMO_LOGICAL_ORDER não é usado ao procurar a primeira ou única mensagem em um grupo.

Segue um exemplo de procura e recuperação de forma destrutiva:

```
GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_MESSAGES_AVAILABE | MQGMO_WAIT
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Examine each remaining message in the group (or as many as
      necessary to decide whether to get it destructively) */
   ...

if ( we want to retrieve the group destructively )

   if ( GroupStatus == ' ' )
      /* We retrieved an ungrouped message */
      GMO.Options = MQGMO_MSG_UNDER_CURSOR | MQGMO_SYNCPOINT
      MQGET GMO.MatchOptions = 0
      /* Process the message */
      ...

   else
      /* We retrieved one or more messages in a group.  The browse cursor */
      /* will not normally be still on the first in the group, so we have */
      /* to match on the GroupId and MsgSeqNumber = 1.                    */
      /* Another way, which works for both grouped and ungrouped messages,*/
      /* would be to remember the MsgId of the first message when it was  */
      /* browsed, and match on that.                                      */
      GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT
      MQGET GMO.MatchOptions = MQMO_MATCH_GROUP_ID
                             | MQMO_MATCH_MSG_SEQ_NUMBER,
           (MQMD.GroupId      = value already in the MD)
            MQMD.MsgSeqNumber = 1
      /* Process first or only message */
      ...

      GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT
```

```
                | MQGMO_LOGICAL_ORDER
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Process each remaining message in the group */
   ...
```

Evitando entrega repetida de mensagens procuradas

Usando determinadas opções open e opções get-message, é possível marcar mensagens como tendo sido procuradas para que não sejam recuperadas novamente pelo aplicativo atual ou outros de cooperação. As mensagens podem ser desmarcadas explícita ou automaticamente para torná-las disponíveis novamente para procura.

Se você procurar mensagens em uma fila, poderá recuperá-las em uma ordem diferente da ordem na qual as recuperaria tivesse obtido as mesmas destrutivamente. Especificamente, é possível procurar a mesma mensagem diversas vezes, o que não será possível se ela for removida da fila. Para evitar isso, é possível *marcar* mensagens conforme são procuradas e evitar recuperar mensagens marcadas. Isso às vezes é referido como *procurar com marca*. Para marcar mensagens procuradas, use a opção get message MQGMO_MARK_BROWSE_HANDLE e, para recuperar somente mensagens não marcadas, use MQGMO_UNMARKED_BROWSE_MSG. Se usar uma combinação de opções MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG e MQGMO_MARK_BROWSE_HANDLE e emitir repetidos MQGETs, irá recuperar cada mensagem na fila da vez. Isso evita entrega repetida de mensagens, embora, MQGMO_BROWSE_FIRST seja usado para assegurar que as mensagens não sejam ignoradas. Essa combinação de opções pode ser representada pela constante única MQGMO_BROWSE_HANDLE. Quando não há mensagens na fila que não foram procuradas, MQRC_NO_MSG_AVAILABLE será retornado.

Se vários aplicativos estiverem procurando na mesma fila, eles podem abrir a fila com as opções MQOO_CO_OP e MQOO_BROWSE. A manipulação de objetos retornada por cada chamada MQOPEN é considerada como parte de um grupo de cooperação. Qualquer mensagem retornada por uma chamada MQGET especificando a opção MQGMO_MARK_BROWSE_CO_OP é considerada como marcada para esse conjunto de identificadores de cooperação.

Se uma mensagem tiver sido marcada por algum tempo, ela poderá ser automaticamente desmarcada pelo gerenciador de filas e disponibilizada para procura novamente. O atributo MsgMarkBrowseInterval do gerenciador de filas fornece o tempo em milissegundos que uma mensagem deve permanecer marcada para o conjunto de identificadores de cooperação. Um MsgMarkBrowseInterval igual a -1 significa que as mensagens nunca são automaticamente desmarcadas.

Quando o processo único ou o conjunto de processos de cooperação de marcação de mensagens para, quaisquer mensagens marcadas serão desmarcadas.

Exemplos de procura cooperativa

Você pode executar várias cópias de um aplicativo dispatcher para procurar mensagens em uma fila e iniciar um consumidor com base no conteúdo de cada mensagem. Em cada dispatcher, abra a fila com MQOO_CO_OP. Isso indica que os dispatchers estão cooperando e estarão cientes das mensagens marcadas uns dos outros. Cada dispatcher faz, então, repetidas chamadas MQGET, especificando as opções MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG e MQGMO_MARK_BROWSE_CO_OP (é possível usar a constante única MQGMO_BROWSE_CO_OP para representar essa combinação de opções). Cada aplicativo dispatcher recupera então somente as mensagens que ainda não foram marcadas por outros dispatchers em cooperação. O dispatcher inicializa um consumidor e passa o MsgToken retornado por MQGET para o consumidor, que destrutivamente obtém a mensagem da fila. Se o consumidor restaurar MQGET da mensagem, então, a mensagem estará disponível para um dos navegadores para novo dispatch, porque ela não está mais marcada. Se o consumidor não executar um MQGET na mensagem, então, após o MsgMarkBrowseInterval ter passado, o gerenciador de filas desmarca a mensagem para o conjunto de identificadores em cooperação e poderá ser despachada novamente.

Em vez de várias cópias do mesmo aplicativo dispatcher, você pode ter diversos aplicativos dispatcher diferentes procurando na fila, cada um apropriado para processar um subconjunto de mensagens na fila. Em cada dispatcher, abra a fila com MQOO_CO_OP. Isso indica que os dispatchers estão cooperando e estarão cientes das mensagens marcadas uns dos outros.

- Se a ordem de processamento de mensagens de um único dispatcher for importante, cada dispatcher faz chamadas MQGET repetidas, especificando as opções MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG e MQGMO_MARK_BROWSE_HANDLE (ou MQGMO_BROWSE_HANDLE). Se a mensagem procurada for adequada para esse dispatcher processar, ele então faz uma chamada MQGET especificando MQMO_MATCH_MSG_TOKEN, MQGMO_MARK_BROWSE_CO_OP e o MsgToken retornado pela chamada MQGET anterior. Se a chamada for bem-sucedida, o dispatcher inicializa o consumidor, passando o MsgToken para ele.
- Se a ordem de processamento de mensagens não for importante e o for esperado que dispatcher processe a maioria das mensagens que encontrar, use as opções MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG e MQGMO_MARK_BROWSE_CO_OP (ou MQGMO_BROWSE_CO_OP). Se o dispatcher procurar uma mensagem que ele não pode processar, ele desmarca a mensagem chamando MQGET com a opção MQMO_MATCH_MSG_TOKEN, MQGMO_UNMARK_BROWSE_CO_OP e o MsgToken retornado anteriormente.

Alguns casos em que a chamada MQGET falha

Se determinados atributos de uma fila forem alterados usando a opção FORCE em um comando entre a emissão de um MQOPEN e uma chamada MQGET, a chamada MQGET falha e retorna o código de razão MQRC_OBJECT_CHANGED.

O gerenciador de filas marca a manipulação de objetos como não sendo mais válida. Isso também acontece se as mudanças se aplicarem a qualquer fila para a qual o nome da fila é resolvido. Os atributos que afetam a manipulação dessa forma são listados na descrição da chamada MQOPEN no MQOPEN. Se a sua chamada retornar o código de razão MQRC_OBJECT_CHANGED, feche a fila, reabra-a e, em seguida, tente obter uma mensagem novamente.

Se as operações get forem inibidas para uma fila a partir da qual você está tentando obter mensagens (ou qualquer fila para a qual o nome da fila é resolvido), a chamada MQGET falhará e retornará o código de razão MQRC_GET_INHIBITED. Isso acontece mesmo se você estiver usando a chamada MQGET para navegação. É possível obter uma mensagem com êxito se você tentar a chamada MQGET posteriormente, se o design do aplicativo for tal que outros programas mudarem os atributos de filas regularmente.

Se uma fila dinâmica (temporária ou permanente) foi excluída, chamadas MQGET que usam uma manipulação de objetos adquirida anteriormente falham e retornam o código de razão MQRC_Q_DELETED.

Escrevendo aplicativos de publicar/assinar

Inicie a gravação de aplicativos de publicação / assinatura do WebSphere MQ

Para obter uma visão geral dos conceitos de publicação / assinatura, consulte Introdução ao WebSphere MQ sistema de mensagens de publicação / assinatura.

Consulte os tópicos a seguir para obter informações sobre como escrever diferentes tipos de aplicativos de publicar/assinar:

- “Gravando aplicativos de publicador” na página 282
- “Escrevendo aplicativos de assinante” na página 289
- “Ciclos de vida de publicar/assinar” na página 307
- “Propriedades de mensagem de publicação/assinatura” na página 312
- “Ordenação de mensagens” na página 314
- “Interceptando publicações” na página 314
- “Opções de publicação” na página 322
- “Opções de Assinatura” na página 322

Conceitos relacionados

“Conceitos de desenvolvimento de aplicativos” na página 8

É possível usar uma opção de linguagens orientadas a objeto ou processuais para gravar aplicativos IBM WebSphere MQ. Use os links neste tópico para obter informações sobre conceitos do IBM WebSphere MQ que são úteis para desenvolvedores de aplicativos.

"Decidindo qual linguagem de programação usar" na página 80
Use estas informações para descobrir sobre linguagens de programação e estruturas suportadas pelo IBM WebSphere MQe algumas considerações para usá-las.

"Projetando aplicativos IBM WebSphere MQ" na página 91
Quando você tiver decidido como seus aplicativos podem aproveitar as plataformas e ambientes disponíveis para você, será necessário decidir como usar os recursos oferecidos pelo WebSphere MQ.

"Programas de amostra do WebSphere MQ" na página 98
Use esta coleção de tópicos para aprender sobre programas de amostra do WebSphere MQ em diferentes plataformas

"Gravando um Aplicativo de Enfileiramento" na página 197
Use essas informações para aprender como escrever aplicativos de enfileiramento, conectar e desconectar de um gerenciador de filas, publicar/assinar e abrir e fechar objetos.

"Gravando aplicativos clientes" na página 356
O que você precisa saber para gravar os aplicativos clientes no WebSphere MQ

"Usando serviços da Web no WebSphere MQ" na página 958
É possível desenvolver aplicativos IBM WebSphere MQ para serviços da web usando o transporte IBM WebSphere MQ para SOAP ou a ponte IBM WebSphere MQ para HTTP (Protocolo de Transporte de Hipertexto)

"Construindo um aplicativo IBM WebSphere MQ" na página 435
Use estas informações para saber como construir um aplicativo IBM WebSphere MQ em diferentes plataformas.

"Manipulando erros do programa" na página 555
Estas informações explicarão os erros associados às chamadas MQI de aplicativos, ao realizar uma chamada ou quando sua mensagem for entregue para seu destino final.

Gravando aplicativos de publicador

Comece a escrever aplicativos do publicador estudando dois exemplos. O primeiro é modelado o mais próximo possível em um aplicativo ponto a ponto colocando mensagens em uma fila e o segundo demonstra a criação de tópicos dinamicamente - um padrão mais comum para aplicativos publicadores.

Gravar um aplicativo publicador simples do WebSphere MQ é como gravar um aplicativo de ponto a ponto do WebSphere MQ que coloca mensagens em uma fila (Tabela 41 na página 282). A diferença é você MQPUT mensagens para um tópico, não para uma fila.

Tabela 41. Padrão do programa WebSphere MQ ponto a ponto versus publicação / assinatura.

Salto	**Ponto a ponto de Chamada MQ**	**Publicação Chamada MQ**
Conecta-se a um gerenciador de filas	MQCONN	MQCONN
Abrir fila	MQOPEN	
Abra tópico		MQOPEN
Coloca mensagem(ns)	MQPUT	MQPUT
tópico Fechar		MQCLOSE
Fechar fila	MQCLOSE	
Desconecte a partir do gerenciador de filas	MQDISC	MQDISC

Para concretizar isso, há dois exemplos de aplicativos para publicar preços de ações. No primeiro exemplo (“Exemplo 1: publicador em um tópico fixo” na página 283), que é modelado de perto colocando mensagens em uma fila, o administrador cria uma definição de tópico de maneira semelhante à criação de uma fila. O programador codifica MQPUT para gravar mensagens no tópico, em vez de gravá-las em uma fila. No segundo exemplo (“Exemplo 2: publicador em um tópico de variável” na página 286), o padrão de interação do programa com o WebSphere MQ é semelhante. A diferença é que o programador fornece o tópico ao qual a mensagem é gravada, em vez do administrador. Na prática, isto geralmente significa que a sequência de tópicos é conteúdo definido ou fornecido por outra origem, como entrada humana através de um navegador.

Conceitos relacionados

“Escrevendo aplicativos de assinante” na página 289
Comece a escrever aplicativos de assinante estudando três exemplos: um aplicativo WebSphere MQ consumindo mensagens de uma fila, um aplicativo que cria uma assinatura e não requer conhecimento de enfileiramento e, finalmente, um exemplo que usa enfileiramento e assinaturas.

Referências relacionadas

DEFINE TOPIC
DISPLAY TOPIC
DISPLAY TPSTATUS

Exemplo 1: publicador em um tópico fixo

Um programa do WebSphere MQ para ilustrar a publicação em um tópico definido administrativamente

Nota: O estilo de codificação compacto destina-se à capacidade de leitura e não ao uso na produção.

Consulte a saída em Figura 38 na página 284

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
int main(int argc, char **argv)
{
    char     topicNameDefault[]   = "IBMSTOCKPRICE";
    char     publicationDefault[] = "129";
    MQCHAR48 qmName = "";

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle                    */
    MQHOBJ   Hobj = MQHO_NONE;            /* object handle sub queue              */
    MQLONG   CompCode = MQCC_OK;          /* completion code                      */
    MQLONG   Reason = MQRC_NONE;          /* reason code                          */
    MQOD     td = {MQOD_DEFAULT};         /* Object descriptor                    */
    MQMD     md = {MQMD_DEFAULT};         /* Message Descriptor                   */
    MQPMO   pmo = {MQPMO_DEFAULT};        /* put message options                  */
    MQCHAR   resTopicStr[151];            /* Returned vale of topic string        */
    char *   topicName   = topicNameDefault;
    char *   publication = publicationDefault;
    memset   (resTopicStr, 0 , sizeof(resTopicStr));

    switch(argc){                         /* replace defaults with args if provided */
        default:
            publication = argv[2];
        case(2):
            topicName = argv[1];
        case(1):
            printf("Optional parameters:  TopicObject Publication\n");
    }
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        td.ObjectType = MQOT_TOPIC;        /* Object is a topic                    */
        td.Version = MQOD_VERSION_4;       /* Descriptor needs to be V4            */
        strncpy(td.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
        td.ResObjectString.VSPtr = resTopicStr;
        td.ResObjectString.VSBufSize = sizeof(resTopicStr)-1;
        MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
        MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode,
&Reason);
        if (CompCode != MQCC_OK) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    if (CompCode == MQCC_OK)
        printf("Published \"%s\" using topic \"%s\" to topic string \"%s\"\n",
               publication, td.ObjectName, resTopicStr);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
```

Figura 37. O publicador do WebSphere MQ simples para um tópico fixo

```
X:\Publish1\Debug>PublishStock
Optional parameters:  TopicObject Publication
Published "129" using topic "IBMSTOCKPRICE" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0

X:\Publish1\Debug>PublishStock IBMSTOCKPRICE 155
Optional parameters:  TopicObject Publication
Published "155" using topic "IBMSTOCKPRICE" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0
```

Figura 38. Saída de amostra do primeiro exemplo de publicador

As linhas de código selecionadas a seguir ilustram aspectos de gravação de um aplicativo publicador para WebSphere MQ.

char topicNameDefault[] = "IBMSTOCKPRICE";
Um nome de tópico padrão é definido no programa. É possível substituí-lo ao fornecer o nome de um objeto do tópico diferente como o primeiro argumento para o programa.

MQCHAR resTopicStr[151];
`resTopicStr` é apontado por `td.ResObjectString.VSPtr` e é usado por `MQOPEN` para retornar a sequência de tópicos resolvida. Torne o comprimento de `resTopicStr` um maior do que o comprimento passado em `td.ResObjectString.VSBufSize` para fornecer espaço para a finalização em nulo.

memset (resTopicStr, 0, sizeof(resTopicStr));
Inicialize `resTopicStr` para nulos para assegurar que a sequência de tópicos resolvida retornada em um `MQCHARV` seja finalizada em nulo.

td.ObjectType = MQOT_TOPIC
Há um novo tipo de objeto para publicar/assinar: o *objeto do tópico*.

td.Version = MQOD_VERSION_4;
Para usar o novo tipo de objeto, deve-se usar pelo menos a *versão 4* do descritor de objeto.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
O `topicName` é o nome de um objeto do tópico, às vezes chamado de um objeto do tópico administrativo. No exemplo, o objeto do tópico precisa ser criado antecipadamente, usando o WebSphere MQ Explorer ou esse comando MQSC,

```
DEFINE TOPIC(IBMSTOCKPRICE) TOPICSTR(NYSE/IBM/PRICE) REPLACE;
```

td.ResObjectString.VSPtr = resTopicStr;
A sequência de tópicos resolvida é ecoada no `printf` final no programa. Configure a estrutura `MQCHARV ResObjectString` para o WebSphere MQ para retornar a sequência resolvida para o programa.

MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
Abra o tópico para saída; exatamente como abrir uma fila para saída.

pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
Você deseja que novos assinantes possam receber a publicação e, especificando `MQPMO_RETAIN` no publicador, ao iniciar um assinante, ele recebe a publicação mais recente, publicada antes do assinante ter sido iniciado, como sua primeira publicação correspondente. A alternativa é fornecer aos assinantes publicações feitas somente após eles terem sido iniciados. Além disso, um assinante tem a opção de recusar a receber uma publicação retida especificando `MQSO_NEW_PUBLICATIONS_ONLY` em sua assinatura.

MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);
Inclua 1 no comprimento da sequência transmitida para `MQPUT` para transmitir o caractere de término nulo para o WebSphere MQ como parte do buffer de mensagem.

O que o primeiro exemplo demonstra? O exemplo imita o mais próximo possível do padrão tradicional experimentado e testado para gravar programas ponto a ponto do WebSphere MQ Um recurso importante do padrão de programação do WebSphere MQ é que o programador não está preocupado para onde as mensagens são enviadas. A tarefa do programador é conectar a um gerenciador de filas e passar a ele as mensagens que devem ser distribuídas a destinatários. No paradigma ponto a ponto, o programador abre uma fila (provavelmente uma fila de alias) que o administrador configurou. A fila de alias roteia mensagens para uma fila de destino, seja no gerenciador de filas locais ou para um gerenciador de filas remotas. Enquanto as mensagens estão esperando para serem entregues, elas são armazenadas em filas em algum lugar entre a origem e o destino.

No padrão de publicar/assinar, em vez de abrir uma fila, o programador abre um tópico. Em nosso exemplo, o tópico é associado a uma sequência de tópicos por um administrador. O gerenciador de filas encaminha a publicação, usando filas, para assinantes locais ou remotos que têm assinaturas que correspondem à sequência de tópicos da publicação. Se as publicações forem retidas, o gerenciador de filas mantém a cópia mais recente da publicação, mesmo se não tiver assinantes agora. A publicação

retida está disponível para encaminhamento para futuros assinantes. O aplicativo publicador não desempenha qualquer papel na seleção ou no roteamento da publicação para um destino; sua tarefa é criar e colocar publicações nos tópicos definidos pelo administrador.

Este exemplo de tópico fixo é atípico de muitos aplicativos de publicar/assinar: é estático. Ele requer que um administrador defina as sequências de tópicos e mude os tópicos nos quais são publicadas. Aplicativos de publicar/assinar geralmente precisam conhecer parte ou toda a árvore de tópicos. Possivelmente, tópicos mudem com frequência ou, embora os tópicos não mudem muito, o número de combinações de tópicos é grande e é demasiado oneroso para um administrador definir um nó de tópico para cada sequência de tópicos nas quais possa ser preciso publicar. Possivelmente, as sequências de tópicos não são conhecidas antes da publicação; um aplicativo publicador pode usar informações do conteúdo de publicação para especificar uma sequência de tópicos ou pode ter informações sobre sequências de tópicos para publicar a partir de outra origem, como inserção manual a partir de um navegador. Para fornecer estilos mais dinâmicos de publicação, o exemplo a seguir mostra como criar tópicos dinamicamente, como parte do aplicativo publicador.

Tópicos colocam publicadores e assinantes em pares. Projetar as regras, ou da arquitetura, para denominar tópicos e organizá-los em árvores de tópicos é uma etapa importante no desenvolvimento de uma solução de publicar/assinar. Verifique cuidadosamente até que ponto a organização da árvore de tópicos junta programas de publicador e assinante e liga os mesmos ao conteúdo da árvore de tópicos. Pergunte a si mesmo se mudanças na árvore de tópicos afetam aplicativos de publicador e assinante e como é possível minimizar o efeito. Construído na arquitetura do modelo de publicação / assinatura do WebSphere MQ é a noção de um objeto de tópico administrativo que fornece a parte raiz ou subárvore raiz de um tópico. O objeto do tópico fornece a opção de definir a parte raiz da árvore de tópicos administrativamente que simplifica a programação de aplicativos e operações, e, consequentemente, melhora a sustentabilidade. Por exemplo, se você estiver implementando diversos aplicativos de publicar/assinar que tenham árvores de tópicos isoladas, então, ao definir administrativamente a parte raiz da árvore de tópicos, será possível garantir o isolamento de árvores de tópicos, mesmo que não houver consistência nas convenções de nomenclatura de tópicos adotadas pelos diferentes aplicativos.

Na prática, os aplicativos do publicador cobrem um espectro de usar exclusivamente tópicos fixos, como neste exemplo, e tópicos variáveis, como no próximo. “Exemplo 2: publicador em um tópico de variável” na página 286 também demonstra combinar o uso de tópicos e as sequências de tópicos.

Conceitos relacionados

“Exemplo 2: publicador em um tópico de variável” na página 286
Um programa do Websphere MQ para ilustrar a publicação em um tópico definido programaticamente.

“Escrevendo aplicativos de assinante” na página 289
Comece a escrever aplicativos de assinante estudando três exemplos: um aplicativo WebSphere MQ consumindo mensagens de uma fila, um aplicativo que cria uma assinatura e não requer conhecimento de enfileiramento e, finalmente, um exemplo que usa enfileiramento e assinaturas.

Exemplo 2: publicador em um tópico de variável

Um programa do Websphere MQ para ilustrar a publicação em um tópico definido programaticamente.

Nota: O estilo de codificação compacto destina-se à capacidade de leitura e não ao uso na produção.

Consulte a saída em Figura 40 na página 288.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
int main(int argc, char **argv)
{
    char     topicNameDefault[]   = "STOCKS";
    char     topicStringDefault[] = "IBM/PRICE";
    char     publicationDefault[] = "130";
    MQCHAR48 qmName = "";

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle           */
    MQHOBJ   Hobj     = MQHO_NONE;   /* object handle sub queue          */
    MQLONG   CompCode = MQCC_OK;     /* completion code                  */
    MQLONG   Reason   = MQRC_NONE;   /* reason code                      */
    MQOD     td = {MQOD_DEFAULT};    /* Object descriptor                */
    MQMD     md = {MQMD_DEFAULT};    /* Message Descriptor               */
    MQPMO   pmo = {MQPMO_DEFAULT};   /* put message options              */
    MQCHAR   resTopicStr[151];       /* Returned value of topic string */
    char *   topicName   = topicNameDefault;
    char *   topicString = topicStringDefault;
    char *   publication = publicationDefault;
    memset   (resTopicStr, 0 , sizeof(resTopicStr));

    switch(argc){            /* Replace defaults with args if provided */
        default:
            publication = argv[3];
        case(3):
            topicString = argv[2];
        case(2):
            if (strcmp(argv[1],"/"))  /* "/" invalid = No topic object  */
                topicName = argv[1];
            else
                *topicName = '\0';
        case(1):
            printf("Provide parameters:  TopicObject TopicString Publication\n");
    }

    printf("Publish \"%s\" to topic \"%-.48s\" and topic string \"%s\"\n", publication,
topicName, topicString);
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        td.ObjectType = MQOT_TOPIC;    /* Object is a topic             */
        td.Version = MQOD_VERSION_4;   /* Descriptor needs to be V4     */
        strncpy(td.ObjectName, topicName,  MQ_TOPIC_NAME_LENGTH);
        td.ObjectString.VSPtr = topicString;
        td.ObjectString.VSLength = (MQLONG)strlen(topicString);
        td.ResObjectString.VSPtr = resTopicStr;
        td.ResObjectString.VSBufSize = sizeof(resTopicStr)-1;
        MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
        MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode,
&Reason);
        if (CompCode != MQCC_OK) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    if (CompCode == MQCC_OK)
        printf("Published \"%s\" to topic string \"%s\"\n",  publication, resTopicStr);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
```

Figura 39. Publicador simples do WebSphere MQ para um tópico variável.

```
X:\Publish2\Debug>PublishStock
Provide parameters:  TopicObject TopicString Publication
Publish "130" to topic "STOCKS" and topic string "IBM/PRICE"
Published "130" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0

X:\Publish2\Debug>PublishStock / NYSE/IBM/PRICE 131
Provide parameters:  TopicObject TopicString Publication
Publish "131" to topic "" and topic string "NYSE/IBM/PRICE"
Published "131" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0
```

Figura 40. Saída de amostra do exemplo do segundo publicador

Há alguns pontos a serem observados sobre este exemplo.

char topicNameDefault[] = "STOCKS";
O padrão de nome de tópico STOCKS define parte da sequência de tópicos. É possível substituir esse nome de tópico fornecendo-o como o primeiro argumento ao programa ou eliminar o uso do nome do tópico fornecendo / como o primeiro parâmetro.

char topicString[101] = "IBM/PRICE";
IBM/PRICE é a sequência de tópicos padrão. É possível substituir essa sequência de tópico fornecendo-a como o segundo argumento ao programa.

O gerenciador de filas combina a sequência de tópicos fornecida pelo STOCKS objeto do tópico, "NYSE", com a sequência de tópicos fornecida pelo programa "IBM/PRICE" e insere um "/" entre as duas sequências de tópicos O resultado é a sequência de tópicos resolvida "NYSE/IBM/PRICE" A sequência de tópicos resultante é a mesma que a definida no objeto do tópico IBMSTOCKPRICE e tem precisamente o mesmo efeito.

O objeto do tópico administrativo associado à sequência de tópico resolvida não é necessariamente o mesmo objeto do tópico que foi passado ao MQOPEN pelo publicador. O WebSphere MQ usa a árvore implícita na sequência de tópicos resolvida para descobrir qual objeto do tópico administrativo define os atributos associados à publicação

Suponha que haja dois objetos de tópico A e Be A define o tópico "a"e B define o tópico "a/b" (Figura 41 na página 289). Se o programa do publicador se referir ao objeto do tópico A e fornecer a sequência de tópicos "b", resolvendo o tópico para a sequência de tópicos "a/b", a publicação herdará suas propriedades do objeto do tópico B porque o tópico corresponde à sequência de tópicos "a/b" definida para B

if (strcmp(argv[1],"/"))
argv[1] é o topicName fornecido opcionalmente. "/" é inválido como um nome de tópico; aqui significa que não há nome de tópico e a sequência de tópicos é fornecida inteiramente pelo programa. A saída em Figura 40 na página 288 mostra a sequência de tópicos inteira a ser fornecida dinamicamente pelo programa.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
Para o caso padrão, o topicName opcional precisa ser criado antecipadamente, usando o WebSphere MQ Explorer ou este comando MQSC:

```
DEFINE TOPIC(STOCKS) TOPICSTR(NYSE) REPLACE;
```

td.ObjectString.VSPtr = topicString;
A sequência de tópicos é um campo MQCHARV no descritor de tópicos

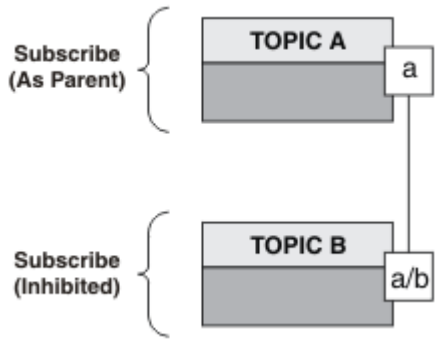


Figura 41. Associações de objeto de tópico

O que o segundo exemplo demonstra? Embora o código seja muito semelhante ao primeiro exemplo - efetivamente, há somente duas linhas de diferença - o resultado é um programa significativamente diferente do primeiro. O programador controla os destinos aos quais publicações são enviadas. Em conjunto com a entrada do administrador mínimo usada para projetar aplicativos de assinante, tópicos ou filas não precisam ser predefinidos para rotear publicações de publicadores a assinantes.

No paradigma do sistema de mensagens ponto a ponto, as filas têm de ser definidas antes que as mensagens sejam capazes de fluir. Para publicar / assinar, eles não o fazem, embora o WebSphere MQ implemente a publicação / assinatura usando seu sistema de enfileiramento subjacente; os benefícios de entrega garantida, transacionalidade e acoplamento solto associados ao sistema de mensagens e enfileiramento são herdados por aplicativos de publicação / assinatura.

Um designer deve decidir se os programas do publicador e do assinante devem estar cientes da árvore de tópicos subjacente ou não e também se os programas do assinante estão cientes do enfileiramento ou não. Estude os aplicativos de exemplo de assinante em seguida. Eles são projetados para serem usados com os exemplos de publicador, geralmente publicando e assinando ao `NYSE/IBM/PRICE`.

Conceitos relacionados

"Exemplo 1: publicador em um tópico fixo" na página 283
Um programa do WebSphere MQ para ilustrar a publicação em um tópico definido administrativamente

"Escrevendo aplicativos de assinante" na página 289
Comece a escrever aplicativos de assinante estudando três exemplos: um aplicativo WebSphere MQ consumindo mensagens de uma fila, um aplicativo que cria uma assinatura e não requer conhecimento de enfileiramento e, finalmente, um exemplo que usa enfileiramento e assinaturas.

Escrevendo aplicativos de assinante

Comece a escrever aplicativos de assinante estudando três exemplos: um aplicativo WebSphere MQ consumindo mensagens de uma fila, um aplicativo que cria uma assinatura e não requer conhecimento de enfileiramento e, finalmente, um exemplo que usa enfileiramento e assinaturas.

No Tabela 42 na página 290 , os três estilos de consumidor ou assinante são listados, juntamente com as sequências de chamadas de funções do WebSphere MQ que os caracterizam..

1. O primeiro estilo, MQ Publicação Consumidor, é idêntico a um programa MQ ponto a ponto que executa apenas MQGET. O aplicativo não tem conhecimento de que está consumindo as publicações – está simplesmente lendo mensagens de uma fila. A assinatura que faz as publicações serem roteadas para a fila é criada administrativamente usando o WebSphere MQ Explorer ou um comando.
2. O segundo estilo é o padrão preferido para a maioria dos aplicativos de assinantes. O aplicativo do assinante cria a assinatura e, em seguida, obtém publicações. O de gerenciamento de filas são todas executadas pelo gerenciador de filas.
3. No terceiro estilo, o aplicativo assinante opta por abrir e fechar a fila subjacente que é utilizada para as publicações bem como emitir assinaturas para preencher a fila com as publicações.

Uma maneira de entender esses estilos é estudar os programas C de exemplo listados em Tabela 42 na página 290 para cada um dos estilos.. Os exemplos são projetados para serem executados em conjunto com o exemplo publicador localizado em "Gravando aplicativos de publicador" na página 282.

Tabela 42. Ponto a ponto vs. assinar os padrões do programa WebSphere MQ

Salto	consumidor de mensagens do MQ	"Exemplo 1: consumidor de publicação do MQ" na página 290	"Exemplo 2: assinante do MQ gerenciado" na página 293	"Exemplo 3: assinante do MQ não gerenciado" na página 298
Conecta-se a um gerenciador de filas	MQCONN	MQCONN	MQCONN	MQCONN
Abrir fila	MQOPEN	MQOPEN		MQOPEN
Assinar			MQSUB	MQSUB
Obtém mensagem(ns)	MQGET	MQGET	MQGET	MQGET
Fechar fila	MQCLOSE	MQCLOSE	(MQCLOSE)	MQCLOSE
Fechar assinatura			MQCLOSE	MQCLOSE
Desconecte a partir do gerenciador de filas	MQDISC	MQDISC	MQDISC	MQDISC

Usar MQCLOSE sempre é opcional, seja para liberar recurso, para passar opções do MQCLOSE ou apenas para simetria com MQOPEN. Como é improvável que você precise especificar as opções MQCLOSE quando a fila de assinaturas é fechada no caso do assinante do MQ gerenciado e o argumento de simetria não é relevante, a fila de assinaturas não é explicitamente fechada no Exemplo 2: assinante do MQ gerenciado .

Outra maneira de entender os padrões de aplicativos de publicação/assinatura é analisar as interações entre as diferentes entidades envolvidas. Linha de vida ou diagramas de sequência UML são uma boa maneira de estudar interações. Três exemplos de linha de vida são descritos em "Ciclos de vida de publicar/assinar" na página 307.

Exemplo 1: consumidor de publicação do MQ

O consumidor da publicação MQ é um consumidor de mensagens IBM WebSphere MQ que não assina tópicos em si.

Para criar a fila de assinatura e de publicação para este exemplo, execute os comandos a seguir ou defina os objetos usando o WebSphere MQ Explorer

```
DEFINE QLOCAL(STOCKTICKER) REPLACE;
DEFINE SUB(IBMSTOCKPRICESUB) DEST(STOCKTICKER) TOPICOBJ(IBMSTOCKPRICE) REPLACE;
```

A assinatura IBMSTOCKPRICESUB faz referência ao objeto do tópico IBMSTOCK criado para o exemplo de publicador e a fila local STOCKTICKER. O objeto do tópico IBMSTOCK define a sequência de tópicos que é usada na assinatura, NYSE/IBM/PRICE. Observe que o objeto do tópico e a fila usada para receber publicações precisam ser definidos antes de a assinatura ser criada.

Existem várias máscaras valiosas para o padrão do consumidor de publicação MQ:

1. Multiprocessamento: compartilhar do trabalho de publicações de leitura. Todas as publicações vão para a única fila associada ao tópico de assinatura. Vários consumidores podem abrir a fila usando MQOO_INPUT_SHARED.
2. Assinaturas gerenciadas centralmente. Os aplicativos não constroem sua própria assinatura ou assinaturas de tópicos; o administrador é responsável por aonde as publicações são enviadas.

3. Concentração de assinatura: várias assinaturas diferentes podem ser enviadas para uma única fila.
4. Durabilidade da assinatura: a fila recebe todas as publicações estando ou não os consumidores ativos.
5. Migração e coexistência: o código do consumidor trabalha igualmente bem para um ponto a ponto e para um cenário de publicação/assinatura.

A assinatura cria um relacionamento entre a sequência de tópicos `NYSE/IBM/PRICE` e a fila `STOCKTICKER`. As publicações, incluindo qualquer publicação retida atualmente, são encaminhadas para `STOCKTICKER` a partir do momento em que a assinatura for criada.

Uma assinatura administrativamente criada pode ser gerenciada ou não gerenciada. Uma assinatura gerenciada entre em vigor assim que ela tiver sido criada, exatamente como uma assinatura não gerenciada. Nem todas as máscaras padrão estão disponíveis para uma assinatura gerenciada. Consulte “Exemplo 3: assinante do MQ não gerenciado” na página 298

Nota: O estilo de codificação compacto destina-se à capacidade de leitura e não ao uso na produção.

Os resultados são mostrados em Figura 43 na página 292.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
int main(int argc, char **argv)
{
    MQCHAR      publicationBuffer[101];
    MQCHAR48 subscriptionQueueDefault = "STOCKTICKER";
    MQCHAR48 qmName = "";                       /* Use default queue manager    */

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN;       /* connection handle            */
    MQHOBJ   Hobj = MQHO_NONE;                  /* object handle sub queue      */
    MQLONG   CompCode = MQCC_OK;                /* completion code              */
    MQLONG   Reason = MQRC_NONE;                /* reason code                  */
    MQLONG   messlen = 0;
    MQOD     od = {MQOD_DEFAULT};               /* Unmanaged subscription queue */
    MQMD     md = {MQMD_DEFAULT};               /* Message Descriptor           */
    MQGMO    gmo = {MQGMO_DEFAULT};             /* Get message options          */
    char *   publication=publicationBuffer;
    char *   subscriptionQueue = subscriptionQueueDefault;

    switch(argc){                     /* Replace defaults with args if provided */
        default:
            subscriptionQueue = argv[1]
        case(1):
            printf("Optional parameter: subscriptionQueue\n");
    }

    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        strncpy(od.ObjectName, subscriptionQueue, MQ_Q_NAME_LENGTH);
        MQOPEN(Hconn, &od, MQOO_INPUT_AS_Q_DEF | MQOO_FAIL_IF_QUIESCING , &Hobj, &CompCode,
&Reason);
        if (CompCode != MQCC_OK) break;
        gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT | MQGMO_CONVERT;
        gmo.WaitInterval = 10000;
        printf("Waiting %d seconds for publications from %s\n", gmo.WaitInterval/1000,
subscriptionQueue);
        do {
            memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
            memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
            md.Encoding       = MQENC_NATIVE;
            md.CodedCharSetId = MQCCSI_Q_MGR;
            memset(publication, 0, sizeof(publicationBuffer));
            MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1, publication, &messlen,
                  &CompCode, &Reason);
            if (Reason == MQRC_NONE)
                printf("Received publication \"%s\"\n", publication);
        }
        while (CompCode == MQCC_OK);
        if (CompCode != MQCC_OK && Reason != MQRC_NO_MSG_AVAILABLE) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
```

Figura 42. Consumidor de publicação MQ.

```
X:\Subscribe1\Debug>Subscribe1
Optional parameter: subscriptionQueue
Waiting 10 seconds for publications from STOCKTICKER
Received publication "129"
Completion code 0 and Return code 0
```

Figura 43. A saída do consumidor de publicação MQ

Há algumas dicas de programação de linguagem padrão do WebSphere MQ *C* para estar ciente de:

memset(publication, 0, sizeof(publicationBuffer));
Assegure-se de que a mensagem tenha um nulo final para fácil formatação usando `printf`. O exemplo do publicador inclui o nulo final no buffer de mensagem passado para o `MQPUT` incluindo 1 ao `strlen(publication)`. Configurar `MQCHAR` buffers para nulo é bom para o estilo de programação de programas C IBM WebSphere MQ que usam os buffers para armazenar sequências, garantindo que um nulo siga uma matriz de caracteres que não preenche totalmente o buffer.

MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1, publication, &messlen, &CompCode, &Reason);
Reserve um nulo no final do buffer de mensagem para assegurar que a mensagem retornada tenha nulo final no caso de "`if (messlen == strlen(publication));`" ser true. Essa dica complementa a anterior e assegura que haja pelo menos uma nula no `publicationBuffer` que não será sobrescrita pelo conteúdo de `publication`.

Conceitos relacionados

“Exemplo 2: assinante do MQ gerenciado” na página 293
O assinante gerenciado do MQ é o padrão preferencial para a maioria dos aplicativos de assinante O exemplo requer *nenhuma* definição administrativa de filas, tópicos ou assinaturas

“Exemplo 3: assinante do MQ não gerenciado” na página 298
O assinante não gerenciado é uma classe importante de aplicativo de assinante. Com ele, você combina os benefícios de publicar/assinar com *controle* de enfileiramento e consumo de publicações. O exemplo demonstra maneiras diferentes de combinar assinaturas e filas.

“Gravando aplicativos de publicador” na página 282
Comece a escrever aplicativos do publicador estudando dois exemplos. O primeiro é modelado o mais próximo possível em um aplicativo ponto a ponto colocando mensagens em uma fila e o segundo demonstra a criação de tópicos dinamicamente - um padrão mais comum para aplicativos publicadores.

Exemplo 2: assinante do MQ gerenciado

O assinante gerenciado do MQ é o padrão preferencial para a maioria dos aplicativos de assinante O exemplo requer *nenhuma* definição administrativa de filas, tópicos ou assinaturas

Esse tipo mais simples do assinante gerenciado do geralmente usa uma assinatura *não durável* O exemplo foca uma assinatura não durável. A assinatura dura apenas enquanto o tempo de vida do identificador de assinatura de `MQSUB`... Quaisquer publicações que correspondam à sequência de tópicos durante o tempo de vida da assinatura são enviadas para a fila de assinaturas (e possivelmente uma publicação retida se o sinalizador `MQSO_NEW_PUBLICATIONS_ONLY` não for configurado ou padronizado, uma publicação anterior correspondente à sequência de tópicos foi retida e a publicação foi persistente ou o gerenciador de filas não foi finalizado, desde que a publicação foi criada.

Também é possível usar uma assinatura *durável* com este padrão. Geralmente se uma assinatura durável gerenciada for usada, isso será feito por motivos de confiabilidade, em vez de estabelecer uma assinatura que, sem que ocorram erros, sobreviveria ao assinante.. Para obter mais informações sobre diferentes ciclos de vida associados a assinaturas gerenciadas, não gerenciadas, duráveis e não duráveis, consulte a seção de tópicos relacionados

Assinaturas duráveis são frequentemente associadas a publicações persistentes e assinaturas não duráveis a publicações não persistentes, mas não há relacionamento necessário entre durabilidade de assinatura e persistência de publicação. Todas as quatro combinações de persistência e durabilidade são possíveis.

Para o caso de não duráveis gerenciadas considerado, o gerenciador de filas cria uma fila de assinaturas que é limpa e excluída quando a fila é fechada. As publicações são removidas da fila quando a assinatura não durável é fechada.

Os aspectos importantes do padrão não durável gerenciada exemplificado por este código são os seguintes:

1. Assinatura sob demanda do : a sequência de tópicos de assinatura é dinâmica. É fornecida pelo aplicativo quando ele é executado.
2. Fila autogerenciada: a fila de assinatura é autodefinida e gerenciada.

3. Ciclo de vida de assinatura de autogerenciamento: *não-durável* assinaturas existem apenas para a duração do aplicativo assinante.
   - Se você definir uma assinatura gerenciada *durável* , isso resultará em uma fila de assinaturas permanente e as publicações continuarão sendo armazenadas nela sem nenhum programa de assinante estar ativo. O gerenciador de filas exclui a fila (e limpa quaisquer publicações não recuperadas da mesma) somente após o aplicativo ou o administrador ter optado por excluir a assinatura. A assinatura pode ser excluída usando um comando administrativo ou fechando a assinatura com a opção `MQCO_REMOVE_SUB`.
   - Considere configurar `SubExpiry` para assinaturas duráveis para que as publicações deixem de ser enviadas à fila e o assinante possa consumir quaisquer publicações restantes antes de remover a assinatura e fazer com que o gerenciador de filas exclua a fila e quaisquer publicações restantes nela.
4. Implementação de sequência de tópicos flexível: o gerenciamento de tópicos de assinatura é simplificado definindo-se a parte raiz da assinatura usando um tópico definido administrativamente. A parte raiz da árvore de tópicos é então ocultada do aplicativo. Ao ocultar a parte raiz, um aplicativo pode ser implementado sem que o aplicativo crie acidentalmente uma árvore de tópicos que se sobreponha a outra árvore de tópicos criada por outra instância ou outro aplicativo.
5. Tópicos administrados: por usando uma sequência de tópicos na qual a primeira parte corresponde a um objeto de tópico definido administrativamente, as publicações são gerenciadas de acordo com os atributos do objeto do tópico.
   - Por exemplo, se a primeira parte da sequência de tópicos corresponder à sequência de tópicos associada a um objeto do tópico em cluster, a assinatura poderá receber publicações de outros membros do cluster
   - A correspondência seletiva de objetos do tópico definidos administrativamente e assinaturas definidas programaticamente permite combinar os benefícios de ambos. O administrador fornece atributos para tópicos e o programador define dinamicamente "sub-tópicos" sem se preocupar com o gerenciamento de tópicos.
   - Ele é a sequência de tópicos resultante que é usada para corresponder ao objeto do tópico que fornece os atributos associados ao tópico e não necessariamente o objeto do tópico denominado em `sd.Objectname`, embora eles geralmente se transformem em um. Consulte “Exemplo 2: publicador em um tópico de variável” na página 286..

Ao tornar a assinatura durável no exemplo, as publicações continuam a ser enviadas para a fila de assinaturas após o assinante ter fechado a assinatura com a `MQCO_KEEP_SUB` opção A fila continua a receber publicações quando o assinante não está ativo. É possível substituir esse comportamento ao criar a assinatura com a opção `MQSO_PUBLICATIONS_ON_REQUEST` e usar `MQSUBRQ` para solicitar a publicação retida.

A assinatura pode ser continuada posteriormente abrindo a assinatura com a opção `MQCO_RESUME`.

É possível usar o identificador de fila, `Hobj`, retornado por `MQSUB` de várias maneiras. O identificador de fila é usado no exemplo para consultar sobre o nome da fila de assinaturas. Filas gerenciadas são abertas usando as filas modelo padrão `SYSTEM.NDURABLE.MODEL.QUEUE` ou `SYSTEM.DURABLE.MODEL.QUEUE`. É possível substituir os padrões fornecendo suas próprias filas de modelos duráveis e não duráveis em um tópico por tópico como propriedades do objeto do tópico associado à assinatura..

Independentemente dos atributos herdados das filas modelo, não é possível reutilizar um identificador de fila gerenciado para criar uma assinatura adicional. Nem é possível obter outro identificador para a fila gerenciada abrindo a fila gerenciada pela segunda vez usando o nome da fila retornado. A fila se comporta como se estivesse aberta para entradaexclusiva.

Filas não gerenciadas são mais flexíveis do que filas gerenciadas. É possível, por exemplo, compartilhar filas não gerenciadas ou definir várias assinaturas em uma fila. O próximo exemplo, “Exemplo 3: assinante do MQ não gerenciado” na página 298, demonstra como combinar assinaturas com uma fila de assinaturas não gerenciada.

Nota: O estilo de codificação compacto destina-se à capacidade de leitura e não ao uso na produção.

Os resultados são mostrados em Figura 46 na página 296.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>

void inquireQname(MQHCONN HConn, MQHOBJ Hobj, MQCHAR48 qName);

int main(int argc, char **argv)
{
    MQCHAR48 topicNameDefault     = "STOCKS";
    char     topicStringDefault[] = "IBM/PRICE";
    MQCHAR48 qmName = "";                 /* Use default queue manager     */
    MQCHAR48 qName = "";                  /* Allocate to query queue name  */
    char     publicationBuffer[101];      /* Allocate to receive messages  */
    char     resTopicStrBuffer[151];   /* Allocate to resolve topic string */

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle             */
    MQHOBJ   Hobj = MQHO_NONE;            /* publication queue handle      */
    MQHOBJ   Hsub = MQSO_NONE;            /* subscription handle           */
    MQLONG   CompCode = MQCC_OK;          /* completion code               */
    MQLONG   Reason = MQRC_NONE;          /* reason code                   */
    MQLONG   messlen = 0;
    MQSD     sd = {MQSD_DEFAULT};         /* Subscription Descriptor       */
    MQMD     md = {MQMD_DEFAULT};         /* Message Descriptor            */
    MQGMO   gmo = {MQGMO_DEFAULT};        /* get message options           */

    char *   topicName   = topicNameDefault;
    char *   topicString = topicStringDefault;
    char *   publication = publicationBuffer;
    char *   resTopicStr = resTopicStrBuffer;
    memset(resTopicStr, 0, sizeof(resTopicStrBuffer));

    switch(argc){               /* Replace defaults with args if provided */
        default:
          topicString = argv[2];
        case(2):
          if (strcmp(argv[1],"/"))       /* "/" invalid = No topic object  */
             topicName = argv[1];
          else
             *topicName = '\0';
        case(1):
             printf("Optional parameters: topicName, topicString\nValues \"%s\"  \"%s\"\n",
                   topicName, topicString);
    }
```

Figura 44. Assinante do MQ gerenciado-parte 1: declarações e manipulação de parâmetro.

Há alguns comentários adicionais para fazer sobre as declarações neste exemplo.

MQHOBJ Hobj = MQHO_NONE;
Não é possível abrir explicitamente uma fila de assinaturas gerenciadas não duráveis para receber publicações, mas é necessário alocar armazenamento para a manipulação de objetos que o gerenciador de filas retorna quando ele abre a fila. É importante inicializar a manipulação para MQHO_OBJECT.. Isso indica para o gerenciador de filas que ele precisa retornar um identificador de fila para a fila de assinaturas

MQSD sd = {MQSD_DEFAULT};
O novo descritor de assinatura, usado em MQSUB.

MQCHAR48 qName;
Embora o exemplo n' t requeira conhecimento da fila de assinaturas, o exemplo consulta o nome da fila de assinaturas-a ligação MQINQ é um pouco estranha na linguagem C, portanto, você pode achar essa parte do exemplo útil para estudar.

```
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        strncpy(sd.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
        sd.ObjectString.VSPtr = topicString;
        sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
        sd.Options = MQSO_CREATE | MQSO_MANAGED | MQSO_NON_DURABLE | MQSO_FAIL_IF_QUIESCING ;
        sd.ResObjectString.VSPtr = resTopicStr;
        sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
        MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT | MQGMO_CONVERT;
        gmo.WaitInterval = 10000;
        inquireQname(Hconn, Hobj, qName);
        printf("Waiting %d seconds for publications matching \"%s\" from \"%-0.48s\"\n",
               gmo.WaitInterval/1000, resTopicStr, qName);
        do {
            memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
            memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
            md.Encoding       = MQENC_NATIVE;
            md.CodedCharSetId = MQCCSI_Q_MGR;
            memset(publicationBuffer, 0, sizeof(publicationBuffer));
            MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer-1),
                publication, &messlen, &CompCode, &Reason);
            if (Reason == MQRC_NONE)
                printf("Received publication \"%s\"\n", publication);
        }
        while (CompCode == MQCC_OK);
        if (CompCode != MQCC_OK && Reason != MQRC_NO_MSG_AVAILABLE) break;
        MQCLOSE(Hconn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
    return;
}
void inquireQname(MQHCONN Hconn, MQHOBJ Hobj, MQCHAR48 qName) {
#define _selectors 1
#define _intAttrs 1

    MQLONG select[_selectors] = {MQCA_Q_NAME}; /* Array of attribute selectors */
    MQLONG intAttrs[_intAttrs];                /* Array of integer attributes  */
    MQLONG CompCode, Reason;
    MQINQ(Hconn, Hobj, _selectors, select, _intAttrs, intAttrs, MQ_Q_NAME_LENGTH, qName,
          &CompCode, &Reason);
    if (CompCode != MQCC_OK) {
        printf("MQINQ failed with Condition code %d and Reason %d\n", CompCode, Reason);
        strcpy(qName, "unknown queue");
    }
    return;
}
```

Figura 45. Assinante gerenciado do MQ -parte 2: corpo do código.

```
W:\Subscribe2\Debug>solution2
Optional parameters: topicName, topicString
Values "STOCKS"  "IBM/PRICE"
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from
"SYSTEM.MANAGED.NDURABLE.48A0AC7403300020        "
Received publication "150"
Completion code 0 and Return code 0

W:\Subscribe2\Debug>solution2 / NYSE/IBM/PRICE
Optional parameters: topicName, topicString
Values ""  "NYSE/IBM/PRICE"
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from
"SYSTEM.MANAGED.NDURABLE.48A0AC7403310020        "
Received publication "150"
Completion code 0 and Return code 0
```

Figura 46. Saída do assinante gerenciado do MQ

Há alguns comentários adicionais para fazer sobre o código neste exemplo.

strncpy(sd.ObjectName, topicName, MQ_Q_NAME_LENGTH);
Se topicName for nulo ou estiver em branco *(valor padrão)*, o nome do tópico não será usado para calcular a sequência de tópicos resolvida.

sd.ObjectString.VSPtr = topicString;
Em vez de usar exclusivamente um objeto do tópico predefinido, neste exemplo, o programador fornece um objeto do tópico e uma sequência de tópicos, que são combinados por MQSUB. Observe que a sequência de tópicos é uma estrutura MQCHARV.

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Uma alternativa para configurar o comprimento de um campo MQCHARV.

sd.Options = MQSO_CREATE | MQSO_MANAGED | MQSO_NON_DURABLE | MQSO_FAIL_IF_QUIESCING;
Após definir a sequência de tópicos, as sinalizações sd.Options precisam de muita atenção. Há muitas opções, o exemplo especifica apenas as mais comumente usadas; as outras são deixadas padrão.

1. Como a assinatura é *não durável*, ou seja, ela tem um tempo de vida da assinatura aberta no aplicativo, configure o MQSO_CREATE sinalizador Também é possível configurar o sinalizador *(padrão)* MQSO_NON_DURABLE para capacidade de leitura.
2. Complementando MQSO_CREATE há MQSO_RESUME. Ambos os sinalizadores podem ser configurados juntos; o gerenciador de filas cria uma nova assinatura ou retoma uma assinatura existente, o que for apropriado No entanto, se você especificar MQSO_RESUME, deve-se também inicializar a estrutura MQCHARV para sd.SubName, mesmo se não houver nenhuma assinatura para continuar. Falha ao inicializar SubName resulta em um código de retorno 2440: MQRC_SUB_NAME_ERROR de MQSUB.

   Nota: MQSO_RESUME sempre e ignorado para uma assinatura gerenciada não durável, mas especificá-lo sem inicializar a estrutura MQCHARV para sd.SubName causa o erro.
3. Além disso, há uma terceira sinalização que afeta como a assinatura é aberta, MQSO_ALTER. Dadas as permissões certas, as propriedades de uma assinatura continuada são alteradas para corresponder a outros atributos especificados em MQSUB.

   Nota: Pelo menos uma das sinalizações MQSO_CREATE, MQSO_RESUME e MQSO_ALTER deve ser especificada. Consulte Opções (MQLONG). Há exemplos de uso de todas as três sinalizações em “Exemplo 3: assinante do MQ não gerenciado” na página 298.
4. Configure MQSO_MANAGED para o gerenciador de filas gerenciar a assinatura para você automaticamente.

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Como opção, omita a configuração do comprimento de MQCHARV para sequências finalizadas em nulo e use a sinalização de terminador nulo.

sd.ResObjectString.VSPtr = resTopicStr;
A sequência de tópicos resultante é ecoada no primeiro printf no programa. Configure MQCHARV ResObjectString para o WebSphere MQ para retornar a sequência resolvida para o programa.

Nota: resTopicStringBuffer é inicializado para nulos em memset(resTopicStr, 0, sizeof(resTopicStrBuffer)). As sequências de tópicos retornadas não terminam com nulos à direita.

sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
Configure o tamanho do buffer de sd.ResObjectString para um a menos que seu tamanho real. Isso evita sobrescrever o terminador nulo fornecido, caso a sequência de tópicos resolvida preencha o buffer inteiro.

Nota: Nenhum erro será retornado se a sequência de tópicos for mais longa que sizeof(resTopicStrBuffer)-1. Mesmo se VSLength > VSBufSiz, o comprimento retornado em sd.ResObjectString.VSLength será o comprimento da sequência completa e não necessariamente o comprimento da sequência retornada. Teste sd.ResObjectString.VSLength < sd.ResObjectString.VSBufSiz para confirmar a sequência de tópicos concluída.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
A função MQSUB cria uma assinatura. Se não for durável, você provavelmente não está interessado em seu nome, embora possa inspecionar seu status no WebSphere MQ Explorer. É possível fornecer o parâmetro sd.SubName como input, para que você saiba qual nome procurar; obviamente é necessário evitar conflitos de nomes com outras assinaturas.

MQCLOSE(Hconn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
Fechar a assinatura e a fila de assinaturas é opcional. No exemplo, a assinatura é fechada, mas não a fila. A opção MQCLOSE MQCO_REMOVE_SUB é o padrão nesse caso de qualquer forma, já que a assinatura é não durável. Usar MQCO_KEEP_SUB é um erro.

Nota: a *fila* de assinaturas não é fechada por MQSUB e seu identificador, Hobj, permanece válido até que a fila seja fechada por MQCLOSE ou MQDISC. Se o aplicativo for finalizado de forma prematura, a fila e a assinatura são limpas pelo gerenciador de filas algum tempo após a finalização do aplicativo.

Conceitos relacionados

"Exemplo 1: consumidor de publicação do MQ" na página 290
O consumidor da publicação MQ é um consumidor de mensagens IBM WebSphere MQ que não assina tópicos em si.

"Exemplo 3: assinante do MQ não gerenciado" na página 298
O assinante não gerenciado é uma classe importante de aplicativo de assinante. Com ele, você combina os benefícios de publicar/assinar com *controle* de enfileiramento e consumo de publicações. O exemplo demonstra maneiras diferentes de combinar assinaturas e filas.

"Gravando aplicativos de publicador" na página 282
Comece a escrever aplicativos do publicador estudando dois exemplos. O primeiro é modelado o mais próximo possível em um aplicativo ponto a ponto colocando mensagens em uma fila e o segundo demonstra a criação de tópicos dinamicamente - um padrão mais comum para aplicativos publicadores.

Exemplo 3: assinante do MQ não gerenciado

O assinante não gerenciado é uma classe importante de aplicativo de assinante. Com ele, você combina os benefícios de publicar/assinar com *controle* de enfileiramento e consumo de publicações. O exemplo demonstra maneiras diferentes de combinar assinaturas e filas.

O padrão não gerenciado é mais comumente associado a assinaturas *duráveis* do que *não duráveis*. Geralmente, o ciclo de vida de uma assinatura criada por um assinante não gerenciado é independente do ciclo de vida do aplicativo de assinatura em si. Ao tornar a assinatura durável, a assinatura recebe publicações mesmo quando nenhum aplicativo de assinatura estiver ativo.

É possível criar assinaturas duráveis *gerenciadas* para atingir o mesmo resultado, mas alguns aplicativos requerem mais flexibilidade e controle sobre filas e mensagens do que é possível com uma assinatura gerenciada. Para uma assinatura gerenciada durável, o gerenciador de filas cria uma fila permanente para as publicações que correspondem ao tópico de assinatura. Ele exclui a fila e as publicações associadas quando a assinatura é excluída.

Geralmente, assinaturas *gerenciadas* duráveis são usadas se o ciclo de vida do aplicativo e da assinatura for essencialmente o mesmo, mas é difícil garantir. Ao tornar a assinatura durável e fazer com que o publicador crie publicações persistentes, não haverá mensagens perdidas caso o gerenciador de filas ou o assinante seja finalizado prematuramente e precise ser recuperado.

O gerenciador de filas abre implicitamente a fila de assinaturas gerenciadas duráveis para um assinante de tal forma que o processamento compartilhado da fila não seja possível. Além disso, não é possível criar mais de uma assinatura para cada fila gerenciada e você pode achar mais difícil gerenciar as filas, pois tem menos controle sobre os nomes das filas. Por esses motivos, considere se o assinante *não gerenciado* do MQ é mais adequado para aplicativos que requerem assinaturas duráveis do que o assinante *gerenciado* do MQ.

O código em Figura 49 na página 304 demonstra um padrão de assinatura durável não gerenciada. Para ilustração, o código também cria assinaturas não gerenciadas, não duráveis. Este exemplo ilustra as máscaras padrão a seguir:

- Assinaturas on demand: as sequências de tópicos de assinatura são dinâmicas. Elas são fornecidas pelo aplicativo quando ele é executado.
- Gerenciamento de tópicos de assinatura simplificado: o gerenciamento de tópicos de assinatura é simplificado definindo-se a parte raiz da sequência de tópicos de assinatura usando um tópico definido administrativamente. Isso oculta a parte raiz da árvore de tópicos do aplicativo. Ao ocultar a parte raiz, um assinante pode ser implementado em diferentes árvores de tópicos.
- Gerenciamento de assinatura flexível: é possível definir uma assinatura administrativamente ou criá-la on demand em um programa do assinante. Não há diferença entre assinaturas criadas administrativamente e programaticamente, exceto um atributo que mostra como a assinatura foi criada. Há um terceiro tipo de assinatura que é criado automaticamente pelo gerenciador de filas para distribuição de assinaturas. Todas as assinaturas são exibidas no WebSphere MQ Explorer.
- Associação flexível de assinaturas com filas: uma fila local predefinida é associada a uma assinatura pela função MQSUB. Há maneiras diferentes de usar MQSUB para associar assinaturas a filas:
  - Associe uma assinatura a uma fila que não tenha *nenhuma* assinatura existente, `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Associe uma *nova* assinatura a uma fila que tenha assinaturas existentes, `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Mova uma assinatura existente para uma fila diferente, `MQSO_ALTER + (Hobj from MQOPEN)`
  - Retome uma assinatura existente associada a uma fila existente, `MQSO_RESUME + (Hobj = MQHO_NONE)` ou `MQSO_RESUME + (Hobj = from MQOPEN of queue with existing subscription)`
  - Ao combinar `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` em diferentes combinações, é possível atender diferentes estados de entrada da assinatura e da fila sem precisar codificar várias versões do MQSUB com diferentes valores de `sd.Options`.
  - Como alternativa, ao codificar uma opção específica `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` , o gerenciador de filas retornará um erro (Tabela 43 na página 300) se os estados da assinatura e da fila fornecidos como entrada para MQSUB estiverem inconsistentes com o valor de `sd.Options`. Figura 55 na página 307 mostra os resultados da emissão de MQSUB para Assinatura X com diferentes configurações individuais do sinalizador `sd.Options` e transmitindo a ele três identificadores de objetos diferentes

  Explore entradas diferentes para o programa de exemplo em Figura 48 na página 303 para se familiarizar com esses tipos diferentes de erros. Um erro comum, `RC = 2440`, que não está incluído nos casos listados na tabela, é um erro de nome da assinatura. geralmente é causado pela transmissão de um nome de assinatura nulo ou inválido com `MQSO_RESUME` ou `MQSO_ALTER` .
- Multiprocessamento: é possível compartilhar entre muitos consumidores o trabalho de leitura de publicações. Todas as publicações vão para a única fila associada ao tópico de assinatura. Consumidores têm a opção de abrir a fila diretamente usando MQOPEN ou continuar a assinatura usando MQSUB.
- Concentração de assinaturas: várias assinaturas podem ser criadas na mesma fila. Tenha cuidado com esse recurso, pois ele pode levar a assinaturas "sobrepostas" e receber a mesma publicação várias vezes. A opção `MQSO_GROUP_SUB` elimina publicações duplicadas causadas pela sobreposição de assinaturas.
- Separação de assinante e consumidor: assim como os três modelos de consumidor ilustrados nos exemplos, outro modelo é separar o consumidor do assinante. É uma variação do Assinante do MQ não gerenciado, mas em vez de emitir o MQOPEN e o MQSUB no mesmo programa, um programa assina publicações e outro programa as consome. Por exemplo, o assinante pode fazer parte de um cluster de publicação/assinatura e o consumidor estar anexado a um gerenciador de filas fora do cluster de gerenciador de filas. O consumidor recebe publicações por meio do enfileiramento distribuído padrão definindo a fila de assinaturas como uma definição de fila remota.

O entendimento do comportamento do `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` é importante, especialmente se você planeja simplificar seu código usando combinações dessas opções Estude a tabela Tabela 43 na página 300 que mostra os resultados de passar diferentes identificadores de filas

para MQSUB e os resultados de executar o programa de exemplo mostrado em Figura 50 na página 305 para Figura 55 na página 307.

O cenário usado para construir a tabela possui uma assinatura X e duas filas, A e B O parâmetro do nome da assinatura sd.SubName é configurado como X, o nome de uma assinatura anexada na fila A. A fila B não tem assinatura anexada a ela.

Em Tabela 43 na página 300, MQSUB é transmitida a assinatura X e a manipulação de fila para a fila A Os resultados das opções de assinatura são os seguintes:

- O MQSO_CREATE falha porque o identificador de filas corresponde à fila A que já possui uma assinatura para X Contraste esse comportamento para a chamada bem-sucedida. Aquela chamada é bem-sucedida porque a fila B não tem uma assinatura para X anexada a ela.
- MQSO_RESUME tem êxito porque o identificador de fila corresponde à fila A que já tem uma assinatura de X. Em contraste, a chamada falha quando a assinatura X não existe na fila A.
- MQSO_ALTER se comporta de maneira semelhante ao MQSO_RESUME com relação à abertura da assinatura e da fila. No entanto, se os atributos contidos no descritor de assinatura passado para MQSUB diferirem dos atributos da assinatura, MQSO_RESUME falhará, enquanto MQSO_ALTER será bem-sucedido, contanto que a instância de programa tenha permissão para alterar os atributos Observe que nunca é possível alterar a sequência de tópicos em uma assinatura, mas em vez de retornar um erro, o MQSUB ignora o nome do tópico e os valores da sequência de tópicos no descritor de assinatura e usa os valores na assinatura existente..

Em seguida, analise Tabela 43 na página 300 em que MQSUB é transmitido por assinatura X e o identificador de fila para a fila B. Os resultados das opções de assinatura são os seguintes:

- MQSO_CREATE é bem-sucedido e cria a assinatura X na fila B , pois essa é uma nova assinatura na fila B
- MQSO_RESUME falha. O MQSUB procura a assinatura X na fila B e não a localiza, mas em vez de retornar *RC = 2428-a assinatura X não existe* , ele retorna *RC = 2019-A fila de assinaturas não corresponde à manipulação de objetos da fila*. O comportamento da terceira opção MQSO_ALTER sugere a razão para esse erro inesperado. MQSUB espera que o manipulador de filas aponte para uma fila com uma assinatura. Ele verifica isso primeiro antes de verificar se a assinatura denominada em sd.SubName existe.
- MQSO_ALTER é bem-sucedido e move a assinatura da fila A para a fila B.

Um caso que não é mostrado na tabela é se o nome da assinatura na fila A não corresponde ao nome da assinatura em sd.SubName. Essa chamada falha com um *RC = 2428-a assinatura X não existe na fila A* .

Tabela 43. Erros de MQSUB com diferentes identificadores de filas e combinações de assinaturas

	Fila A **Assinatura X** **Fila B** **Nenhuma assinatura**	**Fila A** **Nenhuma assinatura** **Fila B** **Nenhuma assinatura**
Hobj para *Fila A* passado para MQSUB	**MQSO_CREATE** RC = 2432 – A assinatura X já existe na Fila A **MQSO_RESUME** Continua a assinatura X na Fila A **MQSO_ALTER** Continua a assinatura X na Fila A e faz alterações permitidas	**MQSO_CREATE** Cria a assinatura X na Fila A **MQSO_RESUME** RC = 2428 – A assinatura X não existe na Fila A **MQSO_ALTER** RC = 2428 – A assinatura X não existe na Fila A

Tabela 43. Erros de MQSUB com diferentes identificadores de filas e combinações de assinaturas (continuação)

	Fila A **Assinatura X** **Fila B** **Nenhuma assinatura**	**Fila A** **Nenhuma assinatura** **Fila B** **Nenhuma assinatura**
Hobj para *Fila B* passado para MQSUB	**MQSO_CREATE** Cria nova assinatura X na Fila B **MQSO_RESUME** RC = 2019 – A fila de assinaturas não corresponde à manipulação de objetos de fila **MQSO_ALTER** Mova a assinatura X da Fila A para a Fila B	**MQSO_CREATE** Cria nova assinatura X na Fila B **MQSO_RESUME** RC = 2428 – a assinatura X não existe na Fila B **MQSO_ALTER** RC = 2428 – a assinatura X não existe na Fila B
MQHO_NONE passado para MQSUB	**MQSO_CREATE** RC = 2019 – Manipulação de objetos inválida: configure a sinalização MQSO_MANAGED para criar uma assinatura gerenciada e criar uma fila gerenciada **MQSO_RESUME** Continua a assinatura X na Fila A e retorna Hobj para a Fila A **MQSO_ALTER** Continua a assinatura X na Fila A, retorna Hobj para a Fila A e faz alterações permitidas	**MQSO_CREATE** RC = 2019 – Manipulação de objetos inválida: configure a sinalização MQSO_MANAGED para criar uma assinatura gerenciada e criar uma fila gerenciada **MQSO_RESUME** RC = 2428 – Nenhuma assinatura X **MQSO_ALTER** RC = 2019 – Manipulação de objetos inválida: nenhuma fila A ou B

Nota: O estilo de codificação compacto destina-se à capacidade de leitura e não ao uso na produção.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>

void inquireQname(MQHCONN HConn, MQHOBJ Hobj, MQCHAR48 qName);

int main(int argc, char **argv)
{
    MQCHAR48 topicNameDefault            = "STOCKS";
    char     topicStringDefault[]        = "IBM/PRICE";
    char     subscriptionNameDefault[]   = "IBMSTOCKPRICESUB";
    char     subscriptionQueueDefault[]  = "STOCKTICKER";
    char     publicationBuffer[101];      /* Allocate to receive messages     */
    char     resTopicStrBuffer[151];      /* Allocate to resolve topic string */
    MQCHAR48 qmName = "";                 /* Default queue manager            */
    MQCHAR48 qName = "";                  /* Allocate storage for MQINQ       */

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle                */
    MQHOBJ   Hobj = MQHO_NONE;            /* subscription queue handle        */
    MQHOBJ   Hsub = MQSO_NONE;            /* subscription handle              */
    MQLONG   CompCode = MQCC_OK;          /* completion code                  */
    MQLONG   Reason = MQRC_NONE;          /* reason code                      */
    MQLONG   messlen = 0;
    MQOD     od = {MQOD_DEFAULT};         /* Unmanaged subscription queue     */
    MQSD     sd = {MQSD_DEFAULT};         /* Subscription Descriptor          */
    MQMD     md = {MQMD_DEFAULT};         /* Message Descriptor               */
    MQGMO   gmo = {MQGMO_DEFAULT};        /* get message options              */
    MQLONG   sdOptions = MQSO_CREATE | MQSO_RESUME | MQSO_DURABLE |
MQSO_FAIL_IF_QUIESCING;

    char *   topicName = topicNameDefault;
    char *   topicString = topicStringDefault;
    char *   subscriptionName =  subscriptionNameDefault;
    char *   subscriptionQueue = subscriptionQueueDefault;
    char *   publication = publicationBuffer;
    char *   resTopicStr = resTopicStrBuffer;
    memset(resTopicStrBuffer, 0, sizeof(resTopicStrBuffer));
```

Figura 47. Assinante do MQ não gerenciado – parte 1: declarações.

```
    switch(argc){                   /* Replace defaults with args if provided */
default:
        switch((argv[5][0])) {
case('A'):  sdOptions =  MQSO_ALTER  | MQSO_DURABLE | MQSO_FAIL_IF_QUIESCING;
            break;
case('C'):  sdOptions =  MQSO_CREATE | MQSO_DURABLE | MQSO_FAIL_IF_QUIESCING;
            break;
case('R'):  sdOptions =  MQSO_RESUME | MQSO_DURABLE | MQSO_FAIL_IF_QUIESCING;
            break;
default:    ;
        }
case(5):
    if (strcmp(argv[4],"/")) /* "/" invalid = No subscription  */
        subscriptionQueue = argv[4];
    else {
        *subscriptionQueue = '\0';
        if (argc > 5) {
            if (argv[5][0] == 'C') {
                sdOptions = sdOptions + MQSO_MANAGED;
            }
        }
        else
            sdOptions = sdOptions + MQSO_MANAGED;
    }

case(4):
    if (strcmp(argv[3],"/")) /* "/" invalid = No subscription  */
        subscriptionName = argv[3];
    else {
        *subscriptionName = '\0';
        sdOptions = sdOptions - MQSO_DURABLE;
    }
case(3):
    if (strcmp(argv[2],"/")) /* "/" invalid = No topic string  */
        topicString = argv[2];
    else
        *topicString = '\0';
case(2):
    if (strcmp(argv[1],"/")) /* "/" invalid = No topic object  */
        topicName = argv[1];
    else
        *topicName = '\0';
case(1):
    sd.Options = sdOptions;
    printf("Optional parameters: "
    printf("topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|
R(esume)\n");
    printf("Values \"%-.48s\"  \"%s\" \"%s\"  \"%-.48s\" sd.Options=%d\n",
           topicName, topicString, subscriptionName, subscriptionQueue, sd.Options);
    }
```

Figura 48. Assinante do MQ não gerenciado - parte 2: manipulação de parâmetros.

Comentários adicionais sobre a manipulação de parâmetros neste exemplo são os seguintes:

switch((argv[5][0]))
Você tem a opção de inserir Alter | C reate | Resume no parâmetro 5 para testar o efeito de substituir parte da configuração da opção MQSUB usada por padrão no exemplo. A configuração padrão usada para o exemplo é MQSO_CREATE | MQSO_RESUME | MQSO_DURABLE

Nota: Configurar MQSO_ALTER ou MQSO_RESUME sem configurar MQSO_DURABLE é um erro e sd.SubName deve ser configurado e referir-se a uma assinatura que pode ser retomada ou alterada.

***subscriptionQueue = '\0';**
sdOptions = sdOptions + MQSO_MANAGED;
Se a fila de assinaturas padrão, STOCKTICKER, for substituída por uma sequência nula, então, contanto que MQSO_CREATE seja configurado, o exemplo configura a sinalização MQSO_MANAGED e cria uma fila de assinaturas dinâmicas. Se Alter or Resume forem configurados no quinto parâmetro, o comportamento do exemplo dependerá do valor de subscriptionName.

***subscriptionName = '\0';**
sdOptions = sdOptions - MQSO_DURABLE;

Se a assinatura padrão, IBMSTOCKPRICESUB , for substituída por uma sequência nula, o exemplo removerá a sinalização MQSO_DURABLE . Se você executar o exemplo fornecendo os valores padrão para os outros parâmetros, uma assinatura temporária adicional destinada a STOCKTICKER será criada e receberá publicações duplicadas. Na próxima vez que executar o exemplo, sem nenhum parâmetro, você receberá somente uma publicação novamente.

```
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        if (strlen(subscriptionQueue)) {
            strncpy(od.ObjectName, subscriptionQueue, MQ_Q_NAME_LENGTH);
            MQOPEN(Hconn, &od, MQOO_INPUT_AS_Q_DEF | MQOO_FAIL_IF_QUIESCING | MQOO_INQUIRE,
                   &Hobj, &CompCode, &Reason);
            if (CompCode != MQCC_OK) break;
        }
        strncpy(sd.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
        sd.ObjectString.VSPtr = topicString;
        sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
        sd.SubName.VSPtr = subscriptionName;
        sd.SubName.VSLength = MQVS_NULL_TERMINATED;
        sd.ResObjectString.VSPtr = resTopicStr;
        sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
        MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT | MQGMO_CONVERT;
        gmo.WaitInterval = 10000;
        gmo.MatchOptions = MQMO_MATCH_CORREL_ID;
        memcpy(md.CorrelId, sd.SubCorrelId, MQ_CORREL_ID_LENGTH);
        inquireQname(Hconn, Hobj, qName);
        printf("Waiting %d seconds for publications matching \"%s\" from %-0.48s\n",
               gmo.WaitInterval/1000, resTopicStr, qName);
        do {
            memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
            memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
            md.Encoding       = MQENC_NATIVE;
            md.CodedCharSetId = MQCCSI_Q_MGR;
            MQGET(Hconn, Hobj, &md, &gmo, sizeof(publication), publication, &messlen,
&CompCode, &Reason);
            if (Reason == MQRC_NONE)
                printf("Received publication \"%s\"\n", publication);
        }
        while (CompCode == MQCC_OK);
        if (CompCode != MQCC_OK && Reason != MQRC_NO_MSG_AVAILABLE) break;
        MQCLOSE(Hconn, &Hsub, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
void inquireQname(MQHCONN Hconn, MQHOBJ Hobj, MQCHAR48 qName) {
#define _selectors 1
#define _intAttrs 1

    MQLONG select[_selectors] = {MQCA_Q_NAME}; /* Array of attribute selectors    */
    MQLONG intAttrs[_intAttrs];            /* Array of integer attributes            */
    MQLONG CompCode, Reason;
    MQINQ(Hconn, Hobj, _selectors, select, _intAttrs, intAttrs, MQ_Q_NAME_LENGTH, qName,
&CompCode, &Reason);
    if (CompCode != MQCC_OK) {
        printf("MQINQ failed with Condition code %d and Reason %d\n", CompCode, Reason);
        strncpy(qName, "unknown queue", MQ_Q_NAME_LENGTH);
    }
    return;
}
```

Figura 49. Assinante do MQ não gerenciado – parte 3: corpo do código.

Comentários adicionais sobre o código neste exemplo são os seguintes:

if (strlen(subscriptionQueue))
Se não houver nome da fila de assinaturas, o exemplo usará MQHO_NONE como o valor de Hobj.

MQOPEN(...);
A fila de assinaturas é aberta e o identificador de filas é salvo em Hobj

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
A assinatura é aberta usando Hobj passado de MQOPEN (ou MQHO_NONE, se não houver nenhum nome da fila de assinatura). Uma fila não gerenciada pode ser continuada sem ser aberta explicitamente com um MQOPEN.

MQCLOSE(Hconn, &Hsub, MQCO_NONE, &CompCode, &Reason);
A assinatura é fechada usando o identificador de assinatura. Dependendo se a assinatura é durável ou não, a assinatura é fechada com um MQCO_KEEP_SUB ou MQCO_REMOVE_SUBimplícito É possível fechar uma assinatura durável com MQCO_REMOVE_SUB, mas *não é possível* fechar uma assinatura não durável com MQCO_KEEP_SUB. A ação de MQCO_REMOVE_SUB é remover a assinatura que para quaisquer publicações adicionais que estejam sendo enviadas para a fila de assinaturas

MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
Nenhuma ação especial será tomada se a assinatura for não gerenciada. Se a fila for gerenciada e a assinatura fechada com um MQCO_REMOVE_SUB explícito ou implícito, todas as publicações serão eliminadas da fila e a fila será excluída neste ponto.

gmo.MatchOptions = MQMO_MATCH_CORREL_ID;
memcpy(md.CorrelId, sd.SubCorrelId, MQ_CORREL_ID_LENGTH);
Assegure-se de que as mensagens recebidas sejam aquelas para a nossa assinatura.

Resultados do exemplo ilustram aspectos de publicar/assinar:

No Figura 50 na página 305 , o exemplo inicia publicando 130 no tópico NYSE/IBM/PRICE .

```
W:\Subscribe3\Debug>..\..\Publish2\Debug\publishstock
Provide parameters:  TopicObject TopicString Publication
Publish "130" to topic "STOCKS" and topic string "IBM/PRICE"
Published "130" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0
```

Figura 50. Publicar 130 em NYSE/IBM/PRICE

Em Figura 51 na página 305, a execução do exemplo usando parâmetros padrão recebe a publicação retida 130. O objeto do tópico e a sequência de tópicos fornecidos são ignorados, conforme mostrado em Figura 55 na página 307. O objeto do tópico e a sequência de tópicos são sempre tomados do objeto de assinatura, quando um é fornecido, e a sequência de tópicos é imutável. O comportamento real do exemplo depende da escolha ou da combinação de MQSO_CREATE, MQSO_RESUMEe MQSO_ALTER Neste exemplo, MQSO_RESUME é a opção selecionada.

```
W:\Subscribe3\Debug>solution3
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8206
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0
```

Figura 51. Receber a publicação retida

Em (Figura 52 na página 306) nenhuma publicação é recebida, porque a assinatura durável já recebeu a publicação retida. Neste exemplo, a assinatura é continuada fornecendo somente o nome da assinatura sem o nome da fila. Se o nome da fila foi fornecido, a fila poderá ser aberta primeiro e o identificador passado para MQSUB.

Nota: O erro 2038 de MQINQ é devido ao MQOPEN implícito de STOCKTICKER por MQSUB não incluindo a opção MQOO_INQUIRE .. Evite o código de retorno 2038 de MQINQ abrindo a fila explicitamente.

```
W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE IBMSTOCKPRICESUB / Resume
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" "IBMSTOCKPRICESUB"  "" sd.Options=8204
MQINQ failed with Condition code 2 and Reason 2038
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from unknown queue
Completion code 0 and Return code 0
```

Figura 52. Continuar assinatura

No Figura 53 na página 306, o exemplo cria uma assinatura não gerenciada não durável usando STOCKTICKER como o destino. Como esta é uma nova assinatura, ela recebe a publicação retida.

```
W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE / STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" ""  "STOCKTICKER" sd.Options=8194
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0
```

Figura 53. Receber publicação retida com a nova assinatura não durável não gerenciada

No Figura 54 na página 306, para demonstrar a sobreposição de assinaturas, outra publicação é enviada, mudando a publicação retida. Em seguida, uma nova assinatura não gerenciada, não durável é criada não fornecendo um nome de assinatura. A publicação retida é recebida duas vezes, uma para a nova assinatura e uma para a assinatura IBMSTOCKPRICESUB durável que ainda está ativa na fila STOCKTICKER. O exemplo é uma ilustração de que é a fila que tem assinaturas e não o aplicativo. Embora não fazendo referência à assinatura IBMSTOCKPRICESUB nesta chamada do aplicativo, o aplicativo recebe a publicação duas vezes: uma da assinatura durável que foi criada administrativamente e uma da assinatura não durável criada pelo próprio aplicativo.

```
W:\Subscribe3\Debug>..\..\Publish2\Debug\publishstock
Provide parameters:  TopicObject TopicString Publication
Publish "130" to topic "STOCKS" and topic string "IBM/PRICE"
Published "130" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0

W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE / STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS"  "IBM/PRICE" ""  "STOCKTICKER" sd.Options=8194
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Received publication "130"
Completion code 0 and Return code 0
```

Figura 54. Assinaturas sobrepostas

No Figura 55 na página 307, o exemplo demonstra que fornecer uma nova sequência de tópicos e uma assinatura existente não resulta em uma assinatura mudada.

1. No primeiro caso, `Resume` continua a assinatura existente, como você poderia esperar e ignora a sequência de tópicos mudada.
2. No segundo caso, `Alter` causa um erro `RC = 2510, Topic not alterable`.
3. No terceiro exemplo, `Create` causa um erro `RC = 2432, Sub already exists`.

```
W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Resume
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values ""  "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8204
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0

W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Alter
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values ""  "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8201
Completion code 2 and Return code 2510

W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values ""  "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB"  "STOCKTICKER" sd.Options=8202
Completion code 2 and Return code 2432
```

Figura 55. Tópicos de Assinatura não podem ser mudados

Conceitos relacionados

“Exemplo 1: consumidor de publicação do MQ” na página 290
O consumidor da publicação MQ é um consumidor de mensagens IBM WebSphere MQ que não assina tópicos em si.

“Exemplo 2: assinante do MQ gerenciado” na página 293
O assinante gerenciado do MQ é o padrão preferencial para a maioria dos aplicativos de assinante O exemplo requer *nenhuma* definição administrativa de filas, tópicos ou assinaturas

“Gravando aplicativos de publicador” na página 282
Comece a escrever aplicativos do publicador estudando dois exemplos. O primeiro é modelado o mais próximo possível em um aplicativo ponto a ponto colocando mensagens em uma fila e o segundo demonstra a criação de tópicos dinamicamente - um padrão mais comum para aplicativos publicadores.

Ciclos de vida de publicar/assinar

Considere os ciclos de vida de tópicos, assinaturas, assinantes, publicações, editores e filas ao projetar aplicativos de publicar/assinar.

O ciclo de vida de um objeto, como uma assinatura, começa com sua criação e termina com sua exclusão. Também pode incluir outros estados e mudanças pelos quais passa, como suspensão temporária, ter tópicos pais e filhos, expiração e exclusão.

Tradicionalmente, objetos do WebSphere MQ , como filas, são criados administrativamente ou por programas administrativos usando Programmable Command Format (PCF). Publicar/assinar é diferente, pois fornece os verbos de API `MQSUB` e `MQCLOSE` para criar e excluir assinaturas, tendo o conceito de assinaturas gerenciadas que não apenas cria e exclui filas, mas também limpa mensagens não consumidas e tendo associações entre objetos de tópicos criados administrativamente e sequências de tópicos criadas programaticamente ou administrativamente.

Essa riqueza funcional atende uma ampla variedade de requisitos de publicar/assinar e também simplifica o design de alguns padrões comuns de aplicativo de publicar/assinar. Assinaturas gerenciadas, por exemplo, simplificam tanto a programação e a administração de uma assinatura destinada a durar apenas o mesmo tempo que o programa que a criou. Assinaturas não gerenciadas simplificam a programação quando houver uma conexão mais livre entre assinatura e consumo de publicações. Assinaturas criadas centralmente são úteis quando o padrão for de roteamento de tráfego de publicação para consumidores com base em um modelo centralizado de controle, por exemplo, envio de informações de voo para portões automatizados, enquanto que as assinaturas programaticamente criadas podem ser usadas se a equipe do portão for responsável por assinar os registros de passageiros do voo, inserindo um número de voo em um portão.

Neste último exemplo, uma assinatura durável gerenciada pode ser apropriada: gerenciada, pois as assinaturas estão sendo criadas com muita frequência e possuem um terminal claro quando o portão se fecha e a assinatura pode ser removida programaticamente; durável, para evitar perder um registro de passageiro devido ao programa de assinantes do portão diminuir por uma razão ou outra[2]Para iniciar a publicação de registros de passageiros do portão, um possível design seria para o aplicativo do portão assinar tanto os registros do passageiro usando o número do portão quanto publicar o evento

[2] O publicar deve enviar os registros de passageiros como mensagens persistentes para evitar outras possíveis falhas, é claro.

de abertura do portão usando o número do portão. O publicador responde ao evento de abertura do portão publicando os registros de passageiros - que podem, então, também ir para outras partes interessadas, como faturamento, para registrar que o voo está ocorrendo e para atendimento ao cliente, para notificações de texto em celulares dos passageiros sobre o número do portão.

A assinatura gerenciada centralmente pode usar um modelo durável não gerenciado, roteando de listas de passageiros para o portão usando uma fila predefinida para cada portão.

Os três exemplos a seguir de ciclos de vida de publicar/assinar ilustram como assinantes gerenciados não duráveis, gerenciados duráveis e não gerenciados duráveis interagirem com assinaturas, tópicos, filas, publicadores e com o gerenciador de filas, e como as responsabilidades podem ser divididas entre a administração e os programas de assinante.

Assinante gerenciado não durável

Figura 56 na página 309 mostra um aplicativo criando uma assinatura gerenciada não durável, obtendo duas mensagens publicadas no tópico identificado na assinatura e finalizando. As interações com rótulo em uma fonte cinza em itálico com setas pontilhadas são implícitas.

Há alguns pontos a serem observados.

1. O aplicativo cria uma assinatura em um tópico para o qual já foi publicado duas vezes. Quando o assinante recebe sua primeira publicação, recebe a *segunda* publicação que é a publicação retida atualmente.
2. O gerenciador de filas cria uma fila de assinatura temporária, assim como cria uma assinatura para o tópico.
3. A assinatura tem uma expiração. Quando a assinatura expira, mais nenhuma publicação no tópico será enviada para essa assinatura, mas o assinante continua a obter mensagens publicadas antes da assinatura expirar. A expiração da publicação não é afetada pela expiração da assinatura.
4. A quarta publicação não é colocada na fila de assinatura e, consequentemente, o último `MQGET` não retorna uma publicação.
5. Embora o assinante fecha sua assinatura, ela não fecha sua conexão com a fila ou com o gerenciador de filas.
6. O gerenciador de filas é limpo pouco depois da finalização do aplicativo. Como a assinatura é gerenciada e não durável, a fila de assinaturas é excluída.

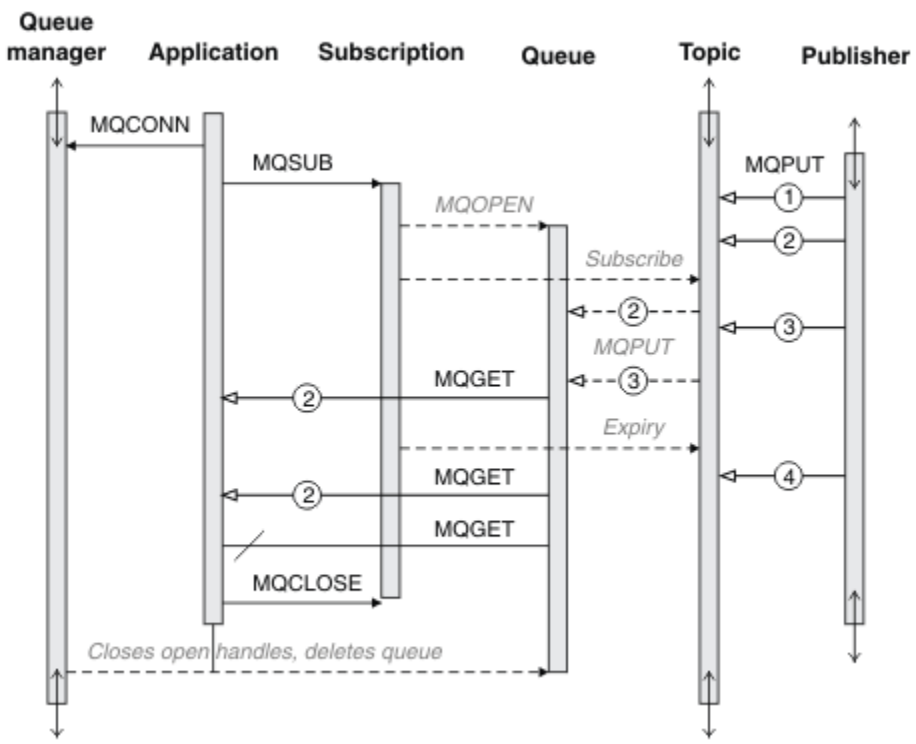


Figura 56. Linhas de vida do assinante gerenciado não durável

Assinante gerenciado durável

O assinante gerenciado durável leva o exemplo anterior um passo à frente e mostra uma assinatura gerenciada que sobrevive à finalização e à reinicialização do aplicativo de assinatura.

Há alguns novos pontos a serem observados.

1. Neste exemplo, diferentemente do anterior, o tópico de publicação não existia antes de ser definido na assinatura.
2. A primeira vez que o assinante é finalizado, ele fecha a assinatura com a opção `MQCO_KEEP_SUB`. Esse é o comportamento padrão para fechar implicitamente uma assinatura gerenciada durável.
3. Quando o assinante continua a assinatura, a fila de assinaturas é reaberta.
4. A nova publicação 2, colocada na fila antes de ser reaberta, está disponível para `MQGET`, mesmo após a assinatura ter sido removida.

   Embora a assinatura seja durável, o assinante recebe de forma confiável todas as mensagens enviadas pelo publicador somente se *ambos* forem verdadeiros, a assinatura durável e as mensagens persistentes. Persistência de mensagem depende da configuração do campo `Persistent` no `MQMD` da mensagem enviada pelo publicador. Um assinante não tem controle sobre isso.
5. Fechar a assinatura com a sinalização `MQCO_REMOVE_SUB` remove a assinatura, parando quaisquer publicações adicionais que estiverem sendo colocadas na fila de assinaturas. Quando a fila de assinatura é fechada, então, o gerenciador de filas remove a publicação 3 não lida e, em seguida, exclui a fila. A ação é equivalente a excluir a assinatura administrativamente.

   Nota: Não exclua a fila manualmente ou emita `MQCLOSE` com a opção `MQCO_DELETE` ou `MQCO_PURGE_DELETE`. Os detalhes de implementação visíveis de uma subscrição gerenciada não fazem parte da interface suportada do WebSphere MQ O gerenciador de filas não pode gerenciar uma assinatura de forma confiável a menos que tenha controle completo.

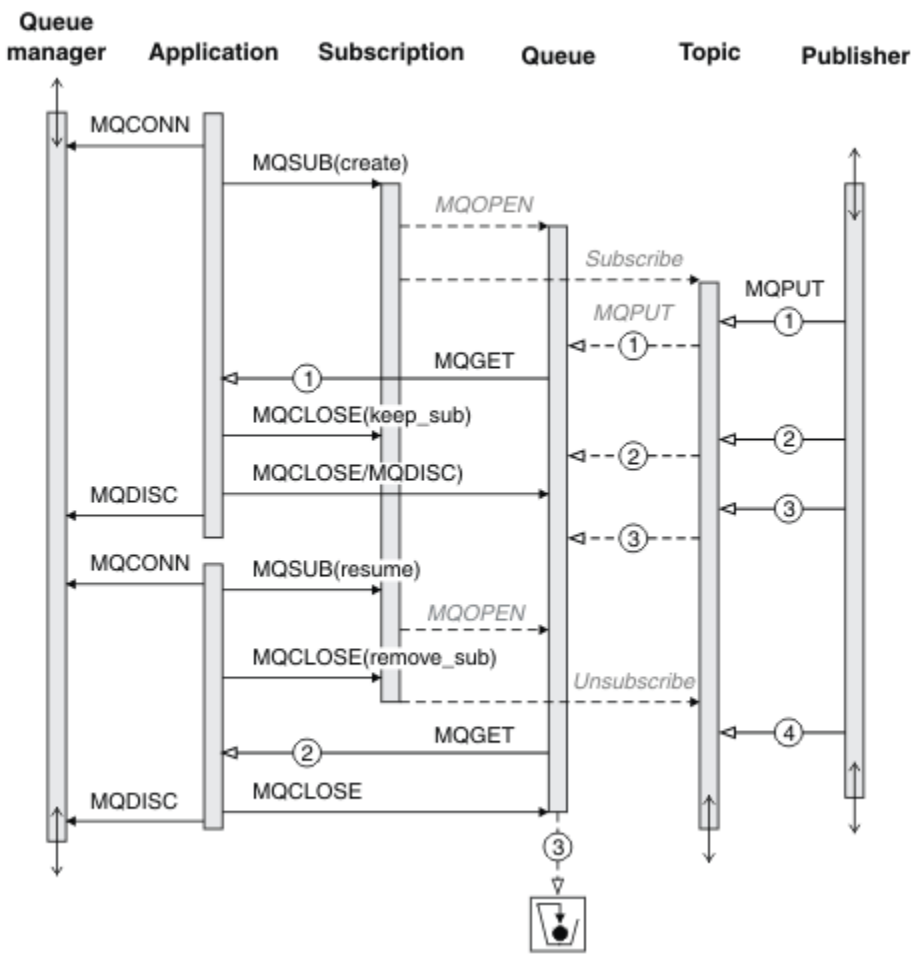


Figura 57. Linhas de vida do assinante gerenciado durável

Assinante não gerenciado durável

Um administrador é incluído no terceiro exemplo: o assinante não gerenciado durável. Este é um bom exemplo para mostrar como o administrador pode interagir com um aplicativo de publicar/assinar.

Os pontos a serem observados são listados.

1. O publicador coloca uma mensagem, 1, em um tópico que posteriormente se tornará associado ao objeto do tópico usado para assinatura. O objeto do tópico define uma sequência de tópicos que corresponde ao tópico ao qual foi publicado usando caracteres curinga.
2. O tópico possui uma publicação retida.
3. O administrador cria um objeto do tópico, uma fila e uma assinatura. O objeto do tópico e a fila precisam ser definidos antes da assinatura.
4. O aplicativo abre a fila associada à assinatura e passa a `MQSUB` o identificador da fila. Poderia, como alternativa, simplesmente abrir a assinatura, passando a ela o identificador de fila `MQHO_NONE`. O inverso não é verdadeiro, não pode continuar uma assinatura passando a ela somente o identificador de fila sem um nome de assinatura – uma fila pode ter várias assinaturas.
5. O aplicativo abre a assinatura usando a opção `MQSO_RESUME`, embora seja a primeira vez que abriu a assinatura. Está continuando uma assinatura criada administrativamente.
6. O assinante recebe a publicação retida, 1. A publicação 2, embora publicada antes que quaisquer publicações tenham sido recebidas pelo assinante, foi publicada após a assinatura ser iniciada e é a segunda publicação na fila de assinaturas.

Nota: Se a publicação retida não for publicada como uma mensagem persistente, então, será perdida após a reinicialização do gerenciador de filas.

7. Neste exemplo, a assinatura é durável. É possível que um programa crie uma assinatura não gerenciada durável; deve estar óbvio que isso não é algo que um administrador possa fazer.
8. O efeito da opção `MQCO_REMOVE_SUB` no fechamento da assinatura é remover a assinatura como se o administrador a tivesse excluído. Isso para quaisquer publicações adicionais que estiverem sendo enviadas para a fila, mas não afeta as publicações que já estão na fila, mesmo quando a fila for fechada, diferentemente de uma assinatura *gerenciada* durável.
9. O administrador posteriormente exclui a mensagem restante, 3, e exclui a fila.

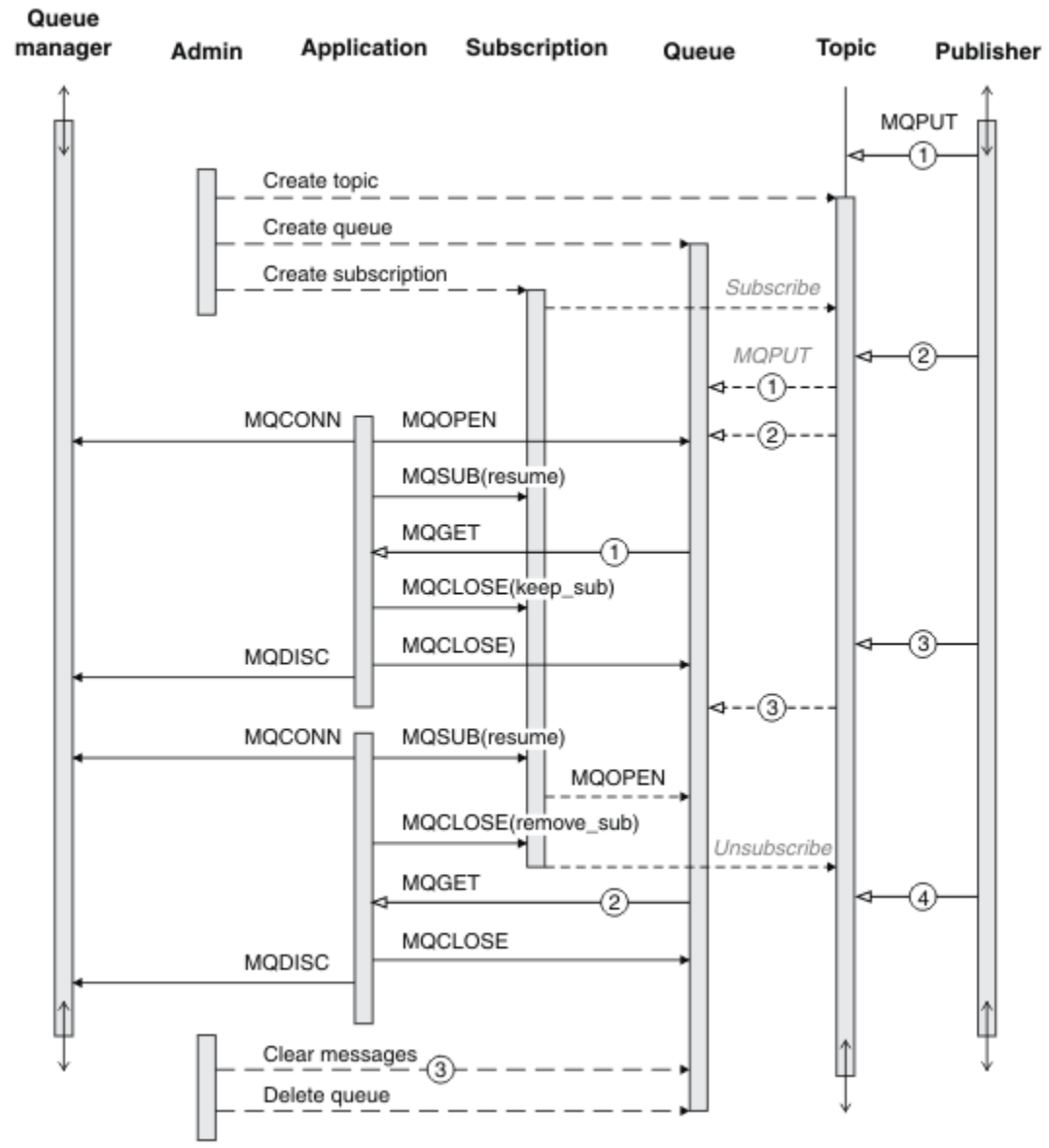


Figura 58. Linhas de vida do assinante não gerenciado durável

Um padrão normal para uma assinatura não gerenciada é a manutenção de filas e assinaturas ser executada pelo administrador. Geralmente, não se tentaria emular o comportamento de um assinante gerenciado e organizar as filas e assinaturas programaticamente no código do aplicativo. Se achar necessário escrever lógica de gerenciamento, questione se é possível obter os mesmos resultados usando um padrão gerenciado. Não é fácil escrever código de gerenciamento fortemente sincronizado e completamente confiável. É mais fácil organizar posteriormente, seja manualmente ou usando

um programa de gerenciamento automatizado, quando for possível ter certeza de que mensagens, assinaturas e filas podem ser simplesmente excluídas, independentemente de seu estado.

Propriedades de mensagem de publicação/assinatura

Várias propriedades de mensagem relacionadas ao sistema de mensagens de publicação / assinatura do WebSphere MQ .

PubAccountingToken

Esse é o valor que estará no campo AccountingToken do Message Descriptor (MQMD) de todas as mensagens de publicação que correspondam a essa assinatura. AccountingToken faz parte do contexto de identidade da mensagem. Para obter mais informações sobre o contexto da mensagem, consulte “Contexto da mensagem” na página 39. Para obter mais informações sobre o campo AccountingToken no MQMD, consulte AccountingToken.

PubApplIdentityData

Esse é o valor que estará no campo ApplIdentityData do Message Descriptor (MQMD) de todas as mensagens de publicação que correspondam a essa assinatura. ApplIdentityData faz parte do contexto de identidade da mensagem. Para obter mais informações sobre o contexto da mensagem, consulte “Contexto da mensagem” na página 39. Para obter mais informações sobre o campo ApplIdentityData no MQMD, consulte ApplIdentityData.

Se a opção MQSO_SET_IDENTITY_CONTEXT não for especificada, ApplIdentityData que será configurado em cada mensagem publicada para essa assinatura ficará em branco, como informações de contexto padrão.

Se a opção MQSO_SET_IDENTITY_CONTEXT for especificada, PubApplIdentityData está sendo gerado pelo usuário e esse campo é um campo de entrada que contém ApplIdentityData a ser configurado em cada publicação para essa assinatura.

PubPriority

Esse é o valor que estará no campo Prioridade do Message Descriptor (MQMD) de todas as mensagens de publicação que correspondam a essa assinatura. Para obter mais informações sobre o campo Prioridade no MQMD, consulte Prioridade.

O valor deve ser maior ou igual a zero; zero é a prioridade mais baixa. Os valores especiais a seguir também podem ser usados:

- MQPRI_PRIORITY_AS_Q_DEF – Quando uma fila de assinatura for fornecida no campo Hobj na chamada MQSUB e não for uma manipulação gerenciada, a prioridade da mensagem será obtida a partir do atributo DefProirity dessa fila. Se a fila assim identificada for uma fila de clusters ou houver mais de uma definição no caminho de resolução do nome da fila, a prioridade será determinada quando a mensagem de publicação for colocada na fila, conforme descrito para Prioridade no MQMD. Se a chamada MQSUB usar uma manipulação gerenciada, a prioridade para essa mensagem será obtida do atributo DefPriority da fila modelo associada ao tópico assinado.
- MQPRI_PRIORITY_AS_PUBLISHED – A prioridade para a mensagem é a prioridade da publicação original. Esse é o valor inicial desse campo.

SubCorrelId

Atenção: Um identificador de correlação só pode ser transmitido entre os gerenciadores de filas em um cluster de publicação/assinatura, não uma hierarquia.

Todas as publicações enviadas para corresponderem a essa assinatura conterão esse identificador de correlação no descritor de mensagens. Se diversas assinaturas usarem a mesma fila para obter suas publicações, usar MQGET por ID de correlação permitirá que somente publicações de uma assinatura

específica sejam obtidas. Esse identificador de correlação pode ser gerado pelo gerenciador de filas ou pelo usuário.

Se a opção MQSO_SET_CORREL_ID não for especificada, o identificador de correlação será gerado pelo gerenciador de filas e esse campo será um campo de saída que contém o identificador de correlação que será configurado em cada mensagem publicada para essa assinatura.

Se a opção MQSO_SET_CORREL_ID for especificada, o identificador de correlação está sendo gerado pelo usuário e esse campo é um campo de entrada que contém o identificador de correlação a ser configurado em cada publicação para essa assinatura. Nesse caso, se o campo contiver MQCI_NONE, o identificador de correlação que será configurado em cada mensagem publicada para essa assinatura será o identificador de correlação criado pelo put original da mensagem.

Se a opção MQSO_GROUP_SUB for especificada e o identificador de correlação especificado for o mesmo que uma assinatura agrupada existente usando a mesma fila e uma sequência de tópicos sobreposta, somente a assinatura mais significativa do grupo será fornecida com uma cópia da publicação.

SubUserData

Esses são os dados do usuário de assinatura. Os dados fornecidos na assinatura nesse campo serão incluídos como a propriedade de mensagem MQSubUserData de cada publicação enviada para essa assinatura.

Propriedades de publicação

Tabela 44 na página 313 lista as propriedades de publicação que são fornecidas com uma mensagem de publicação.

É possível acessar essas propriedades diretamente da pasta **MQRFH2** ou recuperá-las usando MQINQMP. MQINQMP aceita o nome da propriedade ou o nome **MQRFH2** como o nome da propriedade a consultar.

Tabela 44. Propriedades de publicação

Nome da Propriedade	Nome do MQRFH2	Tipo	Descrição
MQTopicString	mqps.Top	MQTYPE_STRING	Cadeia do tópico
MQSubUserData	mqps.Sud	MQTYPE_STRING	Dados do usuário do assinante
MQIsRetained	mqps.Ret	MQTYPE_BOOLEAN	Publicação retida
MQPubOptions	mqps.Pub	MQTYPE_INT32	Opções de publicação
MQPubLevel	mqps.Pbl	MQTYPE_INT32	Nível de publicação
MQPubTime	mqpse.Pts	MQTYPE_STRING	Hora da publicação
MQPubSeqNum	mqpse.Seq	MQTYPE_INT32	Número de sequência da publicação
MQPubStrIntData	mqpse.Sid	MQTYPE_STRING	Dados de sequência/ número inteiro incluídos pelo publicador
MQPubFormat	mqpse.Pfmt	MQTYPE_INT32	Formato da mensagem: MQRFH1 MQRFH2 PCF

Ordenação de mensagens

Para um tópico específico, as mensagens são publicadas pelo gerenciador de filas na mesma ordem que são recebidas dos aplicativos de publicação (sujeitas à reorganização com base na prioridade da mensagem).

Ordenação de mensagens normalmente significa que cada assinante recebe mensagens de um gerenciador de filas específico, em um tópico específico, de um publicador específico na ordem que são publicadas por esse publicador.

No entanto, como com todas as mensagens do WebSphere MQ , é possível que mensagens, ocasionalmente, sejam entregues fora de ordem. Isso pode ocorrer nas situações a seguir:

- Se um link na rede ficar inativo e as mensagens subsequentes forem roteadas novamente por outro link
- Se uma fila ficar temporariamente cheia ou inibida para put, de forma que uma mensagem seja colocada em uma fila de mensagens não entregues e, portanto, atrasada, enquanto as mensagens subsequentes passam direto.
- Se o administrador excluir um gerenciador de filas quando publicadores e assinantes ainda estiverem em operação, fazendo com que mensagens enfileiradas sejam colocadas na fila de mensagens não entregues e assinaturas sejam interrompidas.

Se estas circunstâncias não puderem ocorrer, as publicações sempre serão entregues em ordem.

Nota: Não é possível usar mensagens agrupadas ou segmentadas com Publicar/Assinar.

Interceptando publicações

É possível interceptar uma publicação, modificá-la e, em seguida, publicá-la novamente antes de atingir qualquer outro assinante.

Pode ser que deseje interceptar uma publicação antes de atingir um assinante para executar uma das ações a seguir:

- Anexar informações adicionais à mensagem
- Bloquear a mensagem
- Transformar a mensagem

É possível executar a mesma operação em cada mensagem ou variar a operação, dependendo da assinatura, da mensagem ou do cabeçalho da mensagem.

Referências relacionadas

MQ_PUBLISH_EXIT - saída Publish

Níveis de assinatura

Configure o nível de uma assinatura para interceptar uma publicação antes de atingir seus assinantes finais. Um assinante de interceptação assina em um nível de assinatura superior e publica novamente em um nível de publicação inferior. Construa uma cadeia de assinantes de interceptação para executar o processamento de mensagens em uma publicação antes que seja entregue aos assinantes finais.

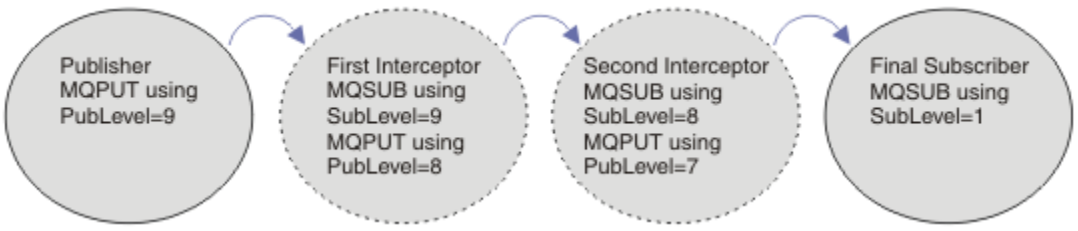


Figura 59. Sequência de assinantes de interceptação

Para interceptar uma publicação, use o atributo **MQSD** `SubLevel`. Após uma mensagem ter sido interceptada, ela pode ser transformada e republicada em uma publicação de nível inferior, mudando o

atributo **MQPMO** PubLevel. A mensagem então vai para os assinantes finais ou é interceptada novamente por um assinante intermediário no nível de assinatura inferior.

O assinante de interceptação geralmente transforma uma mensagem antes de publicá-la novamente. Uma sequência de assinantes de interceptação forma um fluxo de mensagens. Como alternativa, você não pode publicar novamente a publicação interceptada: assinantes em níveis inferiores de assinatura não receberiam a mensagem.

Assegure que o interceptador receba as publicações antes de quaisquer outros assinantes. Configure o nível de assinatura do interceptador para nível superior a dos outros assinantes. Por padrão, os assinantes têm um Subnível de 1. O valor mais alto é 9. Uma publicação deve começar com um PubLevel pelo menos tão alto quanto o mais alto SubLevel. Publique inicialmente com o PubLevel padrão de 9.

- Se você tiver um assinante de interceptação em um tópico, configure o SubLevel para 9.
- Para vários aplicativos de interceptação em um tópico, configure um SubLevel inferior para cada assinante de interceptação sucessivo.
- É possível implementar no máximo 8 aplicativos de interceptação, com níveis de assinatura de 9 a 2, inclusive. O destinatário final da mensagem tem um SubLevel iguala a 1.

O interceptador com o nível de assinatura mais alto que é igual ou menor que o PubLevel da publicação recebe a primeira publicação. Configure somente um assinante de interceptação para um tópico em um determinado nível de assinatura. Ter vários assinantes em um nível de assinatura específico resulta em várias cópias da publicação serem enviadas ao conjunto final de aplicativos de assinatura.

Um assinante com um SubLevel igual a 0 é usado como um depósito. Ele recebe a publicação se nenhum assinante final obtiver a mensagem. Um assinante com SubLevel igual a 0 pode ser usado para monitorar as publicações que nenhum outro assinante recebeu.

Programando um assinante de interceptação

Use as opções de assinatura descritas em Tabela 45 na página 315.

Tabela 45. Opções de assinatura para assinantes de interceptação

Opção de assinatura	**Notes**
MQSO_SET_CORREL_ID e SubCorrelId configurados para MQCI_NONE	Mantenha o CorrelId da publicação interceptada igual ao da publicação original. **Nota:** Não é possível passar o identificador de correlação de uma publicação em uma hierarquia. O campo é usado pelo gerenciador de filas.
PubPriority configurado para MQPRI_PRIORITY_AS_PUBLISHED	Mantenha a prioridade da publicação interceptada igual à da publicação original.

As opções em Tabela 45 na página 315 devem ser usadas por todos os assinantes de interceptação. O resultado é que o identificador de correlação e a prioridade da mensagem não são modificados a partir da configuração do publicador original.

Quando o assinante de interceptação tiver processado a publicação, ele publica novamente a mensagem para o mesmo tópico em um PubLevel um nível inferior ao SubLevel de sua própria assinatura. Se o assinante de interceptação tiver configurado um SubLevel igual a 9, ele publica novamente a mensagem com um PubLevel igual a 8.

Para publicar novamente a mensagem corretamente, várias partes de informações da publicação original são necessárias. Reutilize o mesmo **MQMD** que da mensagem original e configure MQPMO_PASS_ALL_CONTEXT para assegurar que todas as informações no **MQMD** sejam passadas ao próximo assinante. Copie os valores das propriedades da mensagem mostrados em Tabela 46 na página 316 nos campos correspondentes da mensagem publicada novamente. O assinante de interceptação

pode mudar esses valores. Use o operador OR para incluir valores adicionais no campo **MQPMO**.Opções , para combinar as opções de mensagem put..

Deve-se abrir a fila de publicação explicitamente em vez de usar uma fila de publicação gerenciada. Não é possível configurar MQSO_SET_CORREL_ID para uma fila gerenciada. Também não é possível configurar MQOO_SAVE_ALL_CONTEXT em uma fila gerenciada. Consulte os fragmentos de código listados em "Examples" na página 316.

Tabela 46. Valores de MQPUT para mensagens publicadas novamente

Publicar mensagem novamente usando MQPUT	**Informações na mensagem de publicação**
MQOD.ObjectString	Propriedade de mensagem MQTopicString
MQPMO.Options	Propriedade de mensagem MQPubOptions

O assinante final tem a opção de configurar suas opções de assinatura de forma diferente.
Por exemplo, pode configurar a prioridade da publicação explicitamente em vez de para MQPRI_PRIORITY_AS_PUBLISHED. As configurações de um assinante final afetam somente publicação do assinante de interceptação final na cadeia.

Publicações Retidas

Uma publicação retida deve ser preservada após ter sido interceptada, copiando as opções put-message originais na mensagem publicada novamente.

A opção MQPMO_RETAIN é configurada pelo publicador. Cada assinante de interceptação deve transferir MQPubOptions para as opções put-message da mensagem publicada novamente conforme mostrado em Tabela 46 na página 316. Copiar as opções put-message preserva as opções configuradas pelo publicador original, incluindo se deseja reter a publicação.

Quando uma publicação conclui sua passagem pela cadeia de assinantes de interceptação e é entregue a assinantes finais, é finalmente retida. Novos assinantes, no SubLevel 1, que solicitam a publicação retida, recebem a mesma sem qualquer interceptação adicional. Os assinantes em um SubLevel maior que 1 não são enviadas à publicação retida. Como resultado, a publicação retida não é modificada pela cadeia de assinantes de interceptação uma segunda vez.

Examples

Os exemplos são fragmentos de código que podem ser combinados para construir um assinante de interceptação. O código é escrito para ser breve, em vez da qualidade da produção.

As diretivas do pré-processador em Figura 60 na página 316 definem as duas propriedades a serem extraídas a partir das mensagens da publicação necessárias pela chamada MQI MQINQMP.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
#define      MQPUBOPTIONS        (MQPTR)(char*) "MQPubOptions",\
                                 0,\
                                 12,\
                                 MQVS_NULL_TERMINATED,\
                                 MQCCSI_APPL
#define      MQTOPICSTRING       (MQPTR)(char*) "MQTopicString",\
                                 0,\
                                 13,\
                                 MQVS_NULL_TERMINATED,\
                                 MQCCSI_APPL
```

Figura 60. Diretivas do pré-processador

Figura 61 na página 317 lista as declarações usadas nos fragmentos de código. Exceto pelos termos destacados, as declarações são padrão para um aplicativo WebSphere MQ .

As opções Put e Get destacadas são inicializadas para passar todo o contexto. MQTOPICSTRING e MQPUBOPTIONS destacados são inicializadores de MQCHARV para nomes de propriedades definidos nas diretivas do pré-processador. Os nomes são passados para MQINQMP.

```
int main(int argc, char **argv) {
    MQLONG   Reason            = MQRC_NONE;
    MQLONG   CompCode          = MQCC_OK;
    MQHCONN  Hcon              = MQHC_UNUSABLE_HCONN;
    MQCHAR   QMName[49]        = " ";
    MQCMHO   CrtMsgHOpts       = {MQCMHO_DEFAULT};
    MQHMSG   Hmsg              = MQHM_NONE;
    MQMD     md                = {MQMD_DEFAULT};
    MQHOBJ   gHobj             = MQHO_NONE;
    MQOD     getOD             = {MQOD_DEFAULT};
    MQGMO    gmo               = {MQGMO_DEFAULT};
    MQLONG   GO_Options        = MQOO_INPUT_AS_Q_DEF
                               | MQOO_FAIL_IF_QUIESCING
                               | MQOO_SAVE_ALL_CONTEXT;
    MQLONG   GC_Options        = MQCO_DELETE_PURGE;
    MQHOBJ   Hsub              = MQHO_NONE;
    MQSD     sd                = {MQSD_DEFAULT};
    MQLONG   SC_Options        = MQCO_NONE;
    MQHOBJ   pHobj             = MQHO_NONE;
    MQOD     putOD             = {MQOD_DEFAULT};
    MQLONG   PO_Options        = MQOO_OUTPUT
                               | MQOO_FAIL_IF_QUIESCING
                               | MQOO_PASS_ALL_CONTEXT;
    MQLONG   PC_Options        = MQCO_NONE;
    MQPMO    pmo               = {MQPMO_DEFAULT};
    MQIMPO   InqPropOpts       = {MQIMPO_DEFAULT};
    MQPD     PropDesc          = {MQPD_DEFAULT};
    MQLONG   Type              = MQTYPE_AS_SET;
    MQCHARV  TopStrProp        = {MQTOPICSTRING};
    MQCHARV  PubOptProp        = {MQPUBOPTIONS};
    MQLONG   DataLength        = 0;
    MQBYTE   buffer[256]       = "";
    MQLONG   buflen            = sizeof(buffer) - 1;
    MQLONG   messlen           = 0;
    char     TopStrBuf[256]    = "Initial value";
    int      i                 = 0;
```

Figura 61. Declarações

Inicializações que não são facilmente executadas nas declarações são mostradas em Figura 62 na página 318. Os valores destacados requerem explicação.

SYSTEM.NDURABLE.MODEL.QUEUE
Neste exemplo, em vez de usar MQSUB para abrir uma assinatura não durável gerenciada, a fila modelo, SYSTEM.NDURABLE.MODEL.QUEUE, é usada para criar uma fila dinâmica temporária. Seu identificador é passado para MQSUB. Ao abrir a fila diretamente, você é capaz de salvar todo o contexto da mensagem e configurar a opção de assinatura, MQSO_SET_CORREL_ID.

MQGMO_CURRENT_VERSION
É importante usar a versão atual da maioria das estruturas WebSphere MQ . Os campos, como gmo.MsgHandle, estão disponíveis somente na versão mais recente das estruturas de controle.

MQGMO_PROPERTIES_IN_HANDLE
As opções da sequência de tópicos e put message configuradas na publicação original devem ser recuperadas pelo assinante de interceptação usando as propriedades de mensagem. Uma alternativa seria ler a estrutura **MQRFH2** na mensagem diretamente.

MQSO_SET_CORREL_ID
Use MQSO_SET_CORREL_ID em combinação com

```
memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
```

O efeito dessas opções é passar adiante o identificador de correlação. O identificador de correlação configurado pelo publicador original é colocado no campo do identificador de correlação da publicação recebida pelo assinante de interceptação. Cada assinante de interceptação passa adiante o mesmo identificador de correlação. O assinante final tem, então, a opção de receber o mesmo identificador de correlação.

Nota: Se a publicação for passada por meio de uma hierarquia de publicar/assinar, o identificador de correlação nunca será retido.

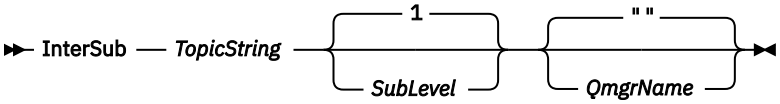
MQPRI_PRIORITY_AS_PUBLISHED
A publicação é colocada na fila de publicação com a mesma prioridade de mensagem que foi publicada.

```
    strncpy(getOD.ObjectName,    "SYSTEM.NDURABLE.MODEL.QUEUE",
            sizeof(getOD.ObjectName));
    gmo.Version               = MQGMO_VERSION_4;
    gmo.Options               = MQGMO_WAIT
                              | MQGMO_PROPERTIES_IN_HANDLE
                              | MQGMO_CONVERT;
    gmo.WaitInterval          = 30000;
    sd.Options                = MQSO_CREATE
                              | MQSO_FAIL_IF_QUIESCING
                              | MQSO_SET_CORREL_ID;
    sd.PubPriority            = MQPRI_PRIORITY_AS_PUBLISHED;
    sd.Version                = MQSD_VERSION_1;
    memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
    putOD.ObjectType            = MQOT_TOPIC;
    putOD.ObjectString.VSPtr    = &TopStrBuf;
    putOD.ObjectString.VSBufSize = sizeof(TopStrBuf);
    putOD.ObjectString.VSLength  = MQVS_NULL_TERMINATED;
    putOD.ObjectString.VSCCSID   = MQCCSI_APPL;
    putOD.Version               = MQOD_VERSION_4;
    pmo.Version               = MQPMO_VERSION_3;
```

Figura 62. Inicializações

Figura 63 na página 319 mostra o fragmento de código para ler os parâmetros da linha de comandos, concluir a inicialização e criar a assinatura de interceptação.

Execute o programa com o comando:



Para tornar a manipulação de erros o mais não obstrutiva possível, o código de razão de cada chamada MQI é armazenado em um elemento de matriz diferente. Após cada chamada, o código de conclusão é testado e, se o valor for MQCC_FAIL, o controle sai do bloco de código do `{ } while(0)`.

As duas linhas de código que valem a pena ser observadas são:

pmo.PubLevel = sd.SubLevel - 1;
Configura o nível de publicação da mensagem publicada novamente para um inferior ao nível de assinatura do assinante de interceptação.

gmo.MsgHandle = Hmsg;
Fornece um identificador de mensagem para MQGET para retornar as propriedades da mensagem.

```
do {
   printf("Intercepting subscriber start\n");
   if (argc < 2) {
       printf("Required parameter missing - topic string\n");
       exit(99);
   } else {
       sd.ObjectString.VSPtr    = argv[1];
       sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
       printf("TopicString = %s\n", sd.ObjectString.VSPtr);
   }
   if (argc > 2) {
       sd.SubLevel  = atoi(argv[2]);
       pmo.PubLevel = sd.SubLevel - 1;
       printf("SubLevel is %d, PubLevel is %d\n", sd.SubLevel, pmo.PubLevel);
   }
   if (argc > 3)
       strncpy(QMName, argv[3], sizeof(QMName));
   MQCONN(QMName, &Hcon, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQOPEN(Hcon, &getOD, GO_Options, &gHobj, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQSUB(Hcon, &sd, &gHobj, &Hsub, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQCRTMH(Hcon, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   gmo.MsgHandle = Hmsg;
```

Figura 63. Preparando para interceptar publicações

O fragmento de código principal, Figura 64 na página 320, recebe mensagens da fila de publicação. Ele consulta as propriedades de mensagem e publica novamente as mensagens usando a sequência de tópicos e as propriedades **MQPMO**.`option` originais da publicação.

Neste exemplo, nenhuma transformação é executada na publicação. A sequência de tópicos da publicação publicada novamente sempre corresponde à sequência de tópicos que o assinante de interceptação assinou. Se o assinante de interceptação for responsável por interceptar várias assinaturas enviadas para a mesma fila de publicação, pode ser necessário consultar a sequência de tópicos para distinguir as publicações que correspondem a assinaturas diferentes.

As chamadas para `MQINQMP` estão destacadas. As propriedades das opções de sequência de tópicos e put message de publicação são gravadas diretamente nas estruturas de controle de saída. A única razão para alterar o campo de comprimento MQCHARV de `putOD.ObjectString` de um comprimento explícito para uma sequência finalizada em nulo é para usar `printf` para saída da sequência.

```
while (CompCode != MQCC_FAILED) {
   memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
   memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
   md.Encoding       = MQENC_NATIVE;
   md.CodedCharSetId = MQCCSI_Q_MGR;
   printf("MQGET : %d seconds wait time\n", gmo.WaitInterval/1000);
   MQGET(Hcon, gHobj, &md, &gmo, buflen, buffer, &messlen,
       &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   buffer[messlen] = '\0';
   MQINQMP(Hcon, Hmsg,    &InqPropOpts, &TopStrProp, &PropDesc, &Type,
       putOD.ObjectString.VSBufSize, putOD.ObjectString.VSPtr,
       &(putOD.ObjectString.VSLength), &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   memset((void *)((MQLONG)(putOD.ObjectString.VSPtr)
       + putOD.ObjectString.VSLength),'\0',1);
   putOD.ObjectString.VSLength = MQVS_NULL_TERMINATED;
   MQINQMP(Hcon, Hmsg, &InqPropOpts, &PubOptProp, &PropDesc, &Type,
       sizeof(pmo.Options), &(pmo.Options), &DataLength,
       &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQOPEN(Hcon, &putOD, PO_Options, &pHobj, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   printf("Republish message <%s> on topic <%s> with options %d\n",
       buffer, putOD.ObjectString.VSPtr, pmo.Options);
   MQPUT(Hcon, pHobj, &md, &pmo, messlen, buffer, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQCLOSE(Hcon, &pHobj, PC_Options, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
   break;
}
```

Figura 64. Interceptar publicação e publicar novamente

O fragmento de código final é mostrado em Figura 65 na página 320.

```
    } while (0);
    if (CompCode == MQCC_FAILED && Reason != MQRC_NO_MSG_AVAILABLE)
        printf("MQI Call failed with reason code %d\n", Reason);
    if (Hsub!= MQHO_NONE)
        MQCLOSE(Hcon, &Hsub, SC_Options, &CompCode, &Reason);
    if (Hcon!= MQHC_UNUSABLE_HCONN)
        MQDISC(&Hcon, &CompCode, &Reason);
}
```

Figura 65. Conclusão

Interceptando publicações e publicar/assinar distribuído

Siga um padrão simples ao implementar a intercepção de assinantes ou saídas Publish para uma topologia distribuída de publicar/assinar. Implemente a interceptação de assinantes nos mesmos gerenciadores de filas que publicadores e saídas Publish nos mesmos gerenciadores de filas que assinantes finais.

Figura 66 na página 321 mostra dois gerenciadores de filas conectados em um cluster de assinatura de publicação. Um publicador cria uma publicação para um tópico de cluster no nível de publicação 9. As setas numeradas mostram a sequência de etapas tomadas pela publicação à medida que ela flui para os assinantes do tópico do cluster. A publicação é interceptada pelo assinante com `Sublevel` 9 e republicada com `Publevel` 8. Ela é interceptada novamente por um assinante no `Sublevel` 8. O assinante republica no `Publevel` 7. O assinante do proxy fornecido pelo gerenciador de filas encaminha a publicação para o gerenciador de filas B, em que uma saída Publish foi implementada além de um assinante final. A publicação é processada pela saída Publish antes de ser finalmente recebida pelo assinante final no `Sublevel` 1. Os assinantes de interceptação e a saída Publish são mostrados com as linhas de saída quebradas.

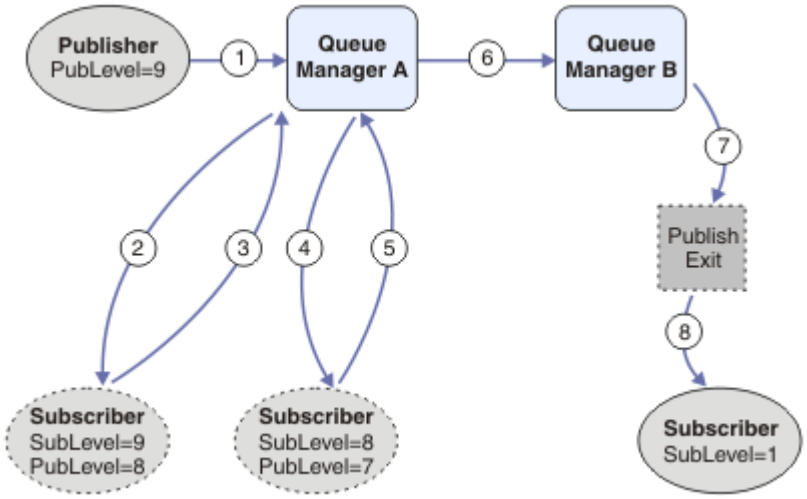


Figura 66. Interceptação e saída Publish em um cluster

O objetivo do padrão simples é para cada assinante receber uma publicação para receber a publicação idêntica. A publicação percorre a mesma sequência de transformações independentemente de onde o assinante está conectado. Provavelmente, você deseja evitar ter a sequência de transformações variando, dependendo de onde os publicadores ou assinantes finais estão conectados. Uma exceção razoável seria customizar a publicação definitivamente entregue a cada assinante individual. Use a saída Publish para customizar o aplicativo com base na fila na qual a publicação é finalmente entregue.

Deve-se considerar cuidadosamente onde implementar assinantes de interceptação e saídas Publish em uma topologia distribuída de publicar/assinar. O padrão direto implementa assinantes de interceptação no mesmo gerenciador de filas que os publicadores e saídas Publish nos mesmos gerenciadores de filas que os assinantes finais.

Antipadrão

Figura 67 na página 322 mostra como as coisas podem ficar distorcidas, se você não seguir um padrão simples. Para complicar a implementação, um assinante final é incluído no gerenciador de filas A e dois assinantes de interceptação adicionais são incluídos no gerenciador de filas B.

A publicação é encaminhada para o gerenciador de filas B no `PubLevel` 7, em que ela é interceptada por um assinante no `subnível` 5 antes de ser consumida pelo assinante final no `subnível` 1. A saída Publish intercepta a publicação antes de ela ser transmitida tanto para o consumidor interceptador quanto para o consumidor final no gerenciador de filas B. A publicação chega ao assinante final no gerenciador de filas A sem ser processada pela saída Publish.

Em uma topologia de publicar/assinar, os assinantes proxy assinam no `SubLevel` 1 e passam no `PubLevel` configurado pelo último assinante de interceptação. Em Figura 67 na página 322, o resultado é que a publicação não é interceptada pelo assinante usando `SubLevel` 9 no gerenciador de filas B.

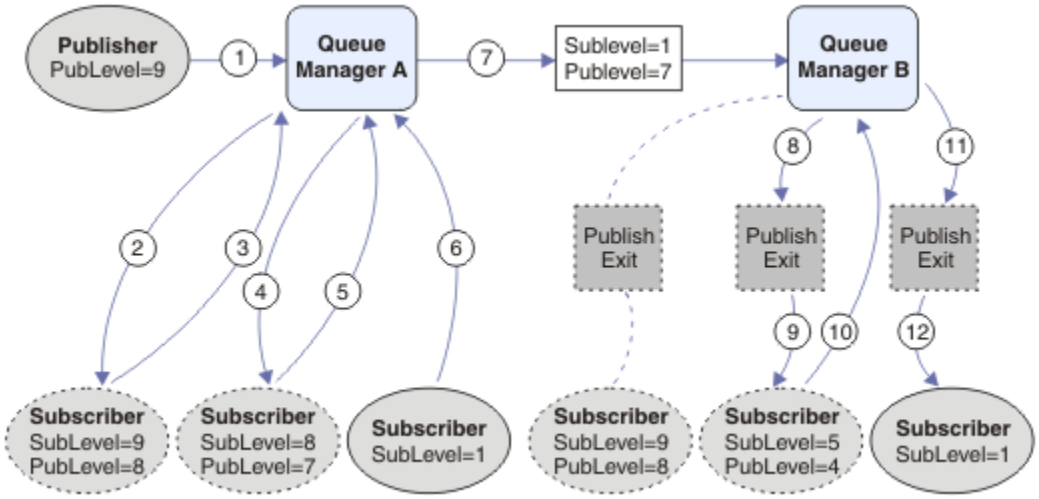


Figura 67. Implementação complexa de assinantes de interceptação

Opções de publicação

Estão disponíveis várias opções que controlam a maneira como as mensagens são publicadas.

Retendo informações de respostas de assinantes

Se não desejar que assinantes possam responder a publicações que recebem, é possível reter as informações nos campos ReplyToQ e ReplyToQmgr de MQMD usando a opção de put-message MQPMO_SUPPRESS_REPLYTO. Se essa opção for usada, o gerenciador de filas remove essas informações do MQMD quando ele recebe a publicação antes de encaminhá-la a todos os assinantes.

Essa opção não pode ser usada em combinação com uma opção de relatório que precisa de um ReplyToQ. Se isso for tentado, a chamada falhará com MQRC_MISSING_REPLY_TO_Q.

Nível de publicação

Usar níveis de publicação é uma maneira de controlar quais assinantes recebem a publicação. O nível de publicação indica o nível de assinatura almejado pela publicação. Somente assinaturas com o nível de assinatura mais alto menor ou igual ao nível da publicação receberão a publicação. Esse valor deve estar no intervalo de zero a nove; zero é o nível de publicação mais baixo. O valor inicial deste campo é de 9. Um dos usos dos níveis de publicação e de assinatura é para interceptar publicações.

Verificando se uma publicação não foi entregue a algum assinante

Para verificar se uma publicação não tiver sido entregue a todos os assinantes, use a opção de put-message MQPMO_WARN_IF_NO_SUBS_MATCHED com a chamada MQPUT. Se um código de conclusão MQCC_WARNING e um código de razão MQRC_NO_SUBS_MATCHED forem retornados pela operação put, a publicação não foi entregue para nenhuma assinaturas. Se a opção MQPMO_RETAIN for especificada na operação put, a mensagem será retida e entregue a qualquer assinatura correspondente definida subsequentemente. Em um sistema distribuído de publicar/assinar, o código de razão MQRC_NO_SUBS_MATCHED será retornado somente se não houver nenhuma assinatura de proxy registrado para o tópico no gerenciador de filas.

Opções de Assinatura

Estão disponíveis várias opções que controlam a maneira como as assinaturas de mensagens são manipuladas.

Persistência de mensagem

Gerenciadores de filas mantêm a persistência das publicações que encaminham aos assinantes conforme configurado pelo publicador. O publicador configura a persistência para uma das opções a seguir:

0
Não persistente

1
Persistente

2
Persistência como definição de fila/tópico

Para publicar/assinar, o publicador resolve o objeto do tópico e **topicString** para um objeto do tópico resolvido. Se o publicador especificar Persistência como definição de fila/tópico, então, a persistência padrão do objeto do tópico resolvido será configurada para a publicação.

Publicações Retidas

Para controlar quando as publicações retidas são recebidas, os assinantes podem usar duas opções de assinatura:

Publicar somente sob solicitação, MQSO_PUBLICATIONS_ON_REQUEST

Se desejar que um assinante tenha controle de quando recebe publicações, será possível usar a opção de assinatura MQSO_PUBLICATIONS_ON_REQUEST. Um assinante pode, então, controlar quando recebe publicações usando a chamada MQSUBRQ (especificando o identificador Hsub que foi retornado da chamada MQSUB original) para solicitar que seja enviada a ele uma publicação retida do tópico. Os assinantes que usam a opção de assinatura MQSO_PUBLICATIONS_ON_REQUEST não recebem nenhuma não retida.

Se você especificar MQSO_PUBLICATIONS_ON_REQUEST, deve-se usar MQSUBRQ para recuperar qualquer publicação. Se não usar MQSO_PUBLICATIONS_ON_REQUEST, obterá as mensagens conforme elas forem publicadas.

Se um assinante usar a chamada MQSUBRQ e usar curingas no tópico da assinatura, a assinatura poderá corresponder a vários tópicos ou nós em uma árvore de tópicos, todos com mensagens retidas (se houver alguma) serão enviados ao assinante.

Essa opção pode ser útil principalmente quando usada com assinaturas duráveis porque um gerenciador de filas continuará a enviar publicações a um assinante se tiver assinado de forma durável mesmo que esse aplicativo de assinante não esteja em execução. Isso pode levar a um acúmulo de mensagens na fila de assinantes. Esse acúmulo pode ser evitado se o assinante se registrar usando a opção MQSO_PUBLICATIONS_ON_REQUEST. Como alternativa, será possível usar assinaturas não duráveis, se apropriado para seu aplicativo, para evitar um acúmulo de mensagens indesejadas.

Se uma assinatura for durável e um publicador usar publicações retidas, o aplicativo de assinante poderá usar a chamada MQSUBRQ para atualizar suas informações de estado após uma reinicialização. O assinante deve, então, atualizar seu estado periodicamente usando a chamada MQSUBRQ.

Nenhuma publicação será enviada como resultado da chamada MQSUB usando essa opção. Uma assinatura durável continuada após desconexão usará a opção MQSO_PUBLICATIONS_ON_REQUEST se a assinatura original tiver sido configurada para usar esta opção.

Somente novas publicações, MQSO_NEW_PUBLICATIONS_ONLY

Se existir uma publicação retida em um tópico, todos os assinantes que fizerem uma assinatura após a publicação receberão uma cópia dessa publicação. Se um assinante não desejar receber as publicações que foram feitas antes da assinatura que está sendo feita, ele poderá usar a opção de assinatura MQSO_NEW_PUBLICATIONS_ONLY.

Agrupando assinaturas

Considere o agrupamento de assinaturas se você tiver configurado uma fila para receber publicações e tiver várias assinaturas sobrepostas alimentando publicações para a mesma fila. Essa situação é semelhante ao exemplo em Assinaturas sobrepostas.

É possível evitar receber publicações duplicadas configurando a opção MQSO_GROUP_SUB ao assinar um tópico. O resultado é que quando mais de uma assinatura no grupo corresponde ao tópico de uma publicação, somente uma assinatura é responsável por colocar a publicação na fila. As outras assinaturas correspondentes ao tópico de publicação são ignoradas.

A assinatura responsável por colocar a publicação na fila é escolhida com base no fato de ter a sequência de tópicos correspondente mais longa, antes de encontrar quaisquer curingas. Ela pode ser considerada como a assinatura correspondente mais próxima. Suas propriedades são propagadas para a publicação, inclusive se tiver a propriedade MQSO_NOT_OWN_PUBS. Se ocorrer, nenhuma publicação será entregue à fila, mesmo que outras assinaturas correspondentes possam não ter a propriedade MQSO_NOT_OWN_PUBS.

Não é possível colocar todas as suas assinaturas em um único grupo para eliminar publicações duplicadas. As assinaturas agrupadas devem atender as condições a seguir:

1. Nenhuma das assinaturas é gerenciada.
2. Um grupo de assinaturas entrega publicações para a mesma fila.
3. Cada assinatura deve estar no mesmo nível de assinatura.
4. A mensagem de publicação para cada assinatura no grupo tem o mesmo identificador de correlação.

   Para assegurar que cada assinatura resulte em uma mensagem de publicação com o mesmo identificador de correlação, configure MQSO_SET_CORREL_ID para criar seu próprio identificador de correlação na publicação e configure o mesmo valor no campo **SubCorrelId** em cada assinatura. Não configure **SubCorrelId** para o valor MQCI_NONE.

Consulte ../com.ibm.mq.ref.dev.doc/q100080_.dita#q100080_/mqso_group_sub para obter mais informações.

Consultando e configurando atributos de objeto

Atributos são as propriedades que definem as características de um objeto WebSphere MQ .

Elas afetam a maneira que um gerenciador de filas processa um objeto. Os atributos de cada tipo de objeto do WebSphere MQ são descritos em detalhes em Atributos de objetos

Alguns atributos são configurados quando o objeto é definido e podem ser alterados apenas usando os comandos WebSphere MQ ; um exemplo de tal atributo é a prioridade padrão para mensagens colocadas em uma fila. Outros atributos são afetados pela operação do gerenciador de filas e podem mudar com o tempo; um exemplo é a profundidade atual de uma fila.

É possível questionar sobre os valores atuais da maioria dos atributos usando a chamada MQINQ. O MQI também fornece uma chamada MQSET com a qual é possível mudar alguns atributos da fila. Não é possível usar as chamadas MQI para mudar os atributos de qualquer outro tipo de objeto; em vez disso, deve-se usar:

Para WebSphere MQ para plataformas Windows, UNIX e Linux

O recurso MQSC, descrito em Referência de MQSC.

Nota: Os nomes dos atributos de objetos são mostrados nesta documentação no formulário que você usa com as chamadas MQINQ e MQSET. Ao usar comandos WebSphere MQ para definir, alterar ou exibir os atributos, você deve identificar os atributos usando as palavras-chave mostradas nas descrições dos comandos nos links do tópico.

Ambas chamadas MQINQ e MQSET usam matrizes de seletores para identificaros atributos que você deseja pesquisar sobre ou configurar. Há um seletor para cada atributo com que é possível trabalhar. O nome do seletor possui um prefixo determinado pela natureza do atributo:

MQCA_	Esses seletores referem-se a atributos que contêm dados de caractere (por exemplo, o nome de uma fila).
MQIA_	Esses seletores se referem a atributos que contêm tanto valores numéricos (como *CurrentQueueDepth*, o número de mensagens em uma fila) ou um valor constante (como *SyncPoint*, se o gerenciador de filas suportar sincronização).

Antes de usar as chamadas MQINQ ou MQSET, o seu aplicativo deve estar conectado ao gerenciador de filas e deve-se usar a chamada MQOPEN para abrir o objeto para configurar ou consultar sobre atributos. Essas operações estão descritas em "Conectando-se e desconectando-se de um gerenciador de filas" na página 209 e "Abrindo e fechando objetos" na página 218.

Use os seguintes links para descobrir mais sobre consultar e configurar atributos do objeto:

- "Consultando sobre os atributos de um objeto" na página 325
- "Alguns casos em que a chamada MQINQ falha" na página 326
- "Configurando os atributos da fila" na página 327

Conceitos relacionados

"Visão geral da Message Queue Interface" na página 198
Aprenda sobre os componentes de Message Queue Interface (MQI).

"Conectando-se e desconectando-se de um gerenciador de filas" na página 209
Para usar WebSphere MQ serviços de programação, um programa deve ter uma conexão com um gerenciador de filas. Use essas informações para aprender como conectar-se e desconectar-se de um gerenciador de filas.

"Abrindo e fechando objetos" na página 218
Estas informações fornecem um insight sobre como abrir e fechar objetos do WebSphere MQ .

"Colocando mensagens em uma fila" na página 228
Use estas informações para aprender como colocar mensagens em uma fila.

"Obtendo mensagens de uma fila" na página 243
Use estas informações para aprender como obter mensagens de uma fila.

"Confirmando e fazendo backup de unidades de trabalho" na página 327
Essas informações descrevem como confirmar e restaurar quaisquer operações de GET e PUT que ocorreram em uma unidade de trabalho.

"Iniciando aplicativos IBM WebSphere MQ usando acionadores" na página 333
Aprenda sobre acionadores e como iniciar aplicativos IBM WebSphere MQ usando acionadores.

"Trabalhando com MQI e clusters" na página 351
Há opções especiais em chamadas e códigos de retorno relacionadas ao armazenamento em cluster.

Consultando sobre os atributos de um objeto

Use a chamada MQINQ para consultar sobre os atributos de qualquer tipo de IBM WebSphere MQ.

Como entrada para essa chamada, deve-se fornecer:

- Uma manipulação de conexões.
- Uma manipulação de objetos.
- O número de seletores.
- Uma matriz de seletores de atributos, cada seletor tendo o formulário MQCA_* ou MQIA_*. Cada seletor representa um atributo com um valor que você deseja consultar e cada seletor deve ser válido para o tipo de objeto que o identificador de objeto representa. É possível especificar os seletores em qualquer ordem.
- O número de atributos de número inteiro sobre os quais você está consultando. Especifique zero se você não estiver consultando sobre atributos de número inteiro.

- O comprimento do buffer de atributos de caracteres em *CharAttrLength*. Deve ter pelo menos a soma dos comprimentos necessários para reter cada sequência de atributos de caracteres. Especifique zero se não estiver consultando sobre atributos de caracteres.

A saída de MQINQ é:

- Um conjunto de valores de atributos de número inteiro copiado para a matriz. O número de valores é determinado por *IntAttrCount*. Se *IntAttrCount* ou *SelectorCount* for zero, esse parâmetro não será usado.
- O buffer no qual os atributos de caracteres são retornados. O comprimento do buffer é fornecido pelo parâmetro *CharAttrLength*. Se *CharAttrLength* ou *SelectorCount* for zero, esse parâmetro não será usado.
- Um código de conclusão. Se o código de conclusão fornecer um aviso, isso significa que a chamada foi concluída apenas parcialmente. Neste caso, examine o código de razão.
- Um código de razão. Há três situações de conclusão parcial:
  - O seletor não se aplica ao tipo de fila
  - Não há espaço suficiente permitido para atributos de número inteiro
  - Não há espaço suficiente permitido para atributos de caracteres

  Se mais de uma dessas situações surgir, o primeiro que se aplicar será retornado.

Se você abrir uma fila para saída ou consulta e ela resolver em uma fila de cluster não local, será possível consultar somente o nome da fila, o tipo de fila e os atributos comuns. Os valores dos atributos comuns são aqueles da fila escolhida se MQOO_BIND_ON_OPEN tiver sido usado. Os valores são aqueles de uma arbitrária entre as filas de cluster possíveis se MQOO_BIND_NOT_FIXED ou MQOO_BIND_ON_GROUP tiver sido usado ou MQOO_BIND_AS_Q_DEF tiver sido usado e o atributo de fila *DefBind* era MQBND_BIND_NOT_FIXED. Consulte “MQOPEN e clusters” na página 352 e MQOPEN para obter mais informações.

Nota: Os valores retornados pela chamada são uma captura instantânea dos atributos selecionados. Os atributos podem mudar antes que seu programa atue nos valores retornados.

Há uma descrição da chamada MQINQ em MQINQ.

Alguns casos em que a chamada MQINQ falha

Se você abrir um alias para consultar sobre seus atributos, será retornado os atributos da fila de alias (o objeto WebSphere MQ utilizado para acessar outra fila), não aqueles da fila base.

No entanto, a definição da fila de base para a qual o alias resolve também é aberta pelo gerenciador de filas e se outro programa mudar o uso da fila de base no intervalo entre suas chamadas MQOPEN e chamadas MQINQ, sua chamada MQINQ falha e retorna o código de razão MQRC_OBJECT_CHANGED. A chamada também falhará se os atributos do objeto da fila de alias forem mudados.

Da mesma forma, quando você abre uma fila remota para consultar sobre seus atributos, os atributos da definição local da fila remota apenas são retornados.

Se você especificar um ou mais seletores que não sejam válidos para o tipo de atributos de filas, o qual você está consultando, a chamada MQINQ será concluída com um aviso e configurará a saída conforme a seguir:

- Para atributos de número inteiro, os elementos correspondentes de *IntAttrs* são configurados como MQIAV_NOT_APPLICABLE.
- Para os atributos de caracteres, as partes correspondentes da sequência *CharAttrs* são configuradas como asteriscos.

Se você especificar um ou mais seletores que não sejam válidos para o tipo de atributos de objeto, o qual você está consultando, a chamada MQINQ falhará e retornará o código de razão MQRC_SELECTOR_ERROR.

Não é possível chamar MQINQ para consultar uma fila modelo; use o recurso MQSC ou os comandos disponíveis em sua plataforma.

Configurando os atributos da fila

Use estas informações para aprender como configurar atributos de filas usando a chamada MQSET.

É possível configurar somente os seguintes atributos de filas usando a chamada MQSET:

- *InhibitGet* (mas não para filas remotas)
- *DistList* (não no z/OS)
- *InhibitPut*
- *TriggerControl*
- *TriggerType*
- *TriggerDepth*
- *TriggerMsgPriority*
- *TriggerData*

A chamada MQSET possui os mesmos parâmetros que a chamada MQINQ. No entanto, para MQSET, todos os parâmetros, exceto o código de conclusão e o código de razão, são parâmetros de entrada. Não há situações conclusão parcial.

Nota: Não é possível usar o MQI para configurar os atributos de objetos WebSphere MQ diferentes de filas definidas localmente.

Para obter mais detalhes sobre a chamada MQSET, consulte MQSET.

Confirmando e fazendo backup de unidades de trabalho

Essas informações descrevem como confirmar e restaurar quaisquer operações de GET e PUT que ocorreram em uma unidade de trabalho.

Os termos a seguir são usados neste tópico:

- Confirmar
- Restaurar
- Coordenação do ponto de sincronização
- Ponto de Sincronização
- Unidade de trabalho
- Single-phase commit
- Two-phase commit

Se você estiver familiarizado com esses termos de processamento de transações, será possível pular para “Considerações sobre o ponto de sincronização em aplicativos IBM WebSphere MQ” na página 329.

Confirmação e restauração

Quando um programa coloca uma mensagem em uma fila em uma unidade de trabalho, essa mensagem é visível para outros programas apenas quando o programa confirma a unidade de trabalho. Para confirmar uma unidade de trabalho, todas as atualizações devem ser bem sucedidas em preservar a integridade dos dados. Se o programa detectar um erro e decidir que a operação put não é permanente, ele poderá restaurar a unidade de trabalho. Quando um programa executa uma restauração, o IBM WebSphere MQ restaura a fila removendo as mensagens que foram colocadas na fila por essa unidade de trabalho. A maneira na qual o programa executa a confirmação e a restauração das operações depende do ambiente no qual o programa está sendo executado.

Da mesma forma, quando um programa obtém uma mensagem de uma fila dentro de uma unidade de trabalho, essa mensagem permanece na fila até que o programa confirme a unidade de trabalho, mas a mensagem não está disponível para ser recuperada por outros programas. A mensagem é permanentemente excluída da fila quando o programa confirma a unidade de trabalho. Se o programa

restaurar a unidade de trabalho, o IBM WebSphere MQ restaurará a fila tornando as mensagens disponíveis para serem recuperadas por outros programas.

Coordenação do ponto de sincronização, ponto de sincronização, unidade de trabalho
Coordenação do ponto de sincronização é o processo pelo qual as unidades de trabalho são confirmadas ou restauradas com integridade de dados.

A decisão de confirmar ou restaurar as mudanças é tomada, no caso mais simples, no final de uma transação. No entanto, pode ser mais útil para um aplicativo sincronizar mudanças de dados em outros pontos lógicos dentro de uma transação. Esses pontos lógicos são chamados *pontos de sincronização* (ou *pontos de sincronização*) e o período de processamento de um conjunto de atualizações entre dois pontos de sincronização é chamado de *unidade de trabalho*. Várias chamadas MQGET e MQPUT podem fazer parte de uma única unidade de trabalho. O número máximo de mensagens dentro de uma unidade de trabalho pode ser controlado pelo atributo MAXUMSGS do comando ALTER QMGR em outras plataformas, exceto z/OS. Consulte a Referência MQSC *WebSphere MQ Script (MQSC)* para obter detalhes desses comandos.

Single-phase commit
Um processo *single-phase commit* é aquele no qual um programa pode confirmar atualizações em uma fila sem coordenar suas mudanças com outros gerenciadores de recursos.

Two-phase commit
Um processo *two-phase commit* é aquele no qual as atualizações que um programa fez para filas do IBM WebSphere MQ podem ser coordenadas com atualizações para outros recursos (por exemplo, bancos de dados sob o controle do DB2). Sob esse processo, as atualizações em todos os recursos são confirmadas ou restauradas juntas.

Para ajudar a manipular unidades de trabalho, o IBM WebSphere MQ fornece o atributo *BackoutCount*. Isso é incrementado cada vez que uma mensagem em uma unidade de trabalho é restaurada. Se a mensagem fizer repetidamente com que a unidade de trabalho seja encerrada de forma anormal, o valor de *BackoutCount* finalmente excederá aquele do *BackoutThreshold*. Este valor é configurado quando a fila é definida. Nesta situação, o aplicativo pode remover a mensagem da unidade de trabalho e colocá-la em outra fila, conforme definido em *BackoutRequeueQName*. Quando a mensagem é movida, a unidade de trabalho pode confirmar.

Use os seguintes links para saber mais sobre a confirmação e a restauração de unidades de trabalho:

- “Considerações sobre o ponto de sincronização em aplicativos IBM WebSphere MQ” na página 329
- “Pontos de sincronização em IBM WebSphere MQ nos sistemas UNIX, Linux, and Windows” na página 330

Conceitos relacionados

“Visão geral da Message Queue Interface” na página 198
Aprenda sobre os componentes de Message Queue Interface (MQI).

“Conectando-se e desconectando-se de um gerenciador de filas” na página 209
Para usar WebSphere MQ serviços de programação, um programa deve ter uma conexão com um gerenciador de filas. Use essas informações para aprender como conectar-se e desconectar-se de um gerenciador de filas.

“Abrindo e fechando objetos” na página 218
Estas informações fornecem um insight sobre como abrir e fechar objetos do WebSphere MQ .

“Colocando mensagens em uma fila” na página 228
Use estas informações para aprender como colocar mensagens em uma fila.

“Obtendo mensagens de uma fila” na página 243
Use estas informações para aprender como obter mensagens de uma fila.

“Consultando e configurando atributos de objeto” na página 324
Atributos são as propriedades que definem as características de um objeto WebSphere MQ .

“Iniciando aplicativos IBM WebSphere MQ usando acionadores” na página 333
Aprenda sobre acionadores e como iniciar aplicativos IBM WebSphere MQ usando acionadores.

“Trabalhando com MQI e clusters” na página 351
Há opções especiais em chamadas e códigos de retorno relacionadas ao armazenamento em cluster.

Considerações sobre o ponto de sincronização em aplicativos IBM WebSphere MQ

Use estas informações para aprender sobre como usar pontos de sincronização em aplicativos IBM WebSphere MQ.

Two-phase commit é suportado sob:

- WebSphere MQ para AIX
- WebSphere MQ para HP-UX
- WebSphere MQ para Linux
- WebSphere MQ para Solaris
- WebSphere MQ para Windows
- CICS para MVS/ESA 4.1
- CICS Transaction Server para z/OS
- TXSeries
- IMS/ESA
- Outros coordenadores externos usando a interface X/Open XA

Single-phase commit é suportado sob:

- WebSphere MQ em sistemas UNIX
- WebSphere MQ para Windows

Nota: Para obter detalhes adicionais sobre interfaces externas, consulte “Interfaces para gerenciadores de pontos de sincronização externos” na página 332 e a documentação do XA *CAE Specification Distributed Transaction Processing: The XA Specification*, publicada pelo The Open Group. Os gerenciadores de transações (como CICS, IMS, Encinae Tuxedo) podem participar do two-phase commit, coordenado com outros recursos recuperáveis. Isso significa que as funções de enfileiramento fornecidas pelo WebSphere MQ podem ser colocadas no escopo de uma unidade de trabalho, gerenciada pelo gerenciador de transações.

Amostras enviadas com o WebSphere MQ mostram WebSphere MQ coordenando bancos de dados compatíveis com XA. Para obter informações adicionais sobre essas amostras, consulte “Programas de amostra para plataformas distribuídas” na página 98.

No aplicativo WebSphere MQ , é possível especificar em cada chamada put e get se você deseja que a chamada esteja sob controle do ponto de sincronização. Para fazer uma operação put operar sob o controle do ponto de sincronização, use o valor MQPMO_SYNCPOINT no campo *Options* da estrutura MQPMO ao chamar MQPUT. Para uma operação get, use o valor MQGMO_SYNCPOINT no campo *Options* da estrutura MQGMO. Se você não escolher explicitamente uma opção, a ação padrão depende da plataforma. O padrão de controle de ponto de sincronização é não.

Quando uma chamada MQPUT1 for emitida com MQPMO_SYNCPOINT, o comportamento padrão muda, de forma que a operação put seja concluída de forma assíncrona. Isso pode causar uma mudança no comportamento de alguns aplicativos que dependem de determinados campos nas estruturas MQOD e MQMD que estão sendo retornadas, mas que agora contêm valores não definidos. Um aplicativo pode especificar MQPMO_SYNC_RESPONSE para assegurar que a operação put seja executada de forma síncrona e que todos os valores de campos apropriados sejam preenchidos.

Quando seu aplicativo recebe um código de razão MQRC_BACKED_OUT em resposta a um MQPUT ou MQGET sob o ponto de sincronização, o aplicativo deve normalmente recuperar a transação atual usando MQBACK e, em seguida, se apropriado, tentar a transação inteira novamente. Se o aplicativo receber MQRC_BACKED_OUT em resposta a uma chamada MQCMIT ou MQDISC, não precisará chamar MQBACK.

Toda vez que uma chamada MQGET for restaurada, o campo *BackoutCount* da estrutura MQMD da mensagem afetada será incrementado. Um *BackoutCount* alto indica uma mensagem que foi repetidamente restaurada. Isso pode indicar um problema com essa mensagem, que é necessário investigar. Veja BackoutCount para obter detalhes de *BackoutCount*.

Se um programa emitir a chamada MQDISC enquanto houver solicitações não confirmadas, um ponto de sincronização implícito ocorrerá Se o programa for encerrado de forma anormal, ocorre uma restauração implícita.

Mudanças nos atributos da fila (pela chamada MQSET ou por comandos) não são afetadas pela confirmação ou restauração das unidades de trabalho.

Pontos de sincronização em IBM WebSphere MQ nos sistemas UNIX, Linux, and Windows

O suporte do ponto de sincronização opera em dois tipos de unidades de trabalho: local e global.

Uma unidade de trabalho *local* é aquela na qual somente os recursos atualizados são os do gerenciador de fila WebSphere MQ. Aqui, a coordenação do ponto de sincronização é fornecida pelo próprio gerenciador de filas usando um procedimento single-phase commit.

Uma unidade de trabalho *global* é aquela na qual os recursos que pertencem a outros gerenciadores de recursos, como bancos de dados, também são atualizados. WebSphere MQ pode coordenar essas próprias unidades de trabalho. Elas também podem ser coordenadas por um controlador de compromisso externo, como outro gerenciador de transações ou o IBM i controlador de confirmação.

Para integridade completa, use um procedimento two-phase commit. O two-phase commit pode ser fornecido por gerenciadores de transações e bancos de dados compatíveis com XA, como TXSeries e UDB.. WebSphere MQ produtos (exceto WebSphere MQ para IBM i e WebSphere MQ para z/OS) podem coordenar unidades globais de trabalho usando um processo two-phase commit.

Unidades de trabalho locais

Unidades de trabalho que envolvem somente o gerenciador de filas são chamadas de unidades de trabalho *locais*. A coordenação do ponto de sincronização é fornecida pelo próprio gerenciador de filas (coordenação interna) utilizando um processo single-phase commit.

Para iniciar uma unidade de trabalho local, o aplicativo emite solicitações MQGET, MQPUT ou MQPUT1 especificando a opção de ponto de sincronização apropriada. A unidade de trabalho é confirmada usando MQCMIT ou retrocedida usando MQBACK. No entanto, a unidade de trabalho também é encerrada quando a conexão entre o aplicativo e o gerenciador de filas é interrompida, intencionalmente ou não.

Se um aplicativo se desconectar (MQDISC) de um gerenciador de filas enquanto uma unidade global de trabalho coordenada pelo WebSphere MQ ainda estiver ativa, será feita uma tentativa de confirmar a unidade de trabalho.. Se, no entanto, o aplicativo for finalizado sem desconectar, a unidade de trabalho será retrocedida, pois é considerado que o aplicativo foi finalizado de forma anormal.

Unidades de trabalho globais

Use unidades de trabalho globais quando também precisar incluir atualizações nos recursos pertencentes a outros gerenciadores de recursos.

Aqui, a coordenação pode ser interna ou externas para o gerenciador de filas:

Coordenação de ponto de sincronização interno

A coordenação do gerenciador de filas de unidades globais de trabalho não é suportada por WebSphere MQ para IBM i ou WebSphere MQ para z/OS Não é suportado em um WebSphere MQ ambiente do cliente MQI.

Aqui, o WebSphere MQ faz a coordenação. Para iniciar uma unidade de trabalho global, o aplicativo emite a chamada MQBEGIN.

Como entrada para a chamada MQBEGIN, deve-se fornecer a manipulação de conexões (*Hconn*) que é retornada pela chamada MQCONN ou MQCONNX. Essa manipulação representa a conexão com o gerenciador de fila do WebSphere MQ

O aplicativo emite solicitações MQGET, MQPUT ou MQPUT1 especificando a opção do ponto de sincronização apropriada. Isso significa que é possível usar MQBEGIN para iniciar uma unidade de trabalho global que atualiza recursos locais, recursos que pertencem a outros gerenciadores de recursos, ou ambos. As atualizações feitas nos recursos que pertencem a outros gerenciadores de recursos são feitas usando a API do gerenciador de recursos. No entanto, não é possível usar a MQI para atualizar filas que pertencem a outros gerenciadores de filas. Emita MQCMIT ou MQBACK antes de iniciar mais unidades de trabalho (local ou global).

A unidade de trabalho global é confirmada usando MQCMIT; isso inicia um two-phase commit de todos os gerenciadores de recursos envolvidos na unidade de trabalho. Um processo two-phase commit é usado por meio do qual os gerenciadores de recursos (por exemplo, gerenciadores de banco de dados compatíveis com XA, como DB2, Oraclee Sybase) são todos solicitados a se preparar para confirmar. Somente se todos estiverem preparados serão solicitados para confirmar. Se qualquer gerenciador de recursos indicar que ele não pode confirmar, cada um é solicitado para restaurar. Como alternativa, é possível usar MQBACK para retroceder as atualizações de todos os gerenciadores de recursos.

Se um aplicativo se desconectar (MQDISC) enquanto uma unidade de trabalho global ainda estiver ativa, a unidade de trabalho será confirmada. Se, no entanto, o aplicativo for finalizado sem desconectar, a unidade de trabalho será retrocedida, pois é considerado que o aplicativo foi finalizado de forma anormal.

A saída de MQBEGIN é um código de conclusão e um código de razão.

Ao usar MQBEGIN para iniciar uma unidade de trabalho global, todos os gerenciadores de recursos externos que foram configurados com o gerenciador de filas são incluídos. No entanto, a chamada iniciar uma unidade de trabalho, mas é concluída com um aviso se:

- Não há gerenciadores de recursos participantes (ou seja, nenhum gerenciador de recurso foi configurado com o gerenciador de filas)

ou

- Um ou mais gerenciadores de recursos não estão disponíveis.

Nesses casos, a unidade de trabalho deve incluir atualizações somente nos gerenciadores de recursos que estavam disponíveis quando a unidade de trabalho foi iniciada.

Se um dos gerenciadores de recursos não puder confirmar suas atualizações, todos os gerenciadores de recursos serão instruídos a retroceder suas atualizações e MQCMIT será concluído com um aviso. Em circunstâncias incomuns (geralmente, a intervenção do operador), uma chamada MQCMIT pode falhar se alguns gerenciadores de recursos confirmarem suas atualizações, mas outros as recuperarem; o trabalho será considerado como tendo sido concluído com um resultado *misto*. Tais ocorrências são diagnosticadas no log de erros do gerenciador de filas para que ação corretiva possa ser executada.

Um MQCMIT de uma unidade de trabalho global será bem-sucedido se todos os gerenciadores de recursos envolvidos confirmarem suas atualizações.

Para obter uma descrição da chamada MQBEGIN, consulte MQBEGIN.

Coordenação de ponto de sincronização externo

Isso ocorre quando um coordenador do ponto de sincronização diferente de WebSphere MQ foi selecionado; por exemplo, CICS, Encina, ou Tuxedo

Nessa situação, o WebSphere MQ em sistemas UNIX and Linux e o WebSphere MQ para Windows registram seu interesse no resultado da unidade de trabalho com o coordenador do ponto de sincronização para que eles possam confirmar ou retroceder qualquer operação get ou put não confirmada, conforme necessário. O coordenador do ponto de sincronização externo determina se os protocolos de one-phase commit e two-phase commit são fornecidos.

Ao usar um coordenador externo, MQCMIT, MQBACK e MQBEGIN não podem ser emitidos. Chamadas para essas funções falham com o código de razão MQRC_ENVIRONMENT_ERROR.

A maneira pela qual uma unidade de trabalho coordenada externamente é iniciada depende da interface de programação fornecida pelo coordenador do ponto de sincronização. Uma chamada explícita pode ser necessária. Se uma chamada explícita for necessária e você emitir uma chamada MQPUT especificando a opção MQPMO_SYNCPOINT quando uma unidade de trabalho não estiver iniciada, o código de conclusão MQRC_SYNCPOINT_NOT_AVAILABLE será retornado.

O escopo da unidade de trabalho é determinado pelo coordenador do ponto de sincronização. O estado da conexão entre o aplicativo e o gerenciador de filas afeta o sucesso ou falha de chamadas MQI que um aplicativo emite, não o estado da unidade de trabalho. Um aplicativo pode, por exemplo, desconectar e reconectar a um gerenciador de filas durante uma unidade de trabalho ativa e executar outras operações MQGET e MQPUT na mesma unidade de trabalho. Isso é conhecido como uma desconexão pendente.

É possível usar chamadas API do WebSphere MQ em programas CICS , se você optar por usar as capacidades XA do CICS Se você não usar XA, as colocações e obtenções de mensagens para e a partir de filas não serão gerenciadas nas unidades atômicas de trabalho do CICS . Uma razão para escolher esse método é que a consistência geral da unidade de trabalho não é importante para você.

Se a integridade de suas unidades de trabalho for importante para você, então, deverá usar XA. Ao usar XA, o CICS usa um protocolo two-phase commit para assegurar que todos os recursos na unidade de trabalho sejam atualizados juntos.

Para obter mais informações sobre como configurar o suporte transacional, consulte “Cenários de Suporte Transacional” na página 42e também a documentação do TXSeries CICS , por exemplo, *TXSeries for Multiplatforms CICS Administration Guide for Open Systems*.

Interfaces para gerenciadores de pontos de sincronização externos

WebSphere MQ em UNIX and Linux sistemas, e WebSphere MQ para Windows coordenação de suporte de transações por gerenciadores de ponto de sincronização externos que usam a interface X/Open XA

Alguns gerenciadores de transações XA (TXSeries) requerem que cada gerenciador de recursos XA forneça o seu nome. Essa é a sequência chamada `name` na estrutura do comutador XA. O gerenciador de recursos para os sistemas WebSphere MQ em UNIX, Linux e Windows é denominado MQSeries_XA_RMI Para obter detalhes adicionais sobre interfaces XA, consulte a documentação do *CAE Specification Distributed Transaction Processing: The XA Specification*, publicada pelo The Open Group.

Em uma configuração XA, os sistemas WebSphere MQ em UNIX, Linuxe Windows cumprem a função de um Resource ManagerXA. Um coordenador do ponto de sincronização XA pode gerenciar um conjunto de Gerenciadores de recursos XA e sincronizar a confirmação ou a restauração de transações em ambos os Gerenciadores de recursos. É assim que funciona para um gerenciador de recursos registrado estaticamente:

1. Um aplicativo notifica o coordenador do ponto de sincronização que deseja iniciar uma transação.
2. O coordenador do ponto de sincronização emite uma chamada para todos os gerenciadores de recursos de seu conhecimento para notificá-los da transação atual.
3. O aplicativo emite chamadas para atualizar os recursos gerenciados pelos gerenciadores de recursos associados à transação atual.
4. O aplicativo solicita que o coordenador do ponto de sincronização confirme ou retroceda a transação.
5. O coordenador do ponto de sincronização emite chamadas para cada gerenciador de recursos usando protocolos two-phase commit para concluir a transação conforme solicitado.

A especificação XA requer que cada Gerenciador de recursos forneça uma estrutura denominada um *Comutador XA*. Essa estrutura declara as capacidades do Gerenciador de recursos e as funções que devem ser chamadas pelo coordenador do ponto de sincronização.

Há duas versões dessa estrutura:

MQRMIXASwitch	Gerenciamento de recursos XA estáticos
MQRMIXASwitchDynamic	Gerenciamento de recursos XA dinâmicos

Para obter uma lista das bibliotecas que contêm essa estrutura, consulte “A estrutura do comutador XA IBM WebSphere MQ” na página 71

O método que deve ser usado para vinculá-los a um coordenador de ponto de sincronização XA é definido pelo coordenador; consulte a documentação fornecida por esse coordenador para determinar como ativar o WebSphere MQ para cooperar com seu coordenador de ponto de sincronização XA.

A estrutura *xa_info* que é passada em qualquer chamada *xa_open* pelo coordenador do ponto de sincronização pode ser o nome do gerenciador de filas que deve ser administrado. Tem o mesmo formato que o nome do gerenciador de filas passado para MQCONN ou MQCONNX e pode ficar em branco se o gerenciador de filas padrão for ser usado. No entanto, é possível usar os dois parâmetros extras TPM e AXLIB

O TPM permite especificar para WebSphere MQ o nome do gerenciador de transações, por exemplo, CICS. AXLIB permite especificar o nome da biblioteca real no gerenciador de transações em que os pontos de entrada XA AX estão localizados.

Se você usar um desses parâmetros ou um gerenciador de filas não padrão, deve-se especificar o nome do gerenciador de filas usando o parâmetro QMNAME. Para obter informações adicionais, consulte Os parâmetros CHANNEL, TRPTYPE, CONNAME e QMNAME da sequência xa_open.

Restrições

1. Unidades de trabalho globais não são permitidas com um Hconn compartilhado (conforme descrito em “Conexões compartilhadas (independentes de encadeamento) com MQCONNX” na página 215).
2. Em sistemas Windows , todas as funções declaradas no comutador XA são declaradas como _cdecl.
3. Um coordenador de ponto de sincronização externo pode administrar somente um gerenciador de filas por vez. Isso porque o coordenador tem uma conexão efetiva para cada gerenciador de filas e, portanto, está sujeito à regra de que somente uma conexão seja permitida por vez.

   Nota: Nota: Um aplicativo cliente JMS (aplicativo CLIENT JEE) em execução em um servidor JEE não tem essa restrição, portanto, uma única transação gerenciada pelo servidor JEE pode coordenar vários gerenciadores de filas na mesma transação... No entanto, um aplicativo do servidor JMS, em execução no modo de ligações, ainda está sujeito à regra de que apenas uma conexão é permitida por vez
4. Todos os aplicativos que são executados usando o coordenador do ponto de sincronização podem se conectar somente ao gerenciador de filas administrado pelo coordenador porque já estão efetivamente conectados a esse gerenciador de filas. Eles devem emitir MQCONN ou MQCONNX para obter uma manipulação de conexões e devem emitir MQDISC antes de sair. Como alternativa, eles podem utilizar a saída UE014015 para TXSeries CICS..

Iniciando aplicativos IBM WebSphere MQ usando acionadores

Aprenda sobre acionadores e como iniciar aplicativos IBM WebSphere MQ usando acionadores.

Alguns aplicativos WebSphere MQ que atendem filas são executados continuamente, portanto eles estão sempre disponíveis para recuperar mensagens que chegam nas filas. No entanto, talvez você não queira isso quando o número de mensagens que chegam nas filas for imprevisível. Neste caso, os aplicativos poderiam estar consumindo os recursos do sistema mesmo quando não houver mensagens a recuperar.

O WebSphere MQ fornece um recurso que permite que um aplicativo seja iniciado automaticamente quando houver mensagens disponíveis para recuperação Este recurso é conhecido como *acionamento*.

Para obter informações sobre canais acionadores, consulte Canais acionadores .

O que É o Acionamento?

O gerenciador de filas define certas condições que constituem os *eventos acionadores*.

Se o acionamento for ativado para uma fila e ocorrer um evento acionador, o gerenciador de filas enviará uma *mensagem do acionador* para uma fila denominada *fila de inicialização*. A presença da mensagem do acionador na fila de inicialização indica que ocorreu um evento acionador.

As mensagens do acionador geradas pelo gerenciador de filas não são persistentes. Isso reduz a criação de log (resultando em um desempenho melhor) e minimiza as duplicatas durante a reinicialização, melhorando, assim, o tempo de reinicialização.

O programa que processa a fila de inicialização é chamado de aplicativo *acionador-monitor* e sua função é ler a mensagem do acionador e executar a ação apropriada, com base nas informações contidas na mensagem do acionador. Geralmente, esta ação é iniciar algum outro aplicativo para processar a fila que gerou a mensagem do acionador. Do ponto de vista do gerenciador de filas, não há nada de especial sobre o aplicativo acionador-monitor; ele é simplesmente outro aplicativo que lê mensagens de uma fila (a fila de inicialização).

Se o acionamento for ativado para uma fila, será possível criar um *objeto de definição de processo* associado a ele. Este objeto contém informações sobre o aplicativo que processa a mensagem que causou o evento do acionador. Se o objeto de definição de processo for criado, o gerenciador de filas extrairá essas informações e as colocará na mensagem do acionador para serem usadas pelo aplicativo acionador-monitor. O nome da definição de processo associada a uma fila é determinado pelo atributo de fila local *ProcessName*. Cada fila pode especificar uma definição de processo diferente ou várias filas podem compartilhar a mesma definição de processo.

Se desejar acionar o início de um canal, não será necessário definir um objeto de definição de processo. A definição da fila de transmissão é usada em substituição.

O acionamento é suportado pelos clientes do WebSphere MQ em execução nos seguintes ambientes:

- Sistemas UNIX and Linux
- Sistemas Windows

Um aplicativo em execução em um ambiente do cliente é o mesmo que um em execução em um ambiente completo do WebSphere MQ , exceto que você o vincula com as bibliotecas do cliente No entanto, o monitor do acionador e o aplicativo a ser iniciado devem estar no mesmo ambiente.

O acionamento envolve:

Fila de aplicativos
Uma *fila de aplicativos* é uma fila local que, quando tem o acionamento configurado e quando as condições são atendidas, requer que as mensagens do acionador sejam gravadas.

Definição de processo
Uma fila de aplicativos pode ter um *objeto de definição de processo* associado a ela que mantém detalhes do aplicativo que obterá mensagens da fila de aplicativos. (Consulte Atributos para definições de processo para obter uma lista de atributos.)

Lembre-se de que se desejar que um acionador inicie um canal, não será necessário definir um objeto de definição de processo.

Fila de transmissão
Você precisará de uma fila de transmissão, se desejar que um acionador inicie um canal.

Para uma fila de transmissão em sistemas AIX, HP-UX, IBM i, Solaris, z/OSou Janelas , o atributo *TriggerData* da fila de transmissão pode especificar o nome do canal a ser iniciado. Isso pode substituir a definição do processo para acionar os canais, mas é usado apenas quando uma definição de processo não é criada.

Evento acionador
Um *evento do acionador* é um evento que faz com que uma mensagem do acionador seja gerada pelo gerenciador de filas. Geralmente, essa é uma mensagem chegando em uma fila de aplicativos, mas também pode ocorrer em outros momentos (consulte “Condições para um evento acionador” na página 339). WebSphere MQ possui um intervalo de opções para permitir que você controle as condições que causam um evento acionador (consulte “Controlando eventos acionadores” na página 343).

Mensagem do acionador
O gerenciador de filas cria uma *mensagem do acionador* quando ele reconhece um evento acionador (consulte “Condições para um evento acionador” na página 339). Ele copia para as informações de mensagem do acionador sobre o aplicativo a ser iniciado. Estas informações vêm da fila de aplicativos

e do objeto de definição de processo associado à fila de aplicativos. As mensagens do acionador têm um formato fixo (consulte “Formato de mensagens do acionador” na página 350).

Fila de Inicialização

Uma *fila de inicialização* é uma fila local na qual o gerenciador de filas coloca mensagens do acionador. Observe que uma fila de inicialização não pode ser uma fila de alias ou uma fila modelo. Um gerenciador de filas pode possuir mais de uma fila de inicialização e cada uma é associada a uma ou mais filas de aplicativos. Uma fila compartilhada, uma fila local acessível por gerenciadores de filas em um grupo de filas compartilhadas, pode ser uma fila de inicialização no WebSphere MQ para z/OS..

Monitor acionador

Um *monitor acionador* é um programa continuamente em execução que atende uma ou mais filas de inicialização. Quando uma mensagem do acionador chega em uma fila de iniciação, o monitor de disparo recupera a mensagem. O monitor acionador usa as informações na mensagem do acionador. Ele emite um comando para iniciar o aplicativo que deve recuperar as mensagens que chegam na fila de aplicativos, transmitindo a ele as informações contidas no cabeçalho da mensagem do acionador, que inclui o nome da fila de aplicativos.

Em todas as plataformas, um monitor acionador especial conhecido como inicializador de canais é responsável por iniciar canais. No z/OS, o inicializador de canais geralmente é iniciado manualmente ou pode ser feito automaticamente quando um gerenciador de filas é iniciado mudando CSQINP2 na JCL de inicialização do gerenciador de filas. Em outras plataformas, ele é iniciado automaticamente quando o gerenciador de filas inicia ou pode ser iniciado manualmente com o comando runmqchi.

(Para obter mais informações, consulte “Processamento da fila de inicialização por monitores acionadores” na página 347.)

Para entender como o acionamento funciona, considere Figura 68 na página 335, que é um exemplo do tipo de acionador FIRST (MQTT_FIRST).

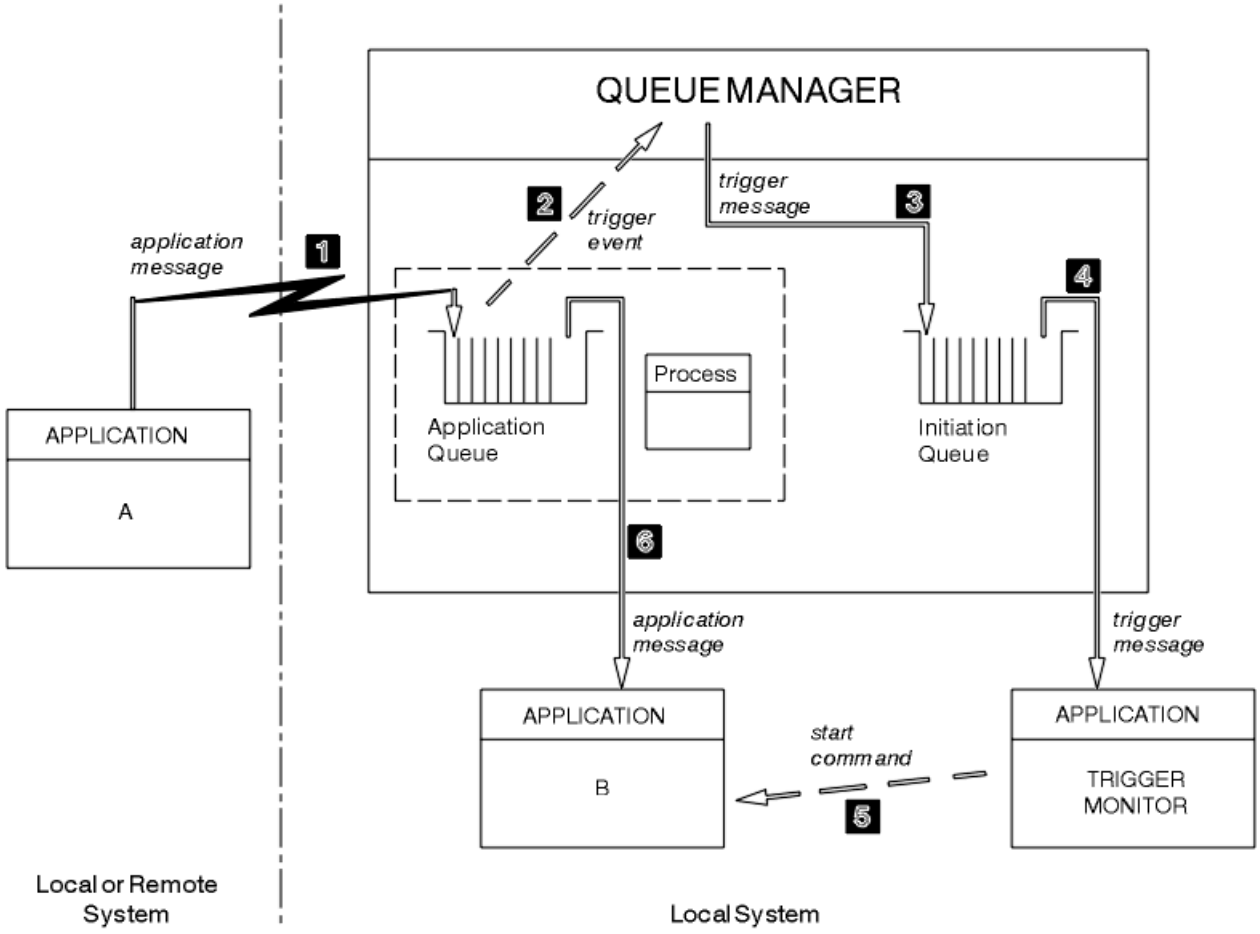


Figura 68. Fluxo de mensagens do aplicativo e do acionador

No Figura 68 na página 335, a sequência de eventos é:

1. O Aplicativo A, que pode ser local ou remoto para o gerenciador de filas, coloca uma mensagem na fila de aplicativos. Nenhum aplicativo tem essa fila aberta para entrada. No entanto, este fato é relevante apenas para acionar o tipo FIRST e DEPTH.
2. O gerenciador de filas verifica se as condições são atendidas sob as quais ele tem que gerar um evento acionador. Elas são, e um evento acionador é gerado. As informações retidas no objeto de definição de processo associado são usadas ao criar a mensagem do acionador.
3. O gerenciador de filas cria uma mensagem do acionador e coloca-a na fila de inicialização associada a esta fila de aplicativos, mas apenas se um aplicativo (monitor acionador) tiver a fila de inicialização aberta para entrada.
4. O monitor acionador recupera a mensagem do acionador da fila de inicialização.
5. O monitor acionador emite um comando para iniciar o aplicativo B (o aplicativo do servidor).
6. O Aplicativo B abre a fila de aplicativos e recupera a mensagem.

Nota:

1. Se a fila de aplicativos for aberta para entrada, por qualquer programa, e tiver o acionamento configurado como FIRST ou DEPTH, nenhum evento acionador ocorrerá, porque a fila já está sendo atendida.
2. Se a fila de inicialização não for aberta para entrada, o gerenciador de filas não gerará nenhuma mensagem do acionador; ele aguardará até que um aplicativo abra a fila de inicialização para entrada.
3. Ao usar o acionamento para os canais, use o tipo de acionador FIRST ou DEPTH.
4. Aplicativos acionados executados sob o ID do usuário e o grupo do usuário que iniciou o monitor acionador, o usuário do CICS ou o usuário que iniciou o gerenciador de filas.

Até o momento, o relacionamento entre as filas no acionamento tem sido apenas na base de um para um. Considere Figura 69 na página 337.

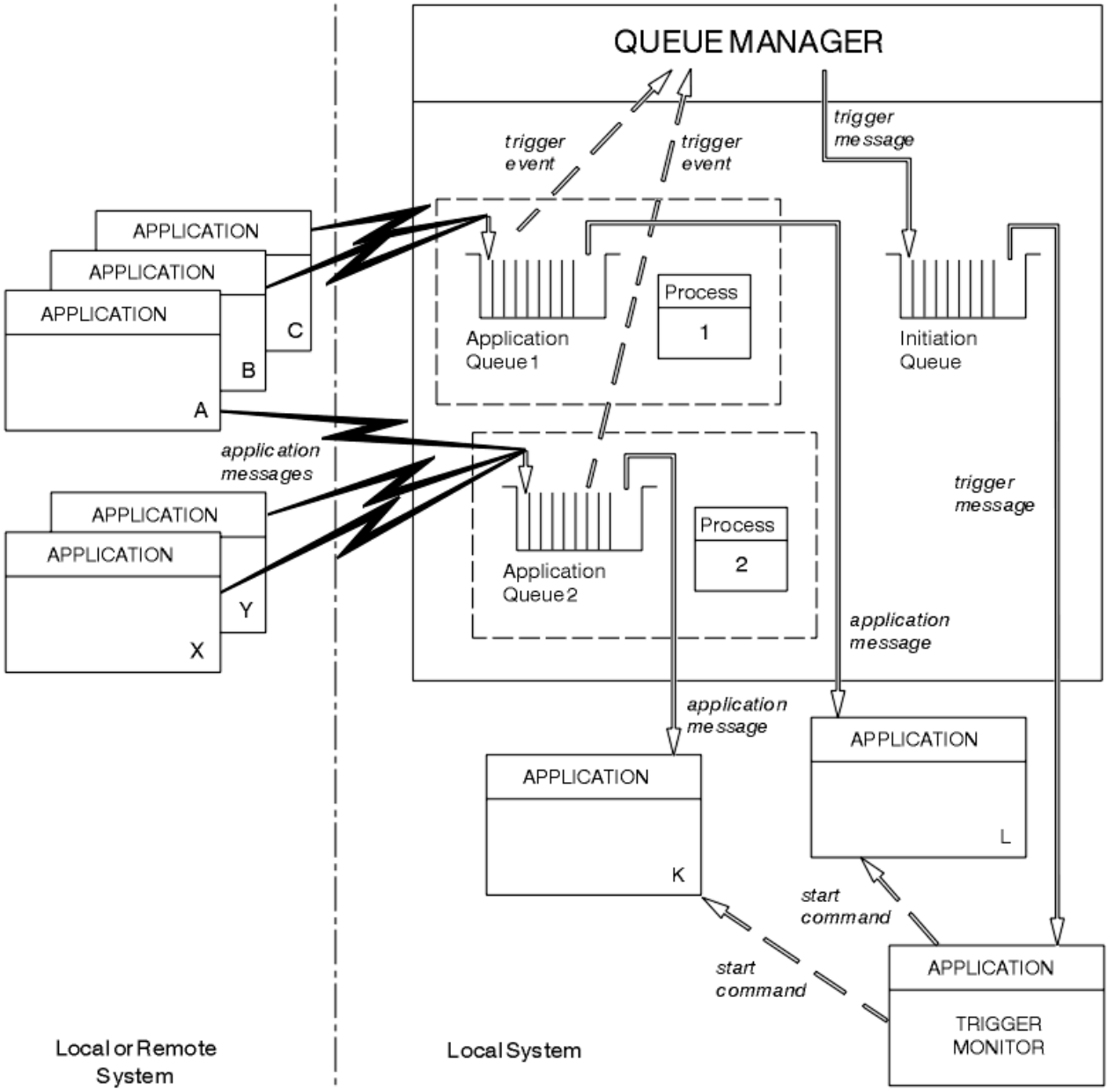


Figura 69. Relacionamento de filas no acionamento

Uma fila de aplicativos possui um objeto de definição de processo associado a ela que retém detalhes do aplicativo que processará a mensagem. O gerenciador de filas coloca as informações na mensagem do acionador, portanto, apenas uma fila de inicialização é necessária. O monitor acionador extrai essas informações da mensagem do acionador e inicia o aplicativo relevante para tratar a mensagem em cada fila de aplicativos.

Lembre-se de que, se desejar acionar o início de um canal, você não precisa definir um objeto de definição de processo. A definição de fila de transmissão pode determinar o canal a ser acionado.

Use os links a seguir para descobrir mais sobre como iniciar aplicativos WebSphere MQ usando acionadores:

- “Pré-requisitos para o acionamento” na página 338
- “Condições para um evento acionador” na página 339
- “Controlando eventos acionadores” na página 343
- “Projetando um aplicativo que usa filas acionadas” na página 346

- “Processamento da fila de inicialização por monitores acionadores” na página 347
- “Propriedades de mensagens do acionador” na página 349
- “Quando o acionamento não funciona” na página 351

Conceitos relacionados

“Visão geral da Message Queue Interface” na página 198
Aprenda sobre os componentes de Message Queue Interface (MQI).

“Conectando-se e desconectando-se de um gerenciador de filas” na página 209
Para usar WebSphere MQ serviços de programação, um programa deve ter uma conexão com um gerenciador de filas. Use essas informações para aprender como conectar-se e desconectar-se de um gerenciador de filas.

“Abrindo e fechando objetos” na página 218
Estas informações fornecem um insight sobre como abrir e fechar objetos do WebSphere MQ .

“Colocando mensagens em uma fila” na página 228
Use estas informações para aprender como colocar mensagens em uma fila.

“Obtendo mensagens de uma fila” na página 243
Use estas informações para aprender como obter mensagens de uma fila.

“Consultando e configurando atributos de objeto” na página 324
Atributos são as propriedades que definem as características de um objeto WebSphere MQ .

“Confirmando e fazendo backup de unidades de trabalho” na página 327
Essas informações descrevem como confirmar e restaurar quaisquer operações de GET e PUT que ocorreram em uma unidade de trabalho.

“Trabalhando com MQI e clusters” na página 351
Há opções especiais em chamadas e códigos de retorno relacionadas ao armazenamento em cluster.

Pré-requisitos para o acionamento

Use essas informações para aprender sobre as etapas a serem executadas antes de usar o acionamento.

Antes de seu aplicativo poder aproveitar o acionamento, conclua as seguintes etapas:

1. Execute um dos dois procedimentos:
   a. Crie uma fila de inicialização para sua fila de aplicativo. Por exemplo:

```
DEFINE QLOCAL (initiation.queue) REPLACE      +
       LIKE (SYSTEM.DEFAULT.INITIATION.QUEUE)      +
       DESCR ('initiation queue description')
```

   ou

   b. Determine o nome de uma fila local que existe e pode ser usada pelo aplicativo (geralmente, esse nome é SYSTEM.DEFAULT.INITIATION.QUEUE ou, se você estiver iniciando canais com acionadores, SYSTEM.CHANNEL.INITQ) e especifique seu nome no campo *InitiationQName* da fila de aplicativos.

2. Associe a fila de inicialização à fila de aplicativos. Um gerenciador de filas pode possuir mais de uma fila de inicialização. Talvez você queira que algumas de suas filas de aplicativos sejam atendidas por programas diferentes; neste caso, é possível usar uma fila de inicialização para cada programa de atendimento, embora isso não seja obrigatório. A seguir está um exemplo de como criar uma fila de aplicativos:

```
DEFINE QLOCAL (application.queue) REPLACE   +
LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE)           +
DESCR ('appl queue description')    +
INITQ ('initiation.queue')          +
PROCESS ('process.name')            +
```

```
TRIGGER                                 +
TRIGTYPE (FIRST)
```

3. Se estiver acionando um aplicativo, crie um objeto de definição de processo para conter informações relacionadas ao aplicativo que deve atender sua fila de aplicativos. Por exemplo, para acionar-iniciar uma transação da folha de pagamento do CICS chamada PAYR:

```
DEFINE PROCESS (process.name) +
       REPLACE +
       DESCR ('process description') +
       APPLICID ('PAYR') +
       APPLTYPE (CICS) +
       USERDATA ('Payroll data')
```

   Quando o gerenciador de filas cria uma mensagem do acionador, ele copia informações dos atributos do objeto de definição de processo para a mensagem do acionador.

Plataforma	Para criar um objeto de definição de processo
Sistemas UNIX, Linuxe Windows	Use DEFINE PROCESS ou use SYSTEM.DEFAULT.PROCESS e modifique usando ALTER PROCESS

4. Opcional: Crie uma definição de fila de transmissão e use espaços em branco para o atributo *ProcessName*.

   O atributo *TrigData* pode conter o nome do canal a ser acionado ou pode ser deixado em branco. Exceto no IBM WebSphere MQ for z/OS, se ele for deixado em branco, o inicializador de canais procurará os arquivos de definição de canal até localizar um canal que esteja associado à fila de transmissão nomeada. Quando o gerenciador de filas cria uma mensagem do acionador, ele copia as informações do atributo *TrigData* da definição de fila de transmissão para a mensagem do acionador.

5. Se você tiver criado um objeto de definição de processo para especificar propriedades do aplicativo que deve atender sua fila de aplicativos, associe o objeto do processo à sua fila de aplicativos nomeando-o no atributo *ProcessName* da fila.

Plataforma	Use os comandos
Sistemas UNIX, Linuxe Windows	ALTER QLOCAL

6. Inicie as instâncias dos monitores do acionador que devem atender as filas de inicialização que você definiu. Consulte “Processamento da fila de inicialização por monitores acionadores” na página 347 para obter mais informações.

Se você deseja estar ciente de quaisquer mensagens do acionador não entregues, certifique-se de que seu gerenciador de filas possua uma fila de devoluções (mensagens não entregues) definida. Especifique o nome da fila no campo do gerenciador de filas *DeadLetterQName*.

É possível, então, configurar as condições do acionador que você precisa, usando os atributos do objeto de fila que define sua fila de aplicativos. Para obter mais informações, consulte “Controlando eventos acionadores” na página 343.

Condições para um evento acionador

As referências a filas compartilhadas neste tópico significam filas compartilhadas em um grupo de filas compartilhadas, disponíveis apenas no WebSphere MQ para z/OS

O gerenciador de filas cria uma mensagem do acionador quando as seguintes condições são satisfeitas:

1. Uma mensagem tem *put* efetuado em uma fila.

2. A mensagem tem uma prioridade maior ou igual à prioridade do acionador de limite da fila. Essa prioridade é configurada no atributo de fila local *TriggerMsgPriority*; se for configurada para zero, nenhuma mensagem se qualifica.
3. O número de mensagens na fila com prioridade maior ou igual a *TriggerMsgPriority* era anteriormente, dependendo de *TriggerType*:
   - Zero (para tipo de acionador MQTT_FIRST)
   - Qualquer número (para tipo de acionador MQTT_EVERY)
   - *TriggerDepth* menos 1 (para tipo de acionador MQTT_DEPTH)

   Nota:

   a. Para filas locais não compartilhadas, o gerenciador de filas conta mensagens confirmadas e não confirmadas quando avalia se as condições para um evento do acionador existem. Consequentemente, um aplicativo pode ser iniciado quando não houver mensagens para ele recuperar porque as mensagens na fila não foram confirmadas. Nessa situação, considere usar a opção de espera com um *WaitInterval* adequado para que o aplicativo espere a chegada de suas mensagens.

   b. Para filas locais compartilhadas, o gerenciador de filas conta somente as mensagens confirmadas.
4. Para acionamento do tipo FIRST ou DEPTH, nenhum programa tem a fila do aplicativo aberta para a remoção de mensagens (ou seja, o atributo de fila local *OpenInputCount* é zero).

   Nota:

   a. Para filas compartilhadas, condições especiais se aplicam quando vários gerenciadores de filas têm monitores acionadores em execução com relação a uma fila. Nesta situação, se um ou mais gerenciadores de filas têm a fila aberta para entrada compartilhada, os critérios do acionador nos outros gerenciadores de filas são tratados como *TriggerType* MQTT_FIRST e *TriggerMsgPriority* zero. Quando todos os gerenciadores de filas fecham a fila para entrada, as condições acionadoras revertem para as condições especificadas na definição de fila.

      Um cenário de exemplo afetado por essa condição é vários gerenciadores de filas QM1, QM2 e QM3 com um monitor acionador em execução para uma fila de aplicativos A. A mensagem chega em A, atendendo às condições para o acionamento e uma mensagem acionadora é gerada na fila de inicialização. O monitor acionador em QM1 obtém a mensagem do acionador e aciona um aplicativo. O aplicativo acionado abre a fila do aplicativo para entrada compartilhada. A partir desse ponto, as condições acionadoras para uma fila do aplicativo são avaliadas como *TriggerType* MQTT_FIRST e *TriggerMsgPriority* zero nos gerenciadores de filas QM2 e QM3, até QM1 fechar a fila do aplicativo.

   b. Para filas compartilhadas, essa condição é aplicada a cada gerenciador de filas. Ou seja, um gerenciador de filas *OpenInputCount* para uma fila deve ser zero para que uma mensagem do acionador seja gerada para a fila por esse gerenciador de filas. No entanto, se qualquer gerenciador de filas no grupo de filas compartilhadas tiver a fila aberta usando a opção MQOO_INPUT_EXCLUSIVE, nenhuma mensagem do acionador será gerada para essa fila por qualquer um dos gerenciadores de filas no grupo de filas compartilhadas.

      A mudança no modo como as condições acionadoras são avaliadas ocorre quando o aplicativo acionado abre a fila para entrada. Em cenários em que há somente um monitor acionador em execução, outros aplicativos podem ter o mesmo efeito, pois abrem a fila do aplicativo para entrada de forma semelhante. Não importa se a fila do aplicativo foi aberta por um aplicativo iniciado por um monitor acionador ou por algum outro aplicativo; é o fato de que a fila está aberta para entrada em outro gerenciador de filas que causa a mudança de critérios do acionador.
5. No WebSphere MQ para z/OS, se a fila de aplicativos for uma com um atributo *Usage* de MQUS_NORMAL, as solicitações de obtenção para ela não serão inibidas (ou seja, o atributo da fila *InhibitGet* é MQQA_GET_ALLOWED).. Além disso, se a fila do aplicativo acionada for uma com um atributo *Usage* de MQUS_XMITQ, as solicitações get para ela não são inibidas.
6. Execute um dos dois procedimentos:

- O atributo *ProcessName* da fila local para a fila não está em branco e o objeto de definição de processo identificado por esse atributo foi criado ou
- O atributo *ProcessName* da fila local para a fila está todo em branco, mas a fila é uma fila de transmissão. Como a definição de processo é opcional, o atributo *TriggerData* também pode conter o nome do canal a ser iniciado. Nesse caso, a mensagem do acionador contém atributos com os valores a seguir:
  - *QName*: nome da fila
  - *ProcessName*: em branco
  - *TriggerData*: dados do acionador
  - *ApplType*: MQAT_UNKNOWN
  - *ApplId*: em branco
  - *EnvData*: em branco
  - *UserData*: em branco

7. Uma fila de inicialização foi criada e foi especificada no atributo *InitiationQName* da fila local. Além disso:
   - Solicitações get não são inibidas para a fila de inicialização (ou seja, o atributo de fila *InhibitGet* é MQQA_GET_ALLOWED).
   - Solicitações put não devem ser inibidas para a fila de inicialização (ou seja, o atributo de fila *InhibitPut* deve ser MQQA_PUT_ALLOWED).
   - O atributo *Usage* da fila de inicialização deve ser MQUS_NORMAL.
   - Em ambientes nos quais filas dinâmicas são suportadas, a fila de inicialização não deve ser uma fila dinâmica que foi marcada como excluída logicamente.
8. Um monitor acionador atualmente tem a fila de inicialização aberta para remover mensagens (ou seja, o atributo *OpenInputCount* da fila local é maior que zero).
9. O controle do acionador (atributo *TriggerControl* da fila local) para a fila do aplicativo está configurado para MQTC_ON. Para fazer isso, configure o atributo *trigger* ao definir a fila ou use o comando ALTER QLOCAL.
10. O tipo de acionador (atributo *TriggerType* da fila local) não é MQTT_NONE.

    Se todas as condições necessárias forem atendidas e a mensagem que causou a condição do acionador for colocada como parte de uma unidade de trabalho, a mensagem do acionador não será disponibilizada para recuperação pelo aplicativo monitor acionador até a unidade de trabalho ser concluída, se a unidade de trabalho for confirmada ou, para o tipo de acionador MQTT_FIRST ou MQTT_DEPTH, restaurada.
11. Uma mensagem adequada é colocado na fila, para um *TriggerType* de MQTT_FIRST ou MQTT_DEPTH e a fila:
    - Não estava vazia anteriormente (MQTT_FIRST) ou
    - Tinha *TriggerDepth* ou mais mensagens (MQTT_DEPTH)

    e as condições “2” na página 340 a “10” na página 341 (excluindo a “3” na página 340) são satisfeitas, se, no caso de MQTT_FIRST, um intervalo suficiente (atributo *TriggerInterval* do gerenciador de filas) tiver decorrido desde que a última mensagem do acionador foi gravada nesta fila.

    Isso é para permitir um servidor da fila que é finalizado antes de processar todas as mensagens na fila. O propósito do intervalo do acionador é reduzir o número de mensagens do acionador duplicadas que são geradas.

    Nota: Se você para e reinicia o gerenciador de filas, o *TriggerInterval timer* é reconfigurado. Há uma pequena janela durante a qual é possível produzir duas mensagens do acionador. A janela existe quando o atributo trigger da fila estiver configurado para ativado ao mesmo tempo que uma mensagem chega e a fila não estava anteriormente vazia (MQTT_FIRST) ou tinha *TriggerDepth* ou mais mensagens (MQTT_DEPTH).

12. O único aplicativo que atende uma fila emite uma chamada MQCLOSE para um *TriggerType* igual a MQTT_FIRST ou MQTT_DEPTH e há pelo menos:

    - Um (MQTT_FIRST) ou
    - *TriggerDepth* (MQTT_DEPTH)

    mensagens na fila de prioridade suficiente (condição “2” na página 340) e as condições “6” na página 340 a “10” na página 341 também são satisfeitas.

    Isso é para permitir um servidor da fila que emite uma chamada MQGET, encontra a fila vazia e assim finalize; no entanto, no intervalo entre as chamadas MQGET e MQCLOSE, uma ou mais mensagens chegam.

    Nota:

    a. Se o programa que está atendendo a fila do aplicativo não recuperar todas as mensagens, isso pode causar um loop fechado. Toda vez que o programa fechar a fila, o gerenciador de filas criará outra mensagem do acionador que fará o monitor acionador iniciar o programa do servidor novamente.

    b. Se o programa que está atendendo a fila do aplicativo recuperar sua solicitação get (ou se o programa for encerrado de forma anormal) antes de fechar a fila, o mesmo acontece. No entanto, se o programa fechar a fila antes de recuperar a solicitação get e a fila estiver vazia, nenhuma mensagem do acionador será criado.

    c. Para evitar que esse loop ocorra, use o campo *BackoutCount* de MQMD para detectar mensagens que são restauradas repetidamente. Para obter mais informações, consulte “Mensagens que são restauradas” na página 38.

13. As condições a seguir são satisfeitas usando MQSET ou um comando:

    a. - *TriggerControl* é mudado para MQTC_ON ou
       - *TriggerControl* já é MQTC_ON e o valor de um *TriggerType*, *TriggerMsgPriority* ou *TriggerDepth* (se relevante) é mudado,

       e há pelo menos:

       - Uma (MQTT_FIRST ou MQTT_EVERY) ou
       - *TriggerDepth* (MQTT_DEPTH)

       mensagens na fila de prioridade suficiente (condição “2” na página 340) e as condições “4” na página 340 a “10” na página 341 (excluindo “8” na página 341) também são satisfeitas.

       Isso é para permitir que um aplicativo ou operador mude os critérios de acionamento, quando as condições para que um acionador ocorra já estiverem satisfeitas.

    b. O atributo de fila *InhibitPut* de uma fila de inicialização muda de MQQA_PUT_INHIBITED para MQQA_PUT_ALLOWED e há pelo menos:

       - Uma (MQTT_FIRST ou MQTT_EVERY) ou
       - *TriggerDepth* (MQTT_DEPTH)

       mensagens de prioridade suficiente (condição “2” na página 340) em qualquer uma das filas para as quais essa é a fila de inicialização e as condições “4” na página 340 a “10” na página 341 também são satisfeitas. (Uma mensagem do acionador é gerada para cada uma dessas filas que esteja satisfazendo as condições.)

       Isso é para permitir que as mensagens do acionador não sejam geradas por causa da condição MQQA_PUT_INHIBITED na fila de inicialização, mas essa condição agora foi mudada.

    c. O atributo de fila *InhibitGet* de uma fila do aplicativo muda de MQQA_GET_INHIBITED para MQQA_GET_ALLOWED e há pelo menos:

       - Uma (MQTT_FIRST ou MQTT_EVERY) ou
       - *TriggerDepth* (MQTT_DEPTH)

mensagens de prioridade suficiente (condição “2” na página 340) na fila e as condições “4” na página 340 a “10” na página 341, excluindo a “5” na página 340, também são satisfeitas.

Isso permite que os aplicativos sejam acionados somente quando puderem recuperar mensagens da fila do aplicativo.

d. Um aplicativo monitor acionador emite uma chamada MQOPEN para entrada a partir de uma fila de inicialização e há pelo menos:

- Uma (MQTT_FIRST ou MQTT_EVERY) ou
- *TriggerDepth* (MQTT_DEPTH)

mensagens de prioridade suficiente (condição “2” na página 340) em qualquer uma das filas do aplicativo para a qual esta é a fila de inicialização e as condições “4” na página 340 a “10” na página 341 (excluindo a “8” na página 341) também são satisfeitas e nenhum outro aplicativo tem a fila de inicialização aberta para entrada (uma mensagem do acionador será gerada para cada fila que satisfaça as condições).

Isso é para permitir que mensagens cheguem nas filas enquanto o monitor acionador não está em execução e o gerenciador de filas seja reiniciado e as mensagens do acionador (que são persistentes) sejam perdidas.

14. MSGDLVSQ está configurado corretamente. Se você configurar MSGDLVSQ=FIFO, as mensagens serão entregues à fila em um modo Primeiro a entrar, primeiro a sair. A prioridade da mensagem é ignorada e a prioridade padrão da fila é designada à mensagem. Se *TriggerMsgPriority* for configurado para um valor maior que a prioridade padrão da fila, nenhuma mensagem será acionada. Se *TriggerMsgPriority* for configurado igual ou menor que a prioridade padrão da fila, o acionamento ocorrerá para tipo FIRST, EVERY e DEPTH. Para obter informações sobre esses tipos, consulte a descrição do campo *TriggerType* campo em “Controlando eventos acionadores” na página 343.

    Se você configurar MSGDLVSQ=PRIORITY e a prioridade da mensagem for igual ou maior que o campo *TriggerMsgPriority*, mensagens são *contadas* somente no sentido de um evento acionador. Nesse caso, o acionamento ocorre para o tipo FIRST, EVERY e DEPTH. Como um exemplo, se você colocar 100 mensagens de prioridade mais baixa que *TriggerMsgPriority*, a profundidade da fila efetiva para propósitos de acionamento ainda será zero. Se, então, colocar outra mensagem na fila, mas, desta vez, a prioridade for maior ou igual a *TriggerMsgPriority*, a profundidade da fila efetiva aumenta de zero para um e a condição para *TriggerType* FIRST é satisfeita.

Nota:

1. A partir da etapa “12” na página 342 (em que as mensagens do acionador são geradas como resultado de algum outro evento diferente de uma mensagem que chega na fila do aplicativo), a mensagem do acionador não será colocada como parte de uma unidade de trabalho. Além disso, se *TriggerType* for MQTT_EVERY e se houver uma ou mais mensagens na fila do aplicativo, somente uma mensagem do acionador será gerada.
2. Se o WebSphere MQ segmentar uma mensagem durante MQPUT, um evento acionador não será processado até que todos os segmentos tenham sido colocados com sucesso na fila. No entanto, quando os segmentos de mensagens estiverem na fila, o WebSphere MQ os tratará como mensagens individuais para fins de acionamento. Por exemplo, uma única mensagem lógica dividida em três partes faz com que somente um evento do acionador seja processado quando inicialmente passa por MQPUT e é segmentada. No entanto, cada um dos três segmentos faz com que seus próprios eventos acionadores sejam processados conforme eles são movidos pela rede do WebSphere MQ .

Controlando eventos acionadores

Controle os eventos acionadores usando alguns dos atributos que definem sua fila do aplicativo. Essas informações também fornecem exemplos do uso dos tipos de acionadores: EVERY, FIRST e DEPTH.

É possível ativar e desativar o acionamento e é possível selecionar o número ou a prioridade das mensagens que contam para um evento acionador. Há uma descrição integral desses atributos em Atributos de objetos.

Os atributos relevantes são:

TriggerControl
Use esse atributo para ativar e desativar o acionamento de uma fila do aplicativo.

TriggerMsgPriority
A prioridade mínima que uma mensagem deve ter para que conte com relação a um evento acionador. Se uma mensagem de prioridade menor que *TriggerMsgPriority* chegar na fila de aplicativos, o gerenciador de filas ignorará a mensagem quando determinar se uma mensagem do acionador deve ser criada ou não. Se *TriggerMsgPriority* estiver configurado como zero, todas as mensagens serão contadas para um evento acionador.

TriggerType
Além do tipo de acionador NONE (que desativa o acionamento exatamente como configurar *TriggerControl* como OFF), é possível usar os tipos de acionador a seguir para configurar a sensibilidade de uma fila para acionar eventos:

EVERY	Um evento acionador ocorre toda vez que uma mensagem chega à fila do aplicativo. Use esse tipo de acionador se desejar diversas instâncias de um aplicativo iniciadas.
PRIM	Ocorre um evento acionador somente quando o número de mensagens na fila do aplicativo é mudado de zero para um. Use esse tipo de acionador se desejar que um programa de serviço seja iniciado quando a primeira mensagem chegar em uma fila, continue até que não haja mais mensagens a serem processadas, em seguida, termine. Deve-se sempre processar a fila até que esteja vazia. Consulte também “Caso especial de tipo de acionador FIRST” na página 345.
PRO	Um evento acionador ocorre apenas quando o número de mensagens na fila de aplicativos atinge o valor do atributo *TriggerDepth* .. Um uso típico desse tipo de acionamento é iniciar um programa quando todas as respostas a um conjunto de solicitações forem recebidas. **Acionamento por profundidade:** Com o acionamento por profundidade, o gerenciador de filas desativa o acionamento (usando o atributo < xph> < pv>TriggerControl< /pv> < /xph>) após criar uma mensagem do acionador. Seu aplicativo deve reativar o próprio acionamento (usando a chamada MQSET) após isso ocorrer. A ação de desativar o acionamento não está sob o controle do ponto de sincronização, portanto, o acionamento não pode ser reativado restaurando uma unidade de trabalho. Se um programa restaurar uma solicitação put que causou um evento acionador ou se o programa for encerrado de forma anormal, deve-se reativar o acionamento usando a chamada MQSET ou o comando ALTER QLOCAL.

TriggerDepth
O número de mensagens em uma fila que causa um evento acionador ao usar o acionamento por profundidade.

As condições que devem ser satisfeitas para um gerenciador de filas criar uma mensagem do acionador estão descritas em “Condições para um evento acionador” na página 339.

Exemplo do uso do tipo de acionador EVERY

Considere um aplicativo que gere solicitações para seguro de carro. O aplicativo pode enviar mensagens de solicitação a diversas seguradoras, especificando a mesma fila de resposta toda vez. Pode configurar

um acionador do tipo EVERY nessa fila de resposta de forma que toda vez que uma resposta chegar, a resposta possa acionar uma instância do servidor para processar a resposta.

Exemplo do uso do tipo de acionador FIRST

Considere uma organização com inúmeras filiais, sendo que cada uma transmite detalhes dos negócios dos dias para a matriz. Todas elas fazem isso ao mesmo tempo, no fim do dia útil, e na matriz existe um aplicativo que processa os detalhes de todas as filiais. A primeira mensagem a chegar na matriz poderia causar um evento acionador que iniciaria esse aplicativo. Esse aplicativo continuaria o processamento até que não houvesse mais mensagens em sua fila.

Exemplo do uso do tipo de acionador DEPTH

Considere um aplicativo de agência de viagem que crie uma única solicitação para confirmar uma reserva de voo, confirmar uma reserva de um quarto de hotel, alugar um carro ou solicitar alguns traveler's checks. O aplicativo poderia separar esses itens em quatro mensagens de solicitação, enviando cada uma a um destino separado. Poderia configurar um acionador do tipo DEPTH em sua fila de resposta (com a profundidade configurada para o valor 4), de forma que seja reiniciado somente quando todas as quatro respostas tiverem chegado.

Se outra mensagem (possivelmente de uma solicitação diferente) chegar à fila de resposta antes da última das quatro respostas, o aplicativo de solicitação será acionado antecipadamente. Para evitar isso, ao usar o acionamento DEPTH para coletar diversas respostas a uma solicitação, sempre use uma nova fila de resposta para cada solicitação.

Caso especial de tipo de acionador FIRST

Com o tipo de acionador FIRST, se já houver uma mensagem na fila do aplicativo quando outra mensagem chegar, o gerenciador de filas normalmente não cria outra mensagem do acionador.

No entanto, o aplicativo que atende a fila pode não abrir a fila efetivamente (por exemplo, o aplicativo pode ser finalizado, possivelmente devido a um problema do sistema). Se um nome de aplicativo incorreto tiver sido colocado no objeto de definição de processo, o aplicativo que atende a fila não captará nenhuma das mensagens. Nessas situações, se outra mensagem chegar à fila do aplicativo, não haverá servidor em execução para processar essa mensagem (e qualquer outra mensagem na fila).

Para lidar com isso, o gerenciador de filas cria mensagens adicionais do acionador sob as circunstâncias a seguir:

- Se outra mensagem chegar na fila do aplicativo, mas somente se um intervalo de tempo predefinido tiver decorrido desde quando o gerenciador de filas criou a última mensagem do acionador para essa fila. Este intervalo de tempo é definido no atributo *TriggerInterval*do gerenciador de filas. Seu valor padrão é 999 999 999 milissegundos.
- No WebSphere MQ para z/OS, as filas de aplicativos que nomeam uma fila de inicialização aberta são varridas periodicamente Se *TRIGINT* milissegundos tiverem passado desde que a última mensagem do acionador foi enviada e a fila atender às condições para um evento acionador e CURDEPTH for maior que zero, uma mensagem do acionador será gerada. Esse processo é chamado de acionamento backstop.

Considere os pontos a seguir ao decidir sobre um valor para o intervalo do acionador a ser usado em seu aplicativo:

- Se você configurar *TriggerInterval* para um valor baixo e não houver nenhum aplicativo que atenda à fila do aplicativo, o tipo de acionador FIRST pode se comportar como o tipo de acionador EVERY. Isso depende da taxa em que as mensagens estão sendo colocadas na fila do aplicativo, que, por sua vez, pode depender de outra atividade do sistema. Isso ocorre porque, se o intervalo do acionador for muito pequeno, outra mensagem do acionador será gerada toda vez que uma mensagem for colocada na fila do aplicativo, embora o tipo de acionador seja FIRST, não EVERY. (O tipo de acionador FIRST com um intervalo do acionador igual a zero é equivalente ao tipo de acionador EVERY.)

- No WebSphere MQ para z/OS , se você configurar *TRIGINT* para um valor baixo e não houver nenhum aplicativo que atenda a fila do aplicativo FIRST do tipo de acionador, o acionamento de backstop gerará uma mensagem do acionador cada vez que a varredura periódica de filas de aplicativos que nomeiam filas de inicialização abertas ocorrer.
- Se uma unidade de trabalho for restaurada (consulte Mensagens do acionador e unidades de trabalho) e o intervalo do acionador tiver sido configurado para um valor alto (ou o valor padrão), uma mensagem do acionador será gerada quando a unidade de trabalho for restaurada. No entanto, se você tiver configurado o intervalo do acionador para um valor baixo ou para zero (fazendo com que o tipo de acionador FIRST se comporte como o tipo de acionador EVERY), muitas mensagens do acionador poderão ser geradas. Se a unidade de trabalho for restaurada, todas as mensagens do acionador ainda serão disponibilizadas. O número de mensagens do acionador que são geradas depende do intervalo do acionador. Se o intervalo do acionador for configurado para zero, o número máximo de mensagens será gerado.

Projetando um aplicativo que usa filas acionadas

Você viu como configurar, controlar e acionar seus aplicativos. Aqui estão algumas dicas a se considerar ao projetar seu aplicativo.

Mensagens do acionador e unidades de trabalho

As mensagens do acionador criadas devido a eventos acionadores que não são parte de uma unidade de trabalho são colocadas na fila de inicialização, fora de qualquer unidade de trabalho, sem dependência de quaisquer outras mensagens e estão disponíveis para recuperação pelo monitor acionador imediatamente.

As mensagens do acionador criadas devido a eventos do acionador que fazem parte de uma unidade de trabalho são disponibilizadas na fila de inicialização quando a UOW é resolvida se a unidade de trabalho for confirmada ou restaurada

Se o gerenciador de filas falhar ao colocar uma mensagem do acionador em uma fila de inicialização, ele será colocado na fila de devoluções (mensagem não entregue).

Nota:

1. O gerenciador de filas conta mensagens confirmadas e não confirmadas quando ele avalia se as condições para um evento acionador existem.

   Com o acionamento do tipo FIRST ou DEPTH, as mensagens do acionador são disponibilizadas, mesmo se a unidade de trabalho for restaurada de modo que uma mensagem do acionador esteja sempre disponível quando as condições necessárias forem atendidas. Por exemplo, considere uma solicitação put em uma unidade de trabalho para uma fila que é acionada com tipo de acionador FIRST. Isso faz com que o gerenciador de filas crie uma mensagem do acionador. Se uma outra solicitação put ocorrer, a partir de outra unidade de trabalho, isso não causa outro evento acionador, porque o número de mensagens na fila do aplicativo foi mudado de um para dois, o que não atende às condições de um evento acionador. Agora, se a primeira unidade de trabalho for restaurada, mas a segunda for confirmada, uma mensagem do acionador ainda será criada.

   No entanto, isso significa que as mensagens do acionador, às vezes, são criadas quando as condições para um evento acionador *não* são atendidas. Aplicativos que usam o acionamento devem sempre estar preparados para lidar com esta situação. É recomendado que você use a opção aguardar com a chamada MQGET, configurando o *WaitInterval* para um valor adequado.

   Mensagens do acionador criadas são sempre disponibilizadas se a unidade de trabalho estiver restaurada ou confirmada.
2. Para filas compartilhadas locais (ou seja, filas compartilhadas em um grupo de filas compartilhadas) o gerenciador de filas conta somente as mensagens confirmadas.

Obtendo mensagens de uma fila acionada

Ao projetar aplicativos que usam o acionamento, esteja ciente de que pode haver um atraso entre um monitor acionador iniciando um programa e outras mensagens se tornando disponíveis na fila do aplicativo. Isso pode acontecer quando a mensagem que causa o evento acionador é confirmada antes das outras.

Para dar tempo de as mensagens chegarem, sempre use a opção de espera ao usar a chamada MQGET para remover mensagens de uma fila para a qual as condições acionadoras são configuradas. O *WaitInterval* deve ser suficiente para permitir o tempo mais longo razoável entre uma mensagem sendo colocada e essa chamada put sendo confirmada. Se a mensagem estiver chegando a partir de um gerenciador de filas remotas, esse tempo será afetado por:

- O número de mensagens que são colocadas antes de serem confirmadas
- A velocidade e a disponibilidade do link de comunicação
- Os tamanhos das mensagens

Para obter um exemplo de uma situação em que é necessário usar a chamada MQGET com a opção de espera, considere o mesmo exemplo que usamos ao descrever unidades de trabalho. Essa era uma solicitação put em uma unidade de trabalho para uma fila que é acionada com tipo de acionador FIRST. Esse evento faz com que o gerenciador de filas crie uma mensagem do acionador. Se uma outra solicitação put ocorrer, a partir de outra unidade de trabalho, isso não causa outro evento acionador porque o número de mensagens na fila do aplicativo não foi mudado de zero para um. Agora, se a primeira unidade de trabalho for restaurada, mas a segunda for confirmada, uma mensagem do acionador ainda será criada. Portanto, a mensagem do acionador é criada no momento em que a primeira unidade de trabalho é restaurada. Se houver um atraso significativo antes de a segunda mensagem ser confirmada, o aplicativo acionado poderá precisar esperar por ela.

Com o acionamento do tipo DEPTH, um atraso pode ocorrer mesmo se todas as mensagens relevantes forem finalmente confirmadas. Suponha que o atributo da fila *TriggerDepth* tenha o valor 2. Quando duas mensagens chegam na fila, a segunda faz com que uma mensagem acionadora seja criada. No entanto, se a segunda mensagem for a primeira a ser confirmada, será nessa altura que a mensagem do acionador se tornará disponível. O monitor acionador inicia o programa do servidor, mas o programa pode recuperar apenas a segunda mensagem até que a primeira seja confirmada. Portanto, o programa pode precisar esperar que a primeira mensagem seja disponibilizada.

Projete seu aplicativo para que ele seja finalizado, se nenhuma mensagem estiver disponível para recuperação quando seu intervalo de espera expirar. Se uma ou mais mensagens chegarem posteriormente, conte com o aplicativo que estava sendo acionado novamente para processá-las. Esse método evita que os aplicativos fiquem inativos e o uso de recursos desnecessariamente.

Processamento da fila de inicialização por monitores acionadores

Para um gerenciador de filas, um monitor acionador é semelhante a qualquer outro aplicativo que atende uma fila. No entanto, um monitor acionador serve filas de inicialização.

Um monitor acionador é geralmente um programa de execução contínua. Quando uma mensagem do acionador chega em uma fila de inicialização, o monitor acionador recupera essa mensagem. Ele utiliza as informações na mensagem para emitir um comando para iniciar o aplicativo que deve processar as mensagens na fila do aplicativo.

O monitor acionador deve transmitir informações suficientes para o programa que está iniciando para que o programa possa executar as ações corretas na fila do aplicativo correto.

Um iniciador de canal é um exemplo de um tipo especial de monitor acionador para agentes do canal de mensagem. Nessa situação, no entanto, deve-se usar o tipo de acionador FIRST ou DEPTH.

Monitores acionadores em sistemas UNIX ou Windows

Este tópico contém informações sobre monitores acionadores fornecidos em sistemas UNIX e Windows ..

Os monitores acionadores a seguir são fornecidos para o ambiente do servidor:

amqstrg0
Este é um monitor acionador de amostra que fornece um subconjunto da função fornecida pelo **runmqtrm**. Consulte “Programas de amostra para plataformas distribuídas” na página 98 para obter mais informações sobre amqstrg0.

runmqtrm
A sintaxe desse comando é **runmqtrm** [*-m QMgrName*] [*-q InitQ*], em que QMgrName é o gerenciador de filas e InitQ é a fila de inicialização. A fila padrão é SYSTEM.DEFAULT.INITIATION.QUEUE no gerenciador de filas padrão. Ela chama programas para as mensagens do acionador apropriadas. Esse monitor acionador suporta o tipo de aplicativo padrão.

A sequência de comandos transmitida pelo monitor acionador para o sistema operacional é construída da forma a seguir:

1. O *ApplId* na definição de PROCESS relevante (se criado)
2. A estrutura MQTMC2, colocada em aspas duplas
3. O *EnvData* na definição de PROCESS relevante (se criado)

em que *ApplId* é o nome do programa a ser executado como seria inserido na linha de comandos.

O parâmetro transmitido é a estrutura de caracteres MQTMC2. Uma sequência de comandos que possui esta sequência é chamada, exatamente conforme fornecida, entre aspas duplas, na ordem em que o comando do sistema a aceitará como um parâmetro.

O monitor acionador não verifica se há outra mensagem na fila de inicialização até a conclusão do aplicativo que acabou de iniciar. Se o aplicativo tiver muito processamento a fazer, o monitor acionador não poderá acompanhar o número de mensagens do acionador que chegarem. Você tem duas opções:

- Ter mais monitores acionadores em execução
- Executar os aplicativos iniciados em segundo plano

Se você tiver mais monitores acionadores em execução, será possível controlar o número máximo de aplicativos que podem ser executados a qualquer momento. Se você executar aplicativos em segundo plano, não haverá nenhuma restrição imposta pelo WebSphere MQ no número de aplicativos que podem ser executados

Para executar o aplicativo iniciado no segundo plano em sistemas Windows , no campo *ApplId* , prefixe o nome do aplicativo com um comando START. Por exemplo:

```
START ?B AMQSECHA
```

Para executar o aplicativo iniciado no plano de fundo em sistemas UNIX , coloque um & no final do *EnvData* da definição PROCESS

Nota: Quando um caminho do Windows possui espaços como parte do nome do caminho, eles devem ser colocados entre aspas (") para assegurar que ele seja manipulado como um único argumento Por exemplo, `" C:\Program Files\Application Directory\Application.exe"`.

A seguir, está um exemplo de uma sequência APPLICID em que o nome do arquivo inclui espaços como parte do caminho:

```
START "" /B "C:\Program Files\Application Directory\Application.exe"
```

A sintaxe do comando START Windows no exemplo inclui uma sequência vazia entre aspas duplas. START especifica que o primeiro argumento entre aspas será tratado como o título do novo comando. Para assegurar que o Windows não erro o caminho do aplicativo para um argumento 'title', inclua uma sequência de título entre aspas duplas para o comando antes do nome do aplicativo

Os seguintes monitores acionadores são fornecidos para o cliente WebSphere MQ :

runmqtmc
Isso é o mesmo que runmqtrm, exceto que ele é vinculado às bibliotecas do cliente MQI do WebSphere MQ .

Para CICS
O monitor acionador amqltmc0 é fornecido para CICS. Ele funciona da mesma maneira que o monitor acionador padrão, runmqtrm, mas você o executa de uma maneira diferente e aciona transações CICS .

Este tópico aplica-se apenas aos sistemas Windows, UNIXe Linux

Ele é fornecido como um programa CICS ; defina-o com um nome de transação de 4 caracteres Insira o nome de 4 caracteres para iniciar o monitor acionador. Ele usa o gerenciador de filas padrão (conforme nomeado no arquivo qm.ini ou no WebSphere MQ para Windows, o registro) e o SYSTEM.CICS.INITIATION.QUEUE.

Se você desejar usar um gerenciador de filas ou uma fila diferente, construa a estrutura do monitor acionador MQTMC2 : isso requer que você grave um programa usando a chamada EXEC CICS START, porque a estrutura é muito longa para ser incluída como um parâmetro. Em seguida, passe a estrutura MQTMC2 como dados para a solicitação START para o monitor acionador.

Ao usar a estrutura MQTMC2, é necessário fornecer somente os parâmetros *StrucId*, *Version*, *QName*e *QMgrName* para o monitor acionador conforme, pois ele não faz referência a nenhum outro campo.

As mensagens são lidas a partir da fila de inicialização e usadas para iniciar transações CICS , usando EXEC CICS START, assumindo que APPL_TYPE na mensagem do acionador seja MQAT_CICS. A leitura de mensagens da fila de inicialização é executada sob o controle de ponto de sincronização do CICS .

As mensagens são geradas quando o monitor é iniciado e parado e quando ocorre um erro. Essas mensagens são enviadas à fila de dados temporários CSMT.

A seguir estão as versões disponíveis do monitor acionador:

Versão	**Usar**
amqltmc0	TXSeries para AIX, HP-UXe Sun Solaris Versão 5.1
amqltmc4	TXSeries para Windows, Versão 5.1
amqltmcc	Versão de limite do cliente do monitor acionador CICS

Se precisar de um monitor acionador para outros ambientes, escreva um programa que possa processar as mensagens do acionador que o gerenciador de filas coloca nas filas de inicialização. Esse programa deve executar as ações a seguir:

1. Use a chamada MQGET para esperar a chegada de uma mensagem na fila de inicialização.
2. Examine os campos da estrutura MQTM da mensagem do acionador para localizar o nome do aplicativo a ser iniciado e o ambiente no qual ele é executado.
3. Emita um comando inicial específico do ambiente.
4. Converta a estrutura MQTM para a estrutura MQTMC2, se necessário.
5. Passe a estrutura MQTMC2 ou MQTM para o aplicativo iniciado. Ela pode conter dados do usuário.
6. Associe à sua fila do aplicativo o aplicativo que deve atender àquela fila. É possível fazer isso denominando o objeto de definição de processo (se criado) no atributo *ProcessName* da fila.

Para obter mais informações sobre a interface do monitor acionador, consulte MQTMC2.

Propriedades de mensagens do acionador

Os tópicos a seguir descrevem algumas outras propriedades de mensagens do acionador.

- “Persistência e prioridade de mensagens do acionador” na página 350

Persistência e prioridade de mensagens do acionador

Mensagens do acionador não são persistentes porque não há requisito para que sejam assim.

No entanto, as condições para gerar eventos de acionamento persistem, de modo que as mensagens do acionador são geradas sempre que essas condições forem atendidas. Se uma mensagem do acionador for perdida, a existência continuada da mensagem do aplicativo na fila do aplicativo garante que o gerenciador de filas gere uma mensagem de acionador assim que todas as condições forem atendidas.

Se uma unidade de trabalho for retrocedida, todas as mensagens do acionador que ela tiver gerado sempre serão entregues.

As mensagens do acionador usam a prioridade padrão da fila de inicialização.

Reinicialização do gerenciador de filas e mensagens do acionador

Após a reinicialização de um gerenciador de filas, quando uma fila de inicialização for aberta da próxima vez para entrada, uma mensagem do acionador poderá ser colocada nessa fila de inicialização se uma fila do aplicativo associada a ele tiver mensagens e estiver definida para acionamento.

Mensagens do acionador e mudanças nos atributos do objeto

Mensagens do acionador são criadas de acordo com os valores dos atributos do acionador em vigor no momento do evento acionador.

Se a mensagem do acionador não for disponibilizada para um monitor acionador até mais tarde (porque a mensagem que causou sua geração foi colocada em uma unidade de trabalho), quaisquer mudanças nos atributos do acionador não terão efeito na mensagem do acionador enquanto isso. Especificamente, desativar o acionamento não evita que uma mensagem do acionador seja disponibilizada quando for criada. Além disso, a fila do aplicativo pode não existir mais no momento que a mensagem do acionador for disponibilizada.

Formato de mensagens do acionador

O formato de uma mensagem do acionador é definido pela estrutura MQTM.

Ela tem os campos a seguir, que o gerenciador de filas preenche ao criar a mensagem do acionador, usando informações nas definições de objeto da fila do aplicativo e do processo associado a essa fila:

StrucId
O identificador de estruturação.

Version
A versão da estrutura.

QName
O nome da fila do aplicativo na qual o evento acionador ocorreu. Quando o gerenciador de filas cria uma mensagem do acionador, ele preenche esse campo usando o atributo *QName* da fila do aplicativo.

ProcessName
O nome do objeto de definição de processo associado à fila do aplicativo. Quando o gerenciador de filas cria uma mensagem do acionador, ele preenche esse campo usando o atributo *ProcessName* da fila do aplicativo.

TriggerData
Um campo de formato livre para ser usado pelo monitor acionador. Quando o gerenciador de filas cria uma mensagem do acionador, ele preenche esse campo usando o atributo *TriggerData* da fila do aplicativo. Em qualquer produto WebSphere MQ , exceto WebSphere MQ para z/OS, esse campo pode ser usado para especificar o nome do canal a ser acionado.

ApplType
O tipo do aplicativo que o monitor acionador deve iniciar. Quando o gerenciador de filas cria uma mensagem do acionador, ele preenche esse campo usando o atributo *ApplType* do objeto de definição de processo identificado em *ProcessName*.

ApplId
Uma sequência de caracteres que identifica o aplicativo que o monitor acionador deve iniciar. Quando o gerenciador de filas cria uma mensagem do acionador, ele preenche esse campo usando o atributo *ApplId* do objeto de definição de processo identificado em *ProcessName*. Ao usar o monitor acionador CKTI ou CSQQTRMN fornecido pelo WebSphere MQ para z/OS, o atributo *ApplId* do objeto de definição de processo é um identificador de transação CICS ou IMS .

EnvData
Um campo de caractere que contém dados relacionados ao ambiente para uso pelo monitor acionador. Quando o gerenciador de filas cria uma mensagem do acionador, ele preenche esse campo usando o atributo *EnvData* do objeto de definição de processo identificado em *ProcessName*. Os monitores acionadores fornecidos pelo WebSphere MQ for z/OS(CKTI ou CSQQTRMN) não usam esse campo, mas outros monitores acionadores podem optar por usá-lo..

UserData
Um campo de caractere que contém dados do usuário para uso pelo monitor acionador. Quando o gerenciador de filas cria uma mensagem do acionador, ele preenche esse campo usando o atributo *UserData* do objeto de definição de processo identificado em *ProcessName*. Esse campo pode ser usado para especificar o nome do canal a ser acionado.

Há uma descrição integral da estrutura da mensagem do acionador em MQTM.

Quando o acionamento não funciona

Um programa não é acionado se o monitor acionador não puder iniciar o programa ou o gerenciador de filas não puder entregar a mensagem do acionador. Por exemplo, o applid no objeto do processo deve especificar que o programa deve ser iniciado em segundo plano; caso contrário, o monitor acionador não pode iniciar o programa.

Se uma mensagem do acionador for criada, mas não puder ser colocada na fila de inicialização (por exemplo, porque a fila está cheia ou o comprimento da mensagem do acionador é maior que o comprimento máximo de mensagem especificado para a fila de inicialização), a mensagem do acionador será colocada na fila de devoluções (da mensagem não entregue).

Se a operação put para a fila de mensagens não entregues não puder ser concluída com êxito, a mensagem do acionador será descartada e uma mensagem de aviso será enviada ao console do z/OS ou ao operador do sistema ou será colocada no log de erros.

Colocar a mensagem do acionador na fila de mensagens não entregues pode gerar uma mensagem do acionador para essa fila. Essa segunda mensagem do acionador será descartada se incluir uma mensagem na fila de mensagens não entregues.

Se o programa for acionado com êxito, mas encerrar de forma anormal antes de receber a mensagem da fila, use um utilitário de rastreamento (por exemplo, CICS AUXTRACE se o programa estiver em execução no CICS) para localizar a causa da falha...

Trabalhando com MQI e clusters

Há opções especiais em chamadas e códigos de retorno relacionadas ao armazenamento em cluster.

Use os links a seguir para descobrir mais sobre as opções disponíveis nas chamadas e nos códigos de retorno para uso com clusters:

- “MQOPEN e clusters” na página 352
- “MQPUT, MQPUT1 e clusters” na página 353
- “MQINQ e clusters” na página 354
- “MQSET e clusters” na página 354

- “Códigos de retorno” na página 355

Conceitos relacionados

“Visão geral da Message Queue Interface” na página 198
Aprenda sobre os componentes de Message Queue Interface (MQI).

“Conectando-se e desconectando-se de um gerenciador de filas” na página 209
Para usar WebSphere MQ serviços de programação, um programa deve ter uma conexão com um gerenciador de filas. Use essas informações para aprender como conectar-se e desconectar-se de um gerenciador de filas.

“Abrindo e fechando objetos” na página 218
Estas informações fornecem um insight sobre como abrir e fechar objetos do WebSphere MQ .

“Colocando mensagens em uma fila” na página 228
Use estas informações para aprender como colocar mensagens em uma fila.

“Obtendo mensagens de uma fila” na página 243
Use estas informações para aprender como obter mensagens de uma fila.

“Consultando e configurando atributos de objeto” na página 324
Atributos são as propriedades que definem as características de um objeto WebSphere MQ .

“Confirmando e fazendo backup de unidades de trabalho” na página 327
Essas informações descrevem como confirmar e restaurar quaisquer operações de GET e PUT que ocorreram em uma unidade de trabalho.

“Iniciando aplicativos IBM WebSphere MQ usando acionadores” na página 333
Aprenda sobre acionadores e como iniciar aplicativos IBM WebSphere MQ usando acionadores.

MQOPEN e clusters

A fila na qual uma mensagem é colocada, ou a partir da qual é lida, quando uma fila de clusters é aberta depende da chamada MQOPEN.

Selecionando a fila de destino

Se não fornecer um nome de gerenciador de filas no descritor de objeto, MQOD, o gerenciador de filas seleciona o gerenciador de filas para o qual enviar a mensagem. Se você fornecer um nome de gerenciador de filas no descritor de objeto, então, mensagens sempre serão enviadas ao gerenciador de filas selecionado.

Se o gerenciador de filas estiver selecionando o gerenciador de filas de destino, a seleção dependerá das opções de ligação, MQOO_BIND_* e se uma fila local existe. Se houver uma instância local da fila, sempre será aberta preferencialmente a uma instância remota, a menos que o atributo CLWLUSEQ esteja configurado para ANY. Caso contrário, a seleção depende das opções de ligação. MQOO_BIND_ON_OPEN ou MQOO_BIND_ON_GROUP deve ser especificado ao usar grupos de mensagens com clusters para assegurar que todas as mensagens no grupo sejam processadas no mesmo destino.

Se o gerenciador de filas estiver selecionando o gerenciador de filas de destino, ele fará isso de forma round-robin, usando o algoritmo de gerenciamento de cargas de trabalho; consulte Balanceamento de Carga de Trabalho

MQOO_BIND_ON_OPEN
A opção MQOO_BIND_ON_OPEN na chamada MQOPEN especifica que o gerenciador de filas de destino deve ser fixo. Use a opção MQOO_BIND_ON_OPEN se houver várias instâncias da mesma fila em um cluster. Todas as mensagens colocadas na fila especificando a manipulação de objetos retornada da chamada MQOPEN são direcionadas para o mesmo gerenciador de filas.

- Use a opção MQOO_BIND_ON_OPEN se as mensagens tiverem afinidades. Por exemplo, se um lote de mensagens for ser totalmente processado pelo mesmo gerenciador de filas, especifique MQOO_BIND_ON_OPEN ao abrir a fila. O IBM WebSphere MQ fixa o gerenciador de filas e a rota a ser tomada por todas as mensagens colocadas nessa fila.

- Se a opção MQOO_BIND_ON_OPEN for especificada, a fila deverá ser reaberta para uma nova instância da fila a ser selecionada.

MQOO_BIND_NOT_FIXED
A opção MQOO_BIND_NOT_FIXED na chamada MQOPEN especifica que o gerenciador de filas de destino não é fixo. As mensagens gravadas na fila especificando a manipulação de objetos retornada da chamada MQOPEN são roteadas para um gerenciador de filas no momento de MQPUT mensagem por mensagem. Use a opção MQOO_BIND_NOT_FIXED se não desejar forçar que todas as suas mensagens sejam gravadas no mesmo destino.

- Não especifique MQOO_BIND_NOT_FIXED e MQMF_SEGMENTATION_ALLOWED ao mesmo tempo. Se fizer isso, os segmentos de sua mensagem poderão ser entregues a diferentes gerenciadores de filas, espalhados por todo o cluster.

MQOO_BIND_ON_GROUP
Permite que um aplicativo solicite que um grupo de mensagens seja alocado para a mesma instância de destino. Essa opção é válida somente para filas e afeta somente filas de clusters. Se especificada para uma fila que não seja uma fila de cluster, a opção será ignorada.

- Grupos são roteados para um único destino somente quando MQPMO_LOGICAL_ORDER é especificado na chamada MQPUT. Quando MQOO_BIND_ON_GROUP for especificado, mas uma mensagem não fizer parte de um grupo, o comportamento BIND_NOT_FIXED será usado em seu lugar

MQOO_BIND_AS_Q_DEF
Se você não especificar MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED ou MQOO_BIND_ON_GROUP, a opção padrão é MQOO_BIND_AS_Q_DEF. Usar MQOO_BIND_AS_Q_DEF faz com que a ligação usada para o identificador de filas seja obtida do atributo da fila DefBind.

Relevância de opções MQOPEN

As MQOPEN opções MQOO_BROWSE, MQOO_INPUT_*ou MQOO_SET requerem uma instância local da fila de clusters para que MQOPEN seja bem-sucedido.

As opções de MQOPEN MQOO_OUTPUT, MQOO_BIND_* ou MQOO_INQUIRE não requerem uma instância local da fila de clusters para obter êxito.

Nome do Gerenciador de Filas Resolvido

Quando um nome do gerenciador de filas é resolvido no momento de MQOPEN, o nome resolvido é retornado ao aplicativo. Se o aplicativo tentar usar esse nome em uma chamada MQOPEN subsequente, ele poderá achar que não está autorizado a acessar o nome.

MQPUT, MQPUT1 e clusters

Se MQOO_BIND_NOT_FIXED for especificado em um MQOPEN, as rotinas de gerenciamento de carga de trabalho escolherão qual destino MQPUT ou MQPUT1 selecionar.

Se MQOO_BIND_NOT_FIXED for especificado em uma chamada MQOPEN, cada chamada MQPUT subsequente chamará a rotina de gerenciamento de carga de trabalho para determinar para qual gerenciador de filas enviar a mensagem. O destino e a rota a serem obtidos são selecionados em uma base mensagem por mensagem. O destino e a rota poderão mudar após a mensagem ter sido colocada se as condições na rede mudarem. A chamada MQPUT1 sempre opera como se MQOO_BIND_NOT_FIXED estivesse em vigor, ou seja, sempre chama a rotina de gerenciamento de carga de trabalho.

Quando a rotina de gerenciamento de carga de trabalho tiver selecionado um gerenciador de filas, o gerenciador de filas local concluirá a operação de colocação. A mensagem pode ser colocada em filas diferentes:

1. Se o destino for a instância local da fila, a mensagem será colocada na fila local.
2. Se o destino for um gerenciador de filas em um cluster, a mensagem será colocada em uma fila de transmissão do cluster.

3. Se o destino for um gerenciador de filas fora de um cluster, a mensagem será colocada em uma fila de transmissão com o mesmo nome que o gerenciador de filas de destino.

Se MQOO_BIND_ON_OPEN for especificado na chamada MQOPEN, as chamadas MQPUT não chamarão a rotina de gerenciamento de carga de trabalho porque o destino e a rota já foram selecionados.

MQINQ e clusters

Qual fila de clusters é consultada depende das opções que forem combinadas com MQOO_INQUIRE.

Antes de poder inquirir sobre uma fila, abra-a usando a chamada MQOPEN e especifique MQOO_INQUIRE.

Para inquirir sobre uma fila de clusters, use a chamada MQOPEN e combine outras opções com MQOO_INQUIRE. Os atributos que podem ser inquiridos dependem se há uma instância local da fila de clusters e de como a fila é aberta:

- Combinar MQOO_BROWSE, MQOO_INPUT_* ou MQOO_SET com MQOO_INQUIRE requer uma instância local da fila de clusters para que a abertura seja bem-sucedida. Neste caso, é possível consultar sobre todos os atributos que são válidos para filas locais.
- Combinar MQOO_OUTPUT com MQOO_INQUIRE e não especificar nenhuma das opções anteriores, a instância aberta será:
  - A instância no gerenciador de filas locais, se houver uma. Neste caso, é possível consultar sobre todos os atributos que são válidos para filas locais.
  - Uma instância em outro lugar no cluster, se não houver instância do gerenciador de filas locais. Neste caso apenas os seguintes atributos podem ser consultados. O atributo QType tem o valor MQQT_CLUSTER nesse caso.
    - DefBind
    - DefPersistence
    - DefPriority
    - InhibitPut
    - QDesc
    - QName
    - QType

Para inquirir sobre o atributo DefBind de uma fila de clusters, use a chamada MQINQ com o seletor MQIA_DEF_BIND. O valor retornado é MQBND_BIND_ON_OPEN ou MQBND_BIND_NOT_FIXED ou MQBND_BIND_ON_GROUP. MQBND_BIND_ON_OPEN ou MQBND_BIND_ON_GROUP deve ser especificado ao usar grupos com clusters.

Para inquirir sobre os atributos CLUSTER e CLUSNL da instância local de uma fila, use a chamada MQINQ com o seletor MQCA_CLUSTER_NAME ou o seletor MQCA_CLUSTER_NAMELIST.

Nota: Se uma fila de clusters for aberta sem corrigir a fila ligada ao MQOPEN, sucessivas chamadas MQINQ podem consultar diferentes instâncias da fila de clusters.

Conceitos relacionados

“Opção MQOPEN para fila de cluster” na página 224
A ligação usada para a manipulação de fila é obtida a partir do atributo da fila *DefBind*, que pode obter o valor MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXED ou MQBND_BIND_ON_GROUP.

MQSET e clusters

A opção MQOO_SET da opção MQOPEN requer que haja uma instância local de uma fila de clusters para que MQSET seja bem-sucedido.

Não é possível usar a chamada MQSET para configurar os atributos de uma fila em qualquer outra parte do cluster.

É possível abrir um alias local ou fila remota definida com o atributo do cluster e usar a chamada MQSET. É possível configurar os atributos do alias local ou a fila remota. Não importa se a fila de destino for uma fila de cluster definida em um gerenciador de filas diferente.

Códigos de retorno

Códigos de retorno específicos para clusters

MQRC_CLUSTER_EXIT_ERROR (2266 X'8DA')

Uma chamada MQOPEN, MQPUT ou MQPUT1 é emitida para abrir uma fila de clusters ou para colocar uma mensagem nela. A saída de carga de trabalho do cluster, definida pelo atributo ClusterWorkloadExit de um gerenciador de filas, falha inesperadamente ou não responde a tempo.

Uma mensagem é gravada no log do sistema no WebSphere MQ para z/OS fornecendo mais informações sobre esse erro.

Chamadas subsequentes MQOPEN, MQPUT e MQPUT1 para esse manipulador de filas são processadas como se o atributo ClusterWorkloadExit estivesse em branco.

MQRC_CLUSTER_EXIT_LOAD_ERROR (2267 X'8DB')

No z/OS, a saída de carga de trabalho do cluster não pode ser carregada

Uma mensagem é gravada no registro do sistema e o processamento continuará como se o atributo ClusterWorkloadExit estivesse em branco.

Em plataformas diferentes do z/OS, uma chamada MQCONN ou MQCONNX é emitida para conectar a um gerenciador de filas. A chamada falha porque a saída de carga de trabalho do cluster, definida pelo atributo ClusterWorkloadExit do gerenciador de filas, não pode ser carregada.

MQRC_CLUSTER_PUT_INHIBITED (2268 X'8DC')

Uma chamada MQOPEN com as opções MQOO_OUTPUT e MQOO_BIND_ON_OPEN é efetivamente emitida para uma fila de clusters. Todas as instâncias da fila no cluster têm atualmente o put inibido ao ter o atributo InhibitPut configurado para MQQA_PUT_INHIBITED. Como não há instâncias de fila disponíveis para receber mensagens, a chamada MQOPEN falhará.

Este código de razão ocorre somente quando ambos dos seguintes itens são verdadeiros:

- Não há ocorrência local da fila. Se houver uma ocorrência local, a chamada MQOPEN terá êxito mesmo se a ocorrência local estiver com o put inibido.
- Não há saída de carga de trabalho do cluster para a fila ou há uma saída de carga de trabalho do cluster mas ela não escolhe uma instância de fila. (Se a saída da carga de trabalho do cluster escolher uma instância de fila, a chamada MQOPEN terá êxito, mesmo se a ocorrência estiver com put inibido.)

Se a opção MQOO_BIND_NOT_FIXED for especificada na chamada MQOPEN, a chamada poderá ser bem-sucedida mesmo se todas as filas no cluster estiverem com put inibido. No entanto, uma chamada subsequente MQPUT pode falhar se todas as filas ainda estiverem com put inibido no horário dessa chamada.

MQRC_CLUSTER_RESOLUTION_ERROR (2189 X'88D')

1. Uma chamada MQOPEN, MQPUT ou MQPUT1 é emitida para abrir uma fila de clusters ou para colocar uma mensagem nela. A definição da fila não pode ser resolvida corretamente porque é necessária uma resposta do gerenciador de filas de repositório completo, mas nenhum está disponível.
2. Uma chamada MQOPEN, MQPUT, MQPUT1 ou MQSUB é emitida para um objeto de tópico especificando PUBSCOPE(ALL) ou SUBSCOPE(ALL). A definição de tópico de cluster não pode ser resolvida corretamente porque uma resposta é necessária do gerenciador de filas do repositório completo, mas nenhuma está disponível.

MQRC_CLUSTER_RESOURCE_ERROR (2269 X'8DD')

Uma chamada MQOPEN, MQPUT ou MQPUT1 é emitida para uma fila de clusters. Um erro ocorre durante a tentativa de usar um recurso requerido para clusterização.

MQRC_NO_DESTINATIONS_AVAILABLE (2270 X'8DE')

Uma chamada MQPUT ou MQPUT1 é emitida para colocar uma mensagem em uma fila de clusters. Na hora da chamada, não há mais nenhuma instância da fila no cluster. O MQPUT falha e a mensagem não é enviada.

O erro pode ocorrer se MQOO_BIND_NOT_FIXED for especificado na chamada MQOPEN que abre a fila ou MQPUT1 for usado para colocar a mensagem.

MQRC_STOPPED_BY_CLUSTER_EXIT (2188 X'88C')

Uma chamada MQOPEN, MQPUT ou MQPUT1 é emitida para abrir ou colocar uma mensagem em uma fila de clusters. A saída de carga de trabalho do cluster rejeita a chamada.

Gravando aplicativos clientes

O que você precisa saber para gravar os aplicativos clientes no WebSphere MQ

Os aplicativos podem ser construídos e executados no ambiente do cliente do WebSphere MQ O aplicativo deve ser construído e vinculado ao cliente MQI WebSphere MQ usado. A maneira como os aplicativos são construídos e vinculados varia de acordo com a plataforma e a linguagem de programação usadas. Para obter informações sobre como construir aplicativos clientes, consulte "Construindo aplicativos para clientes MQI do WebSphere MQ" na página 362.

É possível executar um aplicativo WebSphere MQ em um ambiente completo do WebSphere MQ e em um ambiente do cliente MQI do WebSphere MQ sem alterar seu código, desde que determinadas condições sejam atendidas Para obter mais informações sobre como executar seus aplicativos no ambiente do cliente WebSphere MQ , consulte "Executando aplicativos no ambiente do cliente MQI do IBM WebSphere MQ" na página 364.

Se você usar a interface da fila de mensagens (MQI) para gravar aplicativos a serem executados em um ambiente do cliente MQI do WebSphere MQ , haverá alguns controles adicionais a serem aplicados durante uma chamada MQI para assegurar que o processamento do aplicativo WebSphere MQ não seja interrompido. Para obter mais informações sobre esses controles, consulte "Usando a interface da fila de mensagens (MQI) em um aplicativo cliente" na página 357.

Consulte os tópicos a seguir para obter informações sobre como preparar e executar outros tipos de aplicativos, como aplicativos clientes:

- "Preparando e executando aplicativos CICS e Tuxedo" na página 376
- "Preparando e executando aplicativos Microsoft Transaction Server" na página 41
- "Preparando e executando aplicativos JMS do WebSphere MQ" na página 379

Conceitos relacionados

"Conceitos de desenvolvimento de aplicativos" na página 8
É possível usar uma opção de linguagens orientadas a objeto ou processuais para gravar aplicativos IBM WebSphere MQ. Use os links neste tópico para obter informações sobre conceitos do IBM WebSphere MQ que são úteis para desenvolvedores de aplicativos.

"Decidindo qual linguagem de programação usar" na página 80
Use estas informações para descobrir sobre linguagens de programação e estruturas suportadas pelo IBM WebSphere MQe algumas considerações para usá-las.

"Projetando aplicativos IBM WebSphere MQ" na página 91
Quando você tiver decidido como seus aplicativos podem aproveitar as plataformas e ambientes disponíveis para você, será necessário decidir como usar os recursos oferecidos pelo WebSphere MQ.

"Programas de amostra do WebSphere MQ" na página 98

Use esta coleção de tópicos para aprender sobre programas de amostra do WebSphere MQ em diferentes plataformas

"Gravando um Aplicativo de Enfileiramento" na página 197
Use essas informações para aprender como escrever aplicativos de enfileiramento, conectar e desconectar de um gerenciador de filas, publicar/assinar e abrir e fechar objetos.

"Usando serviços da Web no WebSphere MQ" na página 958
É possível desenvolver aplicativos IBM WebSphere MQ para serviços da web usando o transporte IBM WebSphere MQ para SOAP ou a ponte IBM WebSphere MQ para HTTP (Protocolo de Transporte de Hipertexto)

"Escrevendo aplicativos de publicar/assinar" na página 281
Inicie a gravação de aplicativos de publicação / assinatura do WebSphere MQ

"Construindo um aplicativo IBM WebSphere MQ" na página 435
Use estas informações para saber como construir um aplicativo IBM WebSphere MQ em diferentes plataformas.

"Manipulando erros do programa" na página 555
Estas informações explicarão os erros associados às chamadas MQI de aplicativos, ao realizar uma chamada ou quando sua mensagem for entregue para seu destino final.

Usando a interface da fila de mensagens (MQI) em um aplicativo cliente

Esta coleção de tópicos considera as diferenças entre gravar seu aplicativo WebSphere MQ para executar em um ambiente do cliente MQI do WebSphere MQ e executar no ambiente do gerenciador de filas do WebSphere MQ completo.

Ao projetar um aplicativo, considere quais controles você precisa impor durante uma chamada MQI para assegurar que o processamento do aplicativo do WebSphere MQ não seja interrompido

Limitando o tamanho de uma mensagem em um aplicativo cliente

Um gerenciador de filas tem um comprimento máximo de mensagem, mas o tamanho máximo de mensagem que é possível transmitir de um aplicativo cliente é limitado pela definição de canal.

O atributo de comprimento máximo da mensagem (MaxMsgLength) de um gerenciador de filas é o comprimento máximo de uma mensagem que pode ser manipulada por esse gerenciador de filas.

Em plataformas diferentes do z/OS, é possível aumentar o atributo de comprimento máximo da mensagem de um gerenciador de filas. Detalhes são fornecidos em ALTER QMGR.

É possível descobrir o valor de MaxMsgLength para um gerenciador de filas usando a chamada MQINQ.

Se o atributo MaxMsgLength for mudado, não será feita nenhuma verificação para ver se já existem filas, e até mesmo mensagens, com um comprimento maior que o novo valor. Após mudar esse atributo, reinicie aplicativos e canais para assegurar que a mudança entrou em vigor. Então, não será possível que novas mensagens sejam geradas excedendo o MaxMsgLength do gerenciador de filas ou da fila (a menos que a segmentação do gerenciador de filas seja permitida).

O comprimento máximo da mensagem em uma definição de canal limita o tamanho de uma mensagem que é possível transmitir ao longo de uma conexão do cliente. Se um aplicativo WebSphere MQ tentar usar a chamada MQPUT ou a chamada MQGET com uma mensagem maior que essa, um código de erro será retornado para o aplicativo. O parâmetro de tamanho máximo da mensagem da definição de canal não afeta o tamanho máximo da mensagem que pode ser consumido usando MQCB por meio de uma conexão do cliente.

Escolhendo o identificador de conjunto de caracteres codificados (CCSID) pelo cliente ou servidor

Use o CCSID local para o cliente. O gerenciador de filas executa a conversão necessária. Use a variável de ambiente MQCCSID para substituir o CCSID. Se seu aplicativo desempenhar múltiplas operações PUT, o CCSID e campos de codificação do MQMD poderão ser sobrescritos após a conclusão da primeira PUT.

Os dados transmitidos pelo MQI do aplicativo para o stub do cliente devem estar no CCSID local, codificados para o cliente MQI do WebSphere MQ . Se o gerenciador de filas conectado requerer que os dados sejam convertidos, então a conversão será feita pelo código de suporte a clientes no gerenciador de filas.

O cliente Java em V7, no entanto, poderá fazer a conversão se o gerenciador de fila não puder fazer isso Consulte o “ WebSphere MQ classes para conexões do cliente Java” na página 676

O código do cliente assume que os dados de caractere que cruzam o MQI no cliente estão no CCSID configurados para aquela estação de trabalho. Se este CCSID é um CCSID não suportado ou não é o CCSID necessário, ele poderá ser substituído pela variável de ambiente MQCCSID usando um destes comandos:

- No Windows:

```
SET MQCCSID=850
```

- Em sistemas UNIX:

```
export MQCCSID=850
```

Se este parâmetro estiver configurado no perfil, supõe-se que todos os dados de MQI estarão na página de códigos 850.

Nota: A suposição sobre a página de códigos 850 não se aplica aos dados do aplicativo na mensagem.

Se seu aplicativo estiver executando vários PUTs que incluem cabeçalhos WebSphere MQ após o descritor de mensagens (MQMD), esteja ciente de que os campos CCSID e de codificação do MQMD são sobrescritos após a conclusão do primeiro PUT.

Após o primeiro PUT, esses campos contêm o valor usado pelo gerenciador de filas conectado para converter os cabeçalho WebSphere MQ . Assegure que seu aplicativo reconfigure os valores para os valores que ele requer.

Usando MQINQ em um aplicativo cliente

Alguns valores consultados usando MQINQ são modificados pelo código do cliente.

CCSID
é configurado para o CCSID do cliente, não para aquele do gerenciador de filas.

MaxMsgLength
será reduzido se for restringido pela definição de canal. Isto será menor que:

- O valor definido na definição de fila ou
- O valor definido na definição de canal

Para obter informações adicionais, consulte o MQINQ.

Usando a coordenação de ponto de sincronização em um aplicativo cliente

Um aplicativo em execução no cliente base poderá emitir MQCMIT e MQBACK, mas o escopo do controle do ponto de sincronização é limitado aos recursos do MQI. É possível usar um gerenciador de transações externo com um cliente transacional estendido.

No WebSphere MQ, uma das funções do gerenciador de filas é o controle de ponto de sincronização dentro de um aplicativo Se um aplicativo for executado em um cliente base do WebSphere MQ , ele poderá emitir MQCMIT e MQBACK, mas o escopo do controle do ponto de sincronização será limitado aos recursos MQI. O verbo WebSphere MQ MQBEGIN não é válido em um ambiente do cliente base.

Aplicativos em execução no ambiente do gerenciador de filas completo no servidor podem coordenar múltiplos recursos (por exemplo, bancos de dados) por meio de um monitor de transação. No servidor, é possível usar o Monitor de Transação fornecido com produtos WebSphere MQ ou outro monitor de transação como CICS. Não é possível usar um monitor de transação com um aplicativo cliente base.

É possível usar um gerenciador de transações externas com um cliente transacional estendido WebSphere MQ . Consulte O que é um cliente transacional estendido? para obter os detalhes.

Usando leia mais adiante em um aplicativo cliente

É possível usar leia mais adiante em um cliente para permitir que mensagens não persistentes sejam enviadas a um cliente sem que o aplicativo cliente precise solicitar as mensagens.

Quando um cliente requer uma mensagem de um servidor, ele envia uma solicitação ao servidor. Ele envia uma solicitação separada para cada uma das mensagens que consome. Para melhorar o desempenho de um cliente que consome mensagens não-persistentes evitando a necessidade de enviar estas mensagens de pedido, um cliente pode ser configurado para usar leitura antecipada. Leia mais adiante permite que mensagens sejam enviadas a um cliente sem que um aplicativo precise solicitá-las.

Usar leia mais adiante pode melhorar o desempenho ao consumir mensagens não persistentes a partir de um aplicativo cliente. Essa melhoria de desempenho está disponível para os aplicativos MQI e JMS. Aplicativos clientes que usam MQGET ou consumo assíncrono se beneficiam das melhorias de desempenho ao consumir mensagens não persistentes.

Quando você chama MQOPEN com MQOO_READ_AHEAD, o cliente do WebSphere MQ só permite a leitura antecipada se determinadas condições forem atendidas Essas condições incluem:

- Tanto o cliente quanto o gerenciador de filas remotas deve estar em WebSphere MQ Versão 7 ou posterior.
- O aplicativo cliente deve ser compilado e vinculado às bibliotecas do cliente MQI do WebSphere MQ encadeado.
- O canal do cliente deve estar utilizando o protocolo TCP/IP
- O canal deve ter a configuração de SharingConversations diferente de zero (SHARECNV) em ambas as definições de canal, do cliente e do servidor.

Quando a opção leia mais adiante estiver ativada, as mensagens são enviadas para um buffer de memória no cliente chamado de buffer de leia mais adiante. O cliente tem um buffer de leia mais adiante para cada fila que tiver aberta com a opção leia mais adiante ativada. As mensagens no buffer de leia mais adiante não são persistidas. O cliente atualiza periodicamente o servidor com informações sobre a quantia de dados que consumiu.

Nem todos os designs de aplicativo cliente são adequados para usar leia mais adiante, pois nem todas as opções são suportadas para uso. Algumas opções precisam ser consistentes entre chamadas MQGET quando leia mais adiante está ativada. Se um cliente alterar seus critérios de seleção entre chamadas MQGET, as mensagens que estão sendo armazenadas no buffer de leia mais adiante permanecerão conectadas ao buffer de leia mais adiante do cliente. Para obter informações adicionais, consulte “Melhorando o desempenho de mensagens não persistentes” na página 261

A configuração de leitura antecipada é controlada por três atributos, MaximumSize, PurgeTimee UpdatePercentage, que são especificados na sub-rotina MessageBuffer do arquivo de configuração do cliente do WebSphere MQ

Usando put assíncrono em um aplicativo cliente

Usando put assíncrono, um aplicativo pode colocar uma mensagem em uma fila sem esperar uma resposta do gerenciador de filas. É possível usar isso para melhorar o desempenho do sistema de mensagens em algumas situações.

Normalmente, quando um aplicativo coloca uma mensagem ou mensagens em uma fila, usando MQPUT ou MQPUT1, o aplicativo precisa esperar o gerenciador de filas confirmar se ele processou a solicitação MQI. É possível melhorar o desempenho do sistema de mensagens, principalmente para aplicativos que usam ligações do cliente e aplicativos que colocam um grande número de mensagens pequenas em uma fila, optando por colocar as mensagens de forma assíncrona. Quando um aplicativo efetuar put de uma mensagem de forma assíncrona, o gerenciador de filas não retornará o sucesso ou falha de cada chamada, mas será possível verificar erros periodicamente.

Para colocar uma mensagem em uma fila de forma assíncrona, use a opção MQPMO_ASYNC_RESPONSE no campo *Options* da estrutura MQPMO.

Se uma mensagem não for elegível para put assíncrono, ela será colocada em uma fila de forma síncrona.

Ao solicitar resposta de put assíncrono para MQPUT ou MQPUT1, um CompCode e Reason de MQCC_OK e MQRC_NONE não significam necessariamente que a mensagem foi colocada com sucesso em uma fila. Embora o sucesso ou falha de cada chamada MQPUT ou MQPUT1 individual possa não ser retornado imediatamente, o primeiro erro que ocorreu sob uma chamada assíncrona pode ser determinado posteriormente por meio de uma chamada para MQSTAT.

Para obter mais detalhes sobre MQPMO_ASYNC_RESPONSE, consulte Opções de MQPMO.

O programa de amostra Asynchronous Put demonstra alguns dos recursos disponíveis. Para obter detalhes dos recursos e o design do programa, e como executá-lo, consulte “O programa de amostra Asynchronous Put” na página 117.

Usando conversas de compartilhamento em um aplicativo cliente

Em um ambiente no qual conversas de compartilhamento são permitidas, as conversas podem compartilhar uma instância do canal MQI.

Conversas de compartilhamento são controladas por dois campos, ambos chamados SharingConversations, um dos quais faz parte da estrutura de definição de canal (MQCD) e um deles faz parte da estrutura do parâmetro de saída do canal (MQCXP). O campo SharingConversations no MQCD é um valor de número inteiro, que determina o número máximo de conversas que podem compartilhar uma instância do canal associada ao canal. O campo SharingConversations no MQCXP é um valor booleano, que indica se a instância do canal é atualmente compartilhada.

Em um ambiente no qual conversas de compartilhamento não são permitidas, novas conexões do cliente que especificam MQCDs idênticos não compartilharão uma instância do canal.

Uma nova conexão de aplicativo cliente compartilhará a instância do canal quando as condições a seguir forem verdadeiras:

- Ambas as extremidades de conexão do cliente e de conexão do servidor da instância do canal são configuradas para conversas de compartilhamento e esses valores não são substituídos pelas saídas do canal.
- O valor de MQCD da conexão do cliente (fornecido na chamada MQCONNX do cliente ou a partir da tabela de definição de canal de cliente (CCDT)) corresponde exatamente ao valor de MQCD da conexão do cliente fornecido na chamada MQCONNX do cliente ou a partir da CCDT quando a instância do canal existente foi estabelecida pela primeira vez. Observe que o MQCD original pode ter sido alterado subsequentemente pelas saídas ou pela negociação do canal, mas que a correspondência é feita com relação ao valor que foi fornecido ao sistema do cliente antes dessas mudanças serem feitas.
- O limite de conversas de compartilhamento no lado do servidor não é excedido.

Se uma nova conexão de aplicativo cliente corresponder aos critérios para executar o compartilhamento em uma instância do canal com outras conversas, essa decisão será feita antes de qualquer saída ser chamada nessa conversa. Saídas em uma conversa desse tipo não podem alterar o fato de que ela está compartilhando a instância do canal com outras conversas. Se não houver instâncias do canal existentes correspondentes à nova definição de canal, uma nova instância do canal será conectada.

Negociação de canal ocorre somente para a primeira conversa em uma instância do canal; os valores negociados para a instância do canal são fixados nesse estágio e não podem ser alterados quando conversas subsequentes forem iniciadas. Autenticação TLS/SSL também ocorre somente para a primeira conversa.

Se o valor de MQCD de SharingConversations for alterado durante a inicialização de qualquer saída de segurança, envio ou recebimento para a primeira conversa no soquete na extremidade de conexão do cliente ou da conexão do servidor da instância do canal, o novo valor que terá após todas essas saídas serem inicializadas será usado para determinar o valor de conversas de compartilhamento para a instância do canal (o valor mais baixo terá precedência).

Se o valor negociado para conversas de compartilhamento for zero, a instância do canal nunca será compartilhada. Programas de saída adicionais que configuram esse campo para zero são executados de forma similar em sua própria instância do canal.

Se o valor negociado para conversas de compartilhamento for maior que zero, então, SharingConversations de MQCXP será configurado para TRUE para chamadas subsequentes a saídas, indicando que outros programas de saída nesta instância do canal podem ser inseridos simultaneamente a este.

Quando você escrever um programa de saída de canal, considere se ele será executado em uma instância do canal que pode envolver conversas de compartilhamento. Se a instância do canal precisar envolver conversas de compartilhamento, considere o efeito nas outras instâncias da saída do canal de campos de MQCD em mudança; todos os campos de MQCD têm valores comuns entre todas as conversas de compartilhamento. Após a instância do canal ser estabelecida, se programas de saída tentarem alterar campos de MQCD, eles poderão encontrar problemas porque outras instâncias dos programas de saída em execução na instância do canal poderão estar tentando alterar os mesmos campos ao mesmo tempo. Se essa situação puder surgir para seus programas de saída, você deverá serializar o acesso ao MQCD em seu código de saída.

Se estiver trabalhando com um canal definido para compartilhar conversas, mas não desejar que o compartilhamento ocorra em uma instância do canal específica, configure o valor de MQCD de SharingConversations para 1 ou 0 ao inicializar uma saída do canal na primeira conversa na instância do canal. Consulte SharingConversations para obter uma explicação dos valores de SharingConversations.

exemplo

Conversas de compartilhamento estão ativadas.

Você está usando uma definição de canal de conexão do cliente que especifica um programa de saída.

Na primeira vez que esse canal for iniciado, o programa de saída altera alguns dos parâmetros de MQCD quando ele é inicializado. Isso é influenciado pelo canal, portanto, a definição com a qual o canal está em execução agora é diferente daquela que foi originalmente fornecida. O parâmetro MQCXP SharingConversations é configurado para TRUE.

Na próxima vez que o aplicativo se conectar usando este canal, a conversa será executada na instância do canal que foi iniciada anteriormente, porque ela possui a mesma definição de canal original. A instância do canal à qual o aplicativo se conecta na segunda vez é a mesma instância à qual se conectou na primeira vez. Consequentemente, usa as definições que foram alteradas pelo programa de saída. Quando o programa de saída é inicializado para a segunda conversa, embora possa alterar campos de MQCD, eles *não* são influenciados pelo canal. Essas mesmas características se aplicam a qualquer conversa subsequente que compartilhe a instância do canal.

Usando MQCONNX

É possível usar a chamada MQCONNX para especificar uma estrutura de definição de canal (MQCD) na estrutura MQCNO.

Isto permite que o aplicativo cliente que está chamando especifique a definição do canal de conexão do cliente no tempo de execução. Para obter informações adicionais, consulte Usando a estrutura MQCNO em uma chamada MQCONNX. Quando você usa MQCONNX, a chamada emitida no servidor depende do nível do servidor e da configuração do listener.

Quando você usa MQCONNX a partir de um cliente, as seguintes opções são ignoradas:

- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING

A estrutura MQCD que é possível usar depende do número da versão do MQCD que você está usando. Para obter informações sobre versões do MQCD (MQCD_VERSION), consulte Versão do MQCD. É possível usar a estrutura MQCD, por exemplo, para transmitir programas de saída do canal ao servidor. Se você estiver usando o MQCD Versão 3 ou posterior, poderá usar a estrutura para transmitir uma matriz de saídas ao servidor. É possível usar esta função para executar mais de uma operação na mesma

mensagem, como criptografia e compactação, incluindo uma saída para cada operação, em vez de modificar uma saída existente. Se você não especificar uma matriz na estrutura MQCD, os campos de saída únicos serão verificados. Para obter informações adicionais sobre programas de saída do canal, consulte “Programas de Saída de Canal para Canais de Mensagens” na página 402.

Manipulações de conexões compartilhadas no MQCONNX

É possível compartilhar identificadores entre diferentes encadeamentos dentro do mesmo processo, usando identificadores de conexão compartilhados.

Ao especificar uma manipulação de conexões compartilhada, a manipulação de conexões retornada da chamada MQCONNX poderá ser passada nas chamadas MQI subsequentes em qualquer encadeamento no processo.

Nota: É possível usar uma manipulação de conexões compartilhadas em um cliente MQI do WebSphere MQ para se conectar a um gerenciador de filas do servidor que não suporta manipulações de conexões compartilhadas

Para obter mais informações, consulte “Usando MQCONNX” na página 361.

Construindo aplicativos para clientes MQI do WebSphere MQ

Os aplicativos podem ser construídos e executados no ambiente do cliente MQI do WebSphere MQ . O aplicativo deve ser construído e vinculado ao cliente MQI WebSphere MQ usado. A maneira como os aplicativos são construídos e vinculados varia de acordo com a plataforma e a linguagem de programação usadas.

Se um aplicativo precisar ser executado em um ambiente do cliente, será possível gravá-lo nos idiomas mostrados na seguinte tabela:

Tabela 47. Linguagens de Programação Suportadas nos Ambientes do Cliente

plataforma do cliente	**C**	**C++**	**COBOL**	**pTAL**	**RPG**	**Visual Basic**
AIX	Sim	Sim	Sim			
HP Integrity NonStop Server	Sim		Sim	Sim		
HP-UX	Sim	Sim	Sim			
Linux	Sim	Sim	Sim			
Solaris	Sim	Sim	Sim			
Windows	Sim	Sim	Sim			Sim

Consulte os tópicos relacionados para obter instruções sobre como vincular ou construir aplicativos clientes nesses idiomas

Vinculando aplicativos C ao código do cliente MQI do WebSphere MQ

Tendo gravado seu aplicativo WebSphere MQ que você deseja executar no cliente MQI do WebSphere MQ , você deve vinculá-lo a um gerenciador de filas.

É possível vincular seu aplicativo a um gerenciador de filas de duas maneiras:

1. Diretamente, nesse caso, o gerenciador de filas deve estar na mesma estação de trabalho que seu aplicativo
2. Para um arquivo de biblioteca do cliente, que fornece acesso a gerenciadores de fila na mesma estação de trabalho ou em uma estação de trabalho diferente

WebSphere MQ fornece um arquivo de biblioteca do cliente para cada ambiente:

AIX
A biblioteca libmqic.a para aplicativos não encadeados ou a biblioteca libmqic_r.a para aplicativos encadeados.

HP-UX
A biblioteca libmqic.sl para aplicativos não encadeados ou a biblioteca libmqic_r.sl para aplicativos encadeados.

Linux
A biblioteca libmqic.so para aplicativos não encadeados ou a biblioteca libmqic_r.so para aplicativos encadeados.

Solaris
libmqic.so.

Se desejar usar os programas em uma estação de trabalho que tenha apenas o cliente MQI do WebSphere MQ para Solaris instalado, você deverá recompilar os programas para vinculá-los à biblioteca do cliente:

```
$ /opt/SUNWspro/bin/cc -o <prog> <prog> c -mt -lmqic \
-lsocket -lc -lnsl -ldl
```

Os parâmetros devem ser inseridos na ordem correta, conforme mostrado.

Windows
MQIC32.LIB.

Vinculando aplicativos C++ ao código do cliente MQI do WebSphere MQ

É possível escrever aplicativos para serem executados no cliente no C++. Os métodos de construção variam de acordo com o ambiente.

Para obter informações sobre como vincular seus aplicativos C + +, consulte Construindo WebSphere MQ C++.

Para obter detalhes completos de todos os aspectos do uso de C++, consulte Usando C++

Vinculando aplicativos COBOL ao código do cliente MQI do IBM WebSphere MQ

Tendo gravado um aplicativo COBOL que você deseja executar no cliente MQI do IBM WebSphere MQ , deve-se vinculá-lo a uma biblioteca apropriada

O IBM WebSphere MQ fornece um arquivo de biblioteca de cliente para cada ambiente:

AIX
Vincule seu aplicativo COBOL não encadeado com a biblioteca libmqicb.a ou o aplicativo COBOL encadeado com libmqicb_r.a.

HP-UX
Vincule seu aplicativo COBOL não encadeado com a biblioteca libmqicb.sl ou o aplicativo COBOL encadeado com libmqicb_r.sl.

Linux
Vincule seu aplicativo COBOL não encadeado com a biblioteca libmqicb.so ou o aplicativo COBOL encadeado com libmqicb_r.so.

Solaris
Vincule seu aplicativo COBOL não encadeado com a biblioteca libmqicb.so ou o aplicativo COBOL encadeado com libmqicb_r.so.

Windows
Vincule seu código do aplicativo com a biblioteca MQICCBB para COBOL de 32 bits. O IBM WebSphere MQ cliente MQI para Windows não suporta COBOL de 16 bits.

Vinculando aplicativos Visual Basic com o código do cliente MQI do WebSphere MQ

É possível vincular aplicativos Visual Basic ao código do cliente MQI do WebSphere MQ no Windows.

Vincule seu aplicativo Visual Basic aos seguintes arquivos de inclusão:

CMQB.bas
MQI

CMQBB.bas
MQAI

CMQCFB.bas
comandos PCF

CMQXB.bas
Canais

Configure `mqtype=2` para o cliente no compilador Visual Basic para assegurar a seleção automática correta do cliente dll:

MQIC32.dll
Windows 2000, Windows XP e Windows 2003

Executando aplicativos no ambiente do cliente MQI do IBM WebSphere MQ

É possível executar um aplicativo IBM WebSphere MQ em um ambiente completo do IBM WebSphere MQ e em um ambiente do cliente MQI do IBM WebSphere MQ sem mudar seu código, desde que determinadas condições sejam atendidas

Estas condições são que:

- O aplicativo não precisa se conectar a mais de um gerenciador de filas simultaneamente.
- O nome do gerenciador de filas não é prefixado com um asterisco (*) em uma chamada `MQCONN` ou `MQCONNX`.
- O aplicativo não precisa usar nenhuma das exceções listadas em Quais aplicativos são executados em um IBM WebSphere MQ cliente MQI?

Nota: As bibliotecas usadas no momento de edição de link determinam o ambiente no qual seu aplicativo deve ser executado.

Ao trabalhar no ambiente do cliente MQI do IBM WebSphere MQ , lembre-se de que:

- Cada aplicativo em execução no ambiente do cliente MQI IBM WebSphere MQ tem suas próprias conexões com servidores. Um aplicativo estabelece uma conexão com um servidor toda vez que emite uma chamada `MQCONN` ou `MQCONNX`.
- Um aplicativo envia e recebe mensagens de forma síncrona. Isto indica uma espera entre o momento em que a chamada é emitida no cliente e o retorno de um código de conclusão e de um código de razão pela rede.
- Toda conversão de dados é feita pelo servidor, mas consulte também MQCCSID para obter informações sobre como substituir o CCSID configurado da máquina.

Conectando aplicativos clientes MQI do IBM WebSphere MQ a gerenciadores de filas

Um aplicativo em execução em um ambiente do cliente do MQI do IBM WebSphere MQ pode se conectar a um gerenciador de fila de várias maneiras É possível usar variáveis ambientais, a estrutura MQCNO ou uma tabela de definição do cliente.

Quando um aplicativo em execução em um ambiente do cliente IBM WebSphere MQ emite uma chamada MQCONN ou MQCONNX, o cliente identifica como deve se fazer a conexão. Quando uma chamada

MQCONNX é emitida por um aplicativo em um cliente IBM WebSphere MQ, a biblioteca do cliente MQI procura as informações do canal do cliente na ordem a seguir:

1. Usando os conteúdos dos campos *ClientConnOffset* ou *ClientConnPtr* da estrutura MQCNO (se fornecida). Esses campos identificam a estrutura de definição de canal (MQCD) a ser usada como a definição do canal de conexão do cliente. Os detalhes da conexão podem ser substituídos usando uma saída de pré-conexão. Para obter informações adicionais, consulte "Fazendo referência a definições de conexão usando uma saída de pré-conexão de um repositório" na página 429.
2. Se a variável de ambiente MQSERVER for configurada, o canal que ela define será usado.
3. Se um arquivo `mqclient.ini` for definido e contiver um ServerConnectionParms, o canal que ele define será usado. Para obter mais informações, consulte Configurando um cliente usando um arquivo de configuração e Sub-rotina CHANNELS do arquivo de configuração do cliente.
4. Se as variáveis de ambiente MQCHLLIB e MQCHLTAB forem configuradas, a tabela de definição de canal do cliente elas apontam será usada.
5. Se um arquivo `mqclient.ini` for definido e contiver atributos ChannelDefinitionDirectory e ChannelDefinitionFile, esses atributos serão usados para localizar a tabela de definição do canal do cliente. Para obter mais informações, consulte Configurando um cliente usando um arquivo de configuração e Sub-rotina CHANNELS do arquivo de configuração do cliente.
6. Finalmente, se as variáveis de ambiente não estiverem configuradas, o cliente procurará por uma tabela de definição de canal de cliente com um caminho e nome que são estabelecidos no `DefaultPrefix` no arquivo `mqs.ini`. Se a procura por uma tabela de definição de cliente falhar, o cliente usará os caminhos a seguir:
   - UNIX and Linux sistemas: `/var/mqm/AMQCLCHL.TAB`
   - Windows: `C:\Program Files\IBM\Websphere MQ\amqclchl.tab`

A primeira das opções descritas na lista anterior (usando o *ClientConnOffset* ou *ClientConnPtr* os campos de MQCNO) é suportada somente pela chamada MQCONNX. Se o aplicativo estiver usando MQCONN em vez de MQCONNX, as informações de canal serão procuradas das cinco maneiras restantes na ordem mostrada na lista. Se o cliente falhar para localizar as informações de canal, a chamada MQCONN ou MQCONNX falhará.

O nome do canal (para a conexão do cliente) deve corresponder ao nome do canal de conexão do servidor definido no servidor para que a chamada MQCONN ou MQCONNX seja bem-sucedida.

Se você receber um código de retorno MQRC_Q_MGR_NOT_AVAILABLE de seu aplicativo com uma mensagem de erro no arquivo do log de erros de `AMQ9517 - Arquivo danificado`, consulte Migração e tabelas de definição de canal de cliente (CCDT).

Conceitos relacionados

Tabela de Definições de Canal do Cliente

Tarefas relacionadas

Configurando as Conexões entre o Servidor e o Cliente

Referências relacionadas

MQSERVER

MQCHLLIB

MQCHLTAB

Os aplicativos cliente de conexão para gerenciadores de filas usando variáveis de ambiente

informações do canal do cliente podem ser fornecidas para um aplicativo em execução em um ambiente do cliente pelas variáveis de ambiente MQSERVER, MQCHLLIB e MQCHLTAB.

Veja MQSERVER, MQCHLLIB e MQCHLTAB para obter detalhes dessas variáveis.

Conectando aplicativos clientes a gerenciadores de filas usando a estrutura MQCNO

É possível especificar a definição do canal em uma estrutura de definição de canal (MQCD), que é fornecida usando a estrutura MQCNO da chamada MQCONNX.

Para obter mais informações, consulte Usando a estrutura MQCNO em uma chamada MQCONNX .

Conectando aplicativos clientes a gerenciadores de filas usando uma tabela de definição de canal do cliente

Se você usar o comando MQSC DEFINE CHANNEL, os detalhes fornecidos serão colocados na tabela de definição de canal do cliente (ccdt). O conteúdo do parâmetro *QMgrName* da chamada MQCONN ou MQCONNX determina a qual gerenciador de filas o cliente se conecta.

Esse arquivo é acessado pelo cliente para determinar o canal que um aplicativo usará. Quando houver mais de uma definição de canal adequada, a opção de canal será influenciada pelos atributos do canal de peso do canal do cliente (CLNTWGHT) e de afinidade de conexão (AFFINITY).

Função da tabela de definição de canal do cliente

A tabela de definição de canal do cliente (CCDT) contém definições de canais de conexão do cliente. Ela é especialmente útil se seus aplicativos clientes puderem precisar se conectar a diversos gerenciadores de fila alternativos.

A tabela de definição de canal do cliente é criada quando você define um gerenciador de filas.

Nota: O mesmo arquivo pode ser usado por mais de um cliente IBM WebSphere MQ. Você acessa diferentes versões desse arquivo usando as variáveis de ambiente MQCHLLIB e MQCHLTAB do IBM WebSphere MQ. Consulte Usando o WebSphere MQ variáveis de ambiente para obter informações sobre variáveis de ambiente.

Grupos de gerenciadores de filas na CCDT

É possível definir um conjunto de conexões na tabela de definição de canal de cliente (CCDT) como um *grupo de gerenciadores de filas*. É possível conectar um aplicativo a um gerenciador de filas que faz parte de um grupo de gerenciadores de filas. Isso pode ser feito colocando um prefixo no nome do gerenciador de filas em uma chamada `MQCONN` ou `MQCONNX` um asterisco.

Você pode optar por para definir conexões para mais de uma máquina servidor porque:

- Deseja se conectar a um cliente em qualquer um de um conjunto de gerenciadores de filas que está em execução, para melhorar a disponibilidade.
- Deseja reconectar um cliente ao mesmo gerenciador de filas ao qual ele se conectou com sucesso na última vez, mas se conectar a um gerenciador de filas diferente se a conexão falhar.
- Deseja poder tentar novamente uma conexão do cliente com um gerenciador de filas diferente se a conexão falhar, emitindo `MQCONN` no programa cliente novamente.
- Deseja reconectar automaticamente uma conexão do cliente com outro gerenciador de filas se a conexão falhar, sem escrever qualquer código do cliente.
- Deseja reconectar automaticamente uma conexão do cliente a uma instância diferente de um gerenciador de filas multi-instância se uma instância em espera assumir, sem escrever qualquer código do cliente.
- Deseja equilibrar suas conexões do cliente entre vários gerenciadores de filas, com mais clientes se conectando a alguns gerenciadores de filas que outros.
- Deseja difundir a reconexão de muitas conexões do cliente por vários gerenciadores de filas e ao longo do tempo, caso o alto volume de conexões cause uma falha.
- Deseja poder mover seus gerenciadores de filas sem mudar qualquer código do aplicativo cliente.
- Deseja gravar programas ao aplicativo cliente que não precisam conhecer nomes de gerenciadores de filas.

Nem sempre é apropriado se conectar a gerenciadores de filas diferentes. Um cliente transacional estendido ou um cliente Java no WebSphere Application Server, por exemplo, pode precisar se conectar a uma instância previsível do gerenciador de filas. A reconexão do cliente automática não é suportada pelas classes do WebSphere MQ para Java.

Um grupo de gerenciadores de filas é um conjunto de conexões definidas na tabela de definição de canal de cliente (CCDT). O conjunto é definido por seus membros tendo o mesmo valor do atributo **QMNAME** em suas definições de canal.

Figura 70 na página 367 é uma representação gráfica de uma tabela de conexão do cliente, mostrando três grupos de gerenciadores de filas, dois grupos de gerenciadores de filas nomeados gravados na CCDT como **QMNAME**(QM1) e **QMNAME**(QMGrp1), e um grupo em branco ou padrão gravado como **QMNAME**(' ')

1. O grupo de gerenciadores de filas QM1 tem três canais de conexão do cliente, conectando-o aos gerenciadores de filas QM1 e QM2. QM1 pode ser um gerenciador de filas multi-instância localizado em dois servidores diferentes.
2. O grupo de gerenciadores de filas padrão tem seis canais de conexão do cliente que o conectam a todos os gerenciadores de filas.
3. QMGrp1 tem canais de conexão do cliente para dois gerenciadores de filas, QM4 e QM5.

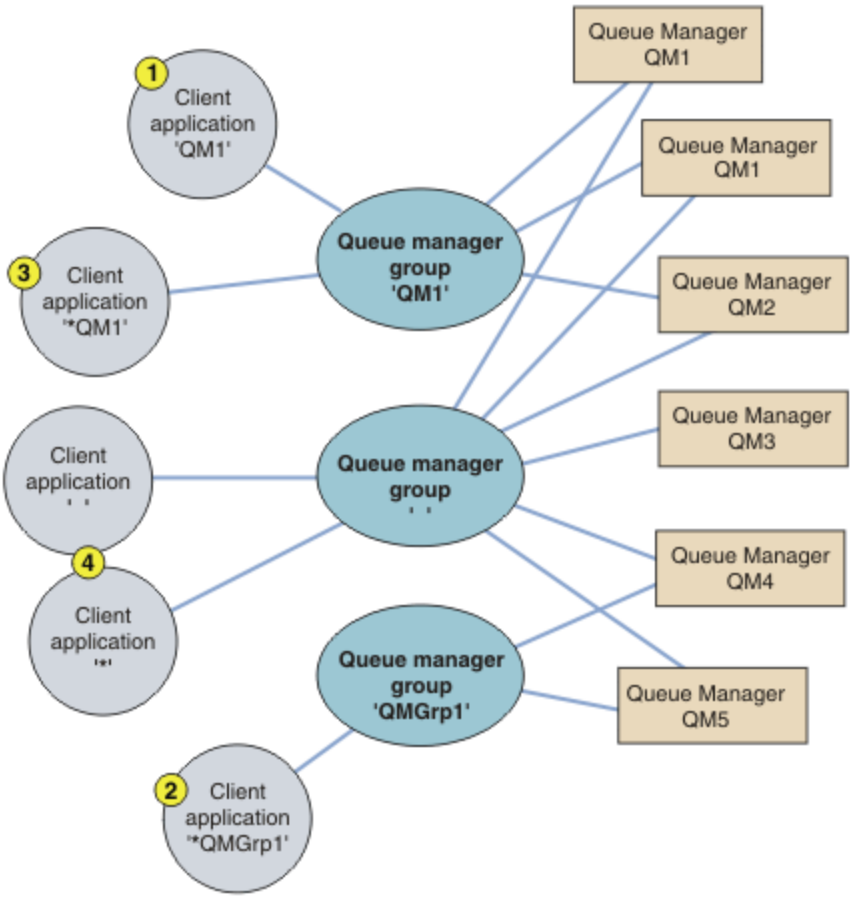


Figura 70. Grupos de gerenciadores de filas

Quatro exemplos de como usar essa tabela de conexões do cliente são descritos com a ajuda dos aplicativos clientes numerados em Figura 70 na página 367.

1. No primeiro exemplo, o aplicativo cliente passa um nome do gerenciador de filas, QM1, como o parâmetro **QmgrName** para sua chamada MQI MQCONN ou MQCONNX. O código do cliente WebSphere MQ seleciona o grupo de gerenciadores de filas correspondente, QM1. O grupo contém três canais de conexão e o cliente MQI do WebSphere MQ tenta se conectar ao QM1 usando cada um desses canais, por sua vez, até localizar um listener do WebSphere MQ para a conexão conectada a um gerenciador de filas em execução chamado QM1.

   A ordem de tentativas de conexão depende do valor do atributo AFFINITY da conexão do cliente e dos pesos de canal do cliente. Dentro dessas restrições, a ordem de tentativas de conexão é aleatória, nas três conexões possíveis e ao longo do tempo, para difundir a carga de criação de conexões.

A chamada MQCONN ou MQCONNX emitida pelo aplicativo cliente é bem-sucedida quando uma conexão é estabelecida com uma instância em execução do QM1.

2. No segundo exemplo, o aplicativo cliente passa um nome do gerenciador de filas com um asterisco como prefixo, *QMGrp1, como o parâmetro **QmgrName** para sua chamada MQI MQCONN ou MQCONNX. O cliente WebSphere MQ seleciona o grupo de gerenciadores de fila correspondente, QMGrp1. Este grupo contém dois canais de conexão do cliente e o cliente MQI do WebSphere MQ tenta se conectar a *qualquer* gerenciador de filas usando cada canal por vez. Neste exemplo, o cliente MQI do WebSphere MQ precisa fazer uma conexão bem-sucedida; o nome do gerenciador de filas ao qual ele se conecta não importa.

   A regra para a ordem de fazer tentativas de conexão é a mesma que antes. A única diferença é que ao colocar como prefixo no nome do gerenciador de filas um asterisco, o cliente indica que o nome do gerenciador de filas não é relevante.

   A chamada MQCONN ou MQCONNX emitida pelo aplicativo cliente é bem-sucedida quando uma conexão é estabelecida com uma instância em execução de qualquer gerenciador de filas conectado pelos canais no grupo de gerenciadores de filas QMGrp1.

3. O terceiro exemplo é essencialmente o mesmo que o segundo porque o parâmetro **QmgrName** tem como prefixo um asterisco, *QM1. O exemplo ilustra que não é possível determinar a qual gerenciador de filas uma conexão de canal do cliente irá se conectar inspecionando o atributo QMNAME em uma definição de canal por si só. O fato de que o atributo **QMNAME** da definição de canal é QM1 não é suficiente para requerer que uma conexão seja feita com um gerenciador de filas chamado QM1. Se seu aplicativo cliente colocar como prefixo de seu parâmetro **QmgrName** um asterisco, então, qualquer gerenciador de filas será um destino de conexão possível.

   Neste caso, as chamadas MQCONN ou MQCONNX emitidas pelo aplicativo cliente são bem-sucedidas quando uma conexão é estabelecida com uma instância em execução de QM1 ou QM2.

4. O quarto exemplo ilustra o uso do grupo padrão. Neste caso, o aplicativo cliente passa um asterisco, '*', ou um espaço em branco ' ', como o parâmetro **QmgrName** para sua chamada MQI MQCONN ou MQCONNX. Por convenção, na definição de canal do cliente, um atributo **QMNAME** significa que o grupo de gerenciadores de filas padrão e um parâmetro **QmgrName** em branco ou com asterisco corresponde a um atributo **QMNAME** em branco.

   Neste exemplo, o grupo de gerenciadores de filas padrão tem conexões de canal do cliente com todos os gerenciadores de filas. Selecionando o grupo de gerenciadores de filas padrão, o aplicativo pode ser conectado a qualquer gerenciador de filas no grupo.

   A chamada MQCONN ou MQCONNX emitida pelo aplicativo cliente é bem-sucedida quando uma conexão é estabelecida com uma instância em execução de qualquer gerenciador de filas.

   Nota: O grupo padrão é diferente de um gerenciador de filas padrão, embora um aplicativo use um parâmetro **QmgrName** em branco para se conectar ao grupo de gerenciadores de filas padrão ou ao gerenciador de filas padrão. O conceito de um grupo de gerenciadores de filas padrão é relevante somente para um aplicativo cliente e um gerenciador de filas padrão para um aplicativo do servidor.

Defina seus canais de conexão do cliente em somente um gerenciador de filas, incluindo os canais que se conectam a um segundo ou terceiro gerenciador de filas. *Não* os defina em dois gerenciadores de filas e, em seguida, tente mesclar as duas tabelas de definição de canal de cliente. Somente uma tabela de definição de canal de cliente pode ser acessada pelo cliente.

Examples

Consulte novamente a lista de razões para usar grupos de gerenciadores de filas no início do tópico. Como usar um grupo de gerenciadores de filas fornece esses recursos?

Conecte-se a qualquer um de um conjunto de gerenciadores de filas.
Defina um grupo de gerenciadores de filas com conexões para todos os gerenciadores de filas no conjunto e conecte-se ao grupo usando o parâmetro **QmgrName** com um asterisco como prefixo.

Reconecte-se ao mesmo gerenciador de filas, mas conecte-se a um diferente se o gerenciador de filas conectado na última vez estiver indisponível.

Defina um grupo de gerenciadores de filas como antes, mas configure o atributo **AFFINITY**(PREFERRED) em cada definição de canal de cliente.

Tente novamente uma conexão com outro gerenciador de filas se uma conexão falhar.

Conecte-se a um grupo de gerenciadores de filas e emita novamente a chamada MQI MQCONN ou MQCONNX se a conexão for interrompida ou se o gerenciador de filas falhar.

Reconecte-se automaticamente a outro gerenciador de filas se uma conexão falhar.

Conecte-se a um grupo de gerenciadores de filas usando a opção MQCONNX **MQCNO** MQCNO_RECONNECT.

Reconecte-se automaticamente a uma instância diferente de um gerenciador de filas multi-instância.

Faça o mesmo que no exemplo anterior. Neste caso, se desejar restringir o grupo de gerenciadores de filas para conectar-se às instâncias de um determinado gerenciador de filas multi-instância, defina o grupo com conexões somente para as instâncias do gerenciador de filas multi-instância.

Também é possível solicitar ao aplicativo cliente para emitir sua chamada MQI MQCONN ou MQCONNX sem asterisco como prefixo no parâmetro **QmgrName**. Dessa maneira, o aplicativo cliente pode se conectar somente ao gerenciador de filas denominado. Por fim, é possível configurar a opção **MQCNO** para MQCNO_RECONNECT_Q_MGR. Essa opção aceita reconexões com o mesmo gerenciador de filas que foi conectado anteriormente. Também é possível usar esse valor para restringir reconexões com a mesma instância de um gerenciador de filas normal.

Balancear conexões do cliente entre gerenciadores de filas, com mais clientes conectados a alguns gerenciadores de filas do que outros.

Defina um grupo de gerenciadores de filas e configure o atributo **CLNTWGHT** em cada definição de canal do cliente para distribuir as conexões desigualmente.

Difunda a carga de reconexão do cliente desigualmente e ao longo do tempo após uma falha de conexão ou do gerenciador de filas.

Faça o mesmo que no exemplo anterior. O cliente MQI do WebSphere MQ randomiza reconexões entre gerenciadores de filas e difunde as reconexões ao longo do tempo.

Mova seu gerenciador de filas sem mudar qualquer código do cliente.

A CCDT isola seu aplicativo cliente do local do gerenciador de filas.

Você tem a opção de distribuir a tabela de conexões do cliente a cada cliente ou colocar a CCDT em um sistema de arquivo compartilhado para cada cliente se referir a ela. Como alternativa, use a versão programática da CCDT suportada na chamada MQI MQCONNX e chame um serviço para passar a CCDT ao aplicativo cliente.

Escreva um aplicativo cliente que não conheça os nomes dos gerenciadores de filas.

Use os nomes do grupo de gerenciadores de filas e estabeleça uma convenção de nomenclatura para os nomes do grupo de gerenciadores de filas que seja relevante para seus aplicativos clientes em sua organização e reflita a arquitetura de suas soluções em vez da nomenclatura dos gerenciadores de filas.

Conectando-se a grupos de filas compartilhadas

É possível conectar seu aplicativo a um gerenciador de filas que faz parte de um grupo de filas compartilhadas. Isto pode ser feito usando o nome do grupo de filas compartilhadas em vez do nome do gerenciador de filas na chamada MQCONN ou MQCONNX.

Grupos de filas compartilhadas possuem um nome de até quatro caracteres. O nome deve ser exclusivo em sua rede e deve ser diferente de qualquer nome de gerenciador de filas.

A definição de canal do cliente deve usar a interface genérica do grupo de filas compartilhadas para conectar-se a um gerenciador de filas disponível no grupo. Para obter mais informações, veja Conectando um cliente a um grupo de filas compartilhadas. Uma verificação é feita para assegurar que o gerenciador de filas o listener se conecta seja um membro do grupo de filas.

Exemplos de peso e afinidade do canal

Esses exemplos ilustram como canais de conexão do cliente são selecionados quando `ClientChannelWeights` diferentes de zero são usados.

Os atributos de canal `ClientChannelWeight` e `ConnectionAffinity` controlam como canais de conexão do cliente são selecionados quando mais de um canal adequado está disponível para uma conexão. Esses canais são configurados para se conectarem a diferentes gerenciadores de filas para fornecer maior disponibilidade, balanceamento de carga de trabalho ou ambos. As chamadas MQCONN que poderiam resultar em uma conexão com um dos vários gerenciadores de filas devem prefixar o nome do gerenciador de filas com um asterisco conforme descrito em:Exemplos de chamadas MQCONN: Exemplo 1. O nome do gerenciador de filas inclui um asterisco (*).

Os canais candidatos aplicáveis para uma conexão são aqueles em que o atributo QMNAME corresponde ao nome do gerenciador de filas especificado na chamada MQCONN. Se todos os canais aplicáveis para uma conexão tiverem um `ClientChannelWeight` de zero (o padrão) então eles são selecionados em ordem alfabética como no exemplo: Exemplos de chamadas MQCONN: Exemplo 1. O nome do gerenciador de filas inclui um asterisco (*).

Os exemplos a seguir ilustram o que acontece quando `ClientChannelWeights` diferentes de zero são usados. Observe que, como esse recurso envolve seleção de canal pseudoaleatória, os exemplos mostram uma sequência de ações que podem ocorrer em vez do que definitivamente ocorrerá.

Exemplo 1. Seleção de canais quando a ConnectionAffinity estiver configurado como PREFERRED
Este exemplo ilustra como um cliente MQI do WebSphere MQ seleciona um canal de uma CCDT, em que ConnectionAffinity é configurado como PREFERRED.

Neste exemplo, diversas máquinas clientes usam uma tabela de definição de canal de cliente (CCDT) fornecida por um gerenciador de filas. A CCDT inclui canais de conexão de cliente com os atributos a seguir (mostrados usando a sintaxe do comando DEFINE CHANNEL):

```
CHANNEL(A) QMNAME(DEV) CONNAME(devqm.it.company.example)
CHANNEL(B) QMNAME(CORE) CONNAME(core1.ops.company.example) CLNTWGHT(5) +
AFFINITY(PREFERRED)
CHANNEL(C) QMNAME(CORE) CONNAME(core2.ops.company.example) CLNTWGHT(3) +
AFFINITY(PREFERRED)
CHANNEL(D) QMNAME(CORE) CONNAME(core3.ops.company.example) CLNTWGHT(2) +
AFFINITY(PREFERRED)
```

O aplicativo emite MQCONN(*CORE)

O canal A não é um candidato para essa conexão, porque o atributo QMNAME não corresponde. Canais B, C e D são identificados como candidatos e são colocados em uma ordem de preferência baseada em seus pesos. Neste exemplo, a ordem pode ser C, B, D. O cliente tenta conectar-se ao gerenciador de filas em core2.ops.company.example. O nome do gerenciador de filas nesse endereço não é verificado, porque a chamada MQCONN incluiu um asterisco no nome do gerenciador de filas.

É importante observar que, com `AFFINITY(PREFERRED)`, toda vez que esta máquina cliente específica se conectar, colocará os canais na mesma ordem preferencial inicial. Isso se aplica mesmo quando as conexões são de processos diferentes ou em momentos diferentes.

Neste exemplo, o gerenciador de filas em core.2.ops.company.example não pode ser atingido. O cliente tenta se conectar a core1.ops.company.example porque o canal B é o próximo na ordem de preferência. Além disso, o canal C será rebaixado para se tornar o menos preferencial.

Uma segunda chamada MQCONN(*CORE) é emitida pelo mesmo aplicativo. O canal C foi rebaixado pela conexão anterior, portanto, o canal mais preferencial agora é B. Esta conexão é feita para core1.ops.company.example.

Uma segunda máquina que compartilha a mesma Tabela de definição de canal de cliente poderá colocar os canais em uma ordem de preferência inicial diferente. Por exemplo, D, B, C. Em circunstâncias normais, com todos os canais funcionando, os aplicativos nesta máquina estão conectados a core3.ops.company.example enquanto aqueles na primeira máquina estão conectados a core2.ops.company.example. Isso permite o balanceamento de carga de trabalho de um grande número

de clientes entre vários gerenciadores de filas enquanto permite que cada cliente individual se conecte ao mesmo gerenciador de filas se ele estiver disponível.

Exemplo 2. Selecionando canais quando o ConnectionAffinity for configurado como NONE
Esse exemplo ilustra como um cliente MQI do WebSphere MQ seleciona um canal de uma CCDT, em que ConnectionAffinity é configurado como NONE.

Neste exemplo, diversos clientes usam uma tabela de definição de canal de cliente (CCDT) fornecida por um gerenciador de filas. A CCDT inclui canais de conexão de cliente com os atributos a seguir (mostrados usando a sintaxe do comando DEFINE CHANNEL):

```
CHANNEL(A) QMNAME(DEV) CONNAME(devqm.it.company.example)
CHANNEL(B) QMNAME(CORE) CONNAME(core1.ops.company.example) CLNTWGHT(5) +
AFFINITY(NONE)
CHANNEL(C) QMNAME(CORE) CONNAME(core2.ops.company.example) CLNTWGHT(3) +
AFFINITY(NONE)
CHANNEL(D) QMNAME(CORE) CONNAME(core3.ops.company.example) CLNTWGHT(2) +
AFFINITY(NONE)
```

O aplicativo emite MQCONN(*CORE). Como no exemplo anterior, o canal A não é considerado porque o QMNAME não corresponde. Canal B, C ou D são selecionados com base em seu peso, com probabilidades de 50%, 30% ou 20%. Neste exemplo, o canal B pode ser selecionado. Não há ordem de preferência persistente criada.

Uma segunda chamada MQCONN(*CORE) é feita. Novamente, um dos três canais aplicáveis é selecionado, com as mesmas probabilidades. Neste exemplo, o canal C é escolhido. No entanto, core2.ops.company.example não responde, portanto, outra opção é feita entre os canais candidatos restantes. O canal B é selecionado e o aplicativo é conectado a core1.ops.company.example.

Com AFFINITY(NONE), cada chamada MQCONN é independente de qualquer outra. Portanto, quando este aplicativo de exemplo faz uma terceira MQCONN(*CORE), ele pode mais uma vez tentar se conectar por meio do canal C interrompido, antes de escolher um dos B ou D.

Exemplos de chamadas MQCONN

Exemplos de uso do MQCONN para conectar-se a um gerenciador de filas específico ou para um de um grupo de gerenciadores de filas.

Em cada um dos exemplos a seguir, a rede é a mesma; há uma conexão definida para dois servidores do mesmo cliente MQI do WebSphere MQ . (Nestes exemplos, a chamada MQCONNX poderia ser usada no lugar da chamada MQCONN.)

Existem dois gerenciadores de filas em execução nas máquinas servidores, um denominado SALE e o outro denominado SALE_BACKUP.

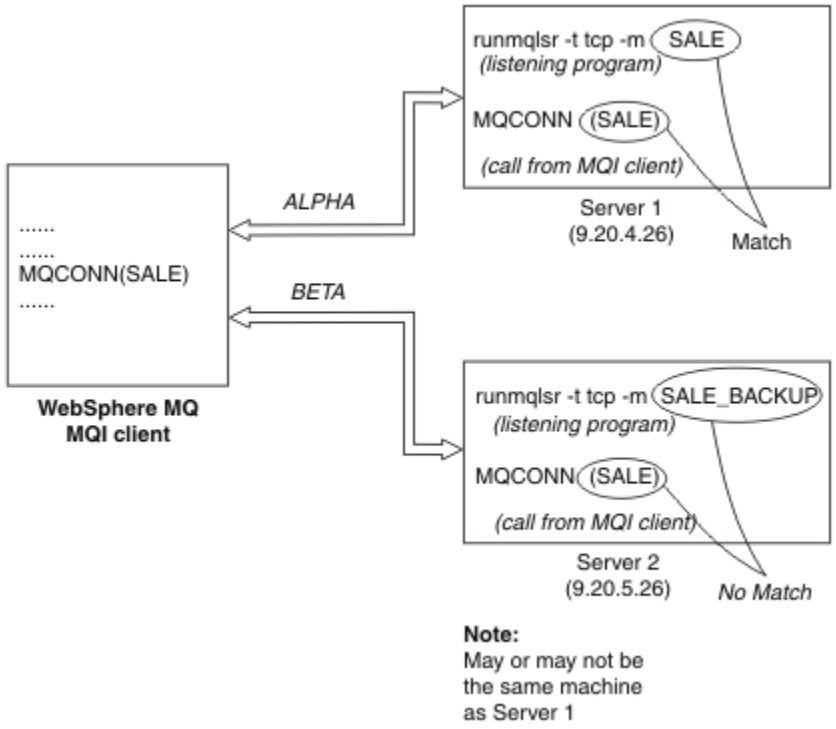


Figura 71. Exemplo de MQCONN

As definições para os canais nestes exemplos são:

Definições de SALE:

```
DEFINE CHANNEL(ALPHA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to WebSphere MQ MQI client')

DEFINE CHANNEL(ALPHA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.4.26) DESCR('WebSphere MQ MQI client connection to server 1') +
QMNAME(SALE)

DEFINE CHANNEL(BETA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.5.26) DESCR('WebSphere MQ MQI client connection to server 2') +
QMNAME(SALE)
```

Definição de SALE_BACKUP:

```
DEFINE CHANNEL(BETA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to WebSphere MQ MQI client')
```

As definições de canal do cliente podem ser resumidas conforme segue:

Nome	CHLTYPE	TRPTYPE	CONNAME	QMNAME
ALPHA	CLNTCONN	TCP	9.20.4.26	SALE
BETA	CLNTCONN	TCP	9.20.5.26	SALE

O que os exemplos de MQCONN demonstram
Os exemplos demonstram o uso de vários gerenciadores de filas como um sistema de backup.

Suponha que o link de comunicação para o Servidor 1 esteja temporariamente interrompido. O uso de vários gerenciadores de filas como um sistema de backup é demonstrado.

Cada exemplo cobre uma chamada MQCONN diferente e fornece uma explicação do que ocorre no exemplo específico apresentado, aplicando as regras a seguir:

1. A tabela de definição de canal de cliente (CCDT) é verificada em ordem alfabética de nome de canal para um nome de gerenciador de filas (campo QMNAME) correspondente àquele fornecido na chamada MQCONN.
2. Se uma correspondência for localizada, a definição de canal será usada.
3. Foi feita uma tentativa de iniciar o canal para na máquina identificada pelo nome de conexão (CONNAME). Se ela for bem-sucedida, o aplicativo continuará. Ele requer:
   - Um listener em execução no servidor.
   - O listener conectado ao mesmo gerenciador de filas que aquele ao qual o cliente deseja se conectar (se especificado).
4. Se a tentativa de iniciar o canal falhar e houver mais de uma entrada na tabela de definição de canal de cliente (neste exemplo, há duas entradas), o arquivo será procurado para uma correspondência adicional. Se uma correspondência for localizada, o processamento continuará na etapa 1.
5. Se nenhuma correspondência for localizada ou não houver mais entradas na tabela de definição de canal de cliente e o canal falhou ao ser iniciado, o aplicativo será incapaz de se conectar. Um código de razão apropriado e um código de conclusão são retornados na chamada MQCONN. O aplicativo pode executar uma ação com base nos códigos de razão e de conclusão retornados.

Exemplo 1. Nome do gerenciador de filas inclui um asterisco (*)
Neste exemplo, o aplicativo não está preocupado a qual gerenciador de filas ele se conecta. O aplicativo emite uma chamada MQCONN para um nome de gerenciador de filas incluindo um asterisco. Um canal adequado é escolhido.

O aplicativo emite:

```
MQCONN (*SALE)
```

Seguindo as regras, isto é o que acontece nessa instância:

1. A tabela de definição de canal de cliente (CCDT) é verificada para o nome de gerenciador de filas SALE, correspondente à chamada MQCONN do aplicativo.
2. Definições de canal para ALPHA e BETA são localizadas.
3. Se um canal tiver um valor de CLNTWGHT igual a 0, esse canal será selecionado. Se ambos tiverem um valor de CLNTWGHT igual a 0, o canal ALPHA será selecionado porque é o primeiro na sequência alfabética. Se ambos os canais tiverem um valor de CLNTWGHT diferente de zero, um canal será selecionado aleatoriamente, com base em seu peso.
4. Uma tentativa de iniciar o canal é feita.
5. Se o canal BETA foi selecionado, a tentativa de iniciá-lo será bem-sucedida.
6. Se o canal ALPHA foi selecionado, a tentativa de iniciá-lo NÃO será bem-sucedida porque o link de comunicação está interrompido. As etapas a seguir se aplicam:
   a. O único outro canal para o nome do gerenciador de filas SALE é BETA.
   b. Uma tentativa de iniciar esse canal é feita - ela é bem-sucedida.
7. Uma verificação para ver se um listener está em execução mostrar que há um em execução. Ele não está conectado ao gerenciador de filas SALE, mas como o parâmetro de chamada MQI tem um asterisco (*) incluído nele, nenhuma verificação é feita. O aplicativo é conectado ao gerenciador de filas SALE_BACKUP e continua o processamento.

Exemplo 2. Nome do gerenciador de filas especificado
Neste exemplo o aplicativo deve se conectar a um gerenciador de filas específico. O aplicativo emite uma chamada MQCONN para esse nome do gerenciador de filas. Um canal adequado é escolhido.

O aplicativo requer uma conexão com um gerenciador de filas específico, denominado SALE, conforme visto na chamada MQI:

```
MQCONN (SALE)
```

Seguindo as regras, isto é o que acontece nessa instância:

1. A tabela de definição de canal de cliente (CCDT) é verificada na sequência alfabética de nome de canal para o nome de gerenciador de filas SALE, correspondente à chamada MQCONN do aplicativo.
2. A primeira definição de canal localizada para correspondência é ALPHA.
3. Uma tentativa de iniciar o canal é feita - ela *não* é bem-sucedida porque o link de comunicação está interrompido.
4. A tabela de definição de canal de cliente é novamente verificada para o nome do gerenciador de filas SALE e o nome do canal BETA é localizado.
5. Uma tentativa de iniciar o canal é feita - ela é bem-sucedida.
6. Uma verificação para ver se um listener está em execução mostra que há um em execução, mas ele não está conectado ao gerenciador de filas SALE.
7. Não há entradas adicionais na tabela de definição de canal de cliente. O aplicativo não pode continuar e recebe o código de retorno MQRC_Q_MGR_NOT_AVAILABLE.

Exemplo 3. Nome do gerenciador de filas está em branco ou tem um asterisco (*)
Neste exemplo, o aplicativo não está preocupado a qual gerenciador de filas ele se conecta. O aplicativo emite uma chamada MQCONN especificando um nome de gerenciador de filas em branco ou um asterisco. Um canal adequado é escolhido.

Ele é tratado da mesma maneira que “Exemplo 1. Nome do gerenciador de filas inclui um asterisco (*)” na página 373.

Nota: Se esse aplicativo estivesse em execução em um ambiente diferente de um cliente MQI do WebSphere MQ e o nome estivesse em branco, ele estaria tentando se conectar ao gerenciador de filas padrão Esse *não* é o caso quando ele é executado a partir de um ambiente do cliente; o gerenciador de filas acessado é aquele associado ao listener ao qual o canal se conecta.

O aplicativo emite:

```
MQCONN ("")
```

ou

```
MQCONN (*)
```

Seguindo as regras, isto é o que acontece nessa instância:

1. A tabela de definição de canal de cliente (CCDT) é verificada na sequência alfabética de nome de canal para um nome de gerenciador de filas que está em branco, correspondente à chamada MQCONN do aplicativo.
2. A entrada para o nome do canal ALPHA tem um nome de gerenciador de filas na definição de SALE. Isso *não* corresponde ao parâmetro da chamada MQCONN, que requer que o nome do gerenciador de filas esteja em branco.
3. A próxima entrada é para o nome do canal BETA.
4. O `queue manager name` na definição é SALE. Mais uma vez, isso *não* corresponde ao parâmetro da chamada MQCONN, que requer que o nome do gerenciador de filas esteja em branco.
5. Não há entradas adicionais na tabela de definição de canal de cliente. O aplicativo não pode continuar e recebe o código de retorno MQRC_Q_MGR_NOT_AVAILABLE.

Acionando no ambiente do cliente

As mensagens enviadas pelos aplicativos WebSphere MQ em execução nos clientes MQI WebSphere MQ contribuem para o acionamento exatamente da mesma maneira que qualquer outra mensagem e podem ser usadas para acionar programas no servidor e no cliente.

Acionamento é explicado em detalhe em Canais de acionamento.

O monitor acionador e o aplicativo a serem iniciados devem estar no mesmo sistema.

As características padrão da fila acionada são as mesmas que as no ambiente do servidor. Em particular, se nenhuma opção de controle de ponto de sincronização MQPMO for especificada em um aplicativo cliente colocando mensagens em uma fila acionada que seja local para um gerenciador de fila z/OS , as mensagens serão colocadas em uma unidade de trabalho. Se a condição de acionamento for então atendida, a mensagem do acionador será colocada na fila de inicialização na mesma unidade de trabalho e não poderá ser recuperada pelo monitor acionador até a unidade de trabalho ser finalizada. O processo que deve ser acionado não é iniciado até que a unidade de trabalho seja finalizada.

Definição de processo

Deve-se definir a definição do processo no servidor, pois isso está associado à fila que tem o acionamento configurado.

O objeto do processo define o que deve ser acionado. Se o cliente e o servidor não estiverem em execução na mesma plataforma, qualquer processo iniciado pelo monitor acionador deverá definir *ApplType*, caso contrário, o servidor usará suas definições padrão (ou seja, o tipo de aplicativo que está normalmente associado à máquina servidor) e causará uma falha.

Por exemplo, se o monitor acionador estiver em execução em um cliente Windows e desejar enviar uma solicitação para um servidor em outro sistema operacional, MQAT_WINDOWS_NT deverá ser definido, caso contrário, o outro sistema operacional usará suas definições padrão e o processo falhará

Monitor acionador

O monitor acionador fornecido por produtos nãoz/OS WebSphere MQ é executado nos ambientes do cliente para sistemas UNIX Linux e Windows .

Para executar o monitor acionador, emita um destes comandos:

- Windows Linux UNIX Nas plataformas Windows, UNIX e Linux :

```
runmqtmc [-m QMgrName] [-q InitQ]
```

A fila de inicialização padrão é SYSTEM.DEFAULT.INITIATION.QUEUE no gerenciador de filas padrão. A fila de inicialização é onde o monitor acionador procura mensagens do acionador. Ela então chama programas para as mensagens do acionador apropriadas. Esse monitor acionador suporta o tipo de aplicativo padrão e é o mesmo que `runmqtrm`, exceto que vincula as bibliotecas do cliente.

A sequência de comandos, construída pelo monitor acionador, é a seguinte:

1. O *ApplicId* da definição de processo relevante. *ApplicId* é o nome do programa a executar, como seria inserido na linha de comandos.
2. A estrutura MQTMC2, entre aspas, obtida a partir da fila de inicialização. Uma sequência de comandos é iniciada contendo essa sequência, exatamente conforme fornecida, entre aspas na ordem que o comando do sistema a aceita como um parâmetro.
3. O *EnvrData* da definição de processo relevante.

O monitor acionador não verifica se há outra mensagem na fila de inicialização até a conclusão do aplicativo que iniciou. Se o aplicativo tiver muito processamento a executar, o monitor acionador pode não acompanhar o número de mensagens do acionador que chegam. Há duas maneiras de lidar com esta situação:

1. Ter mais monitores acionadores em execução

Se optar por ter mais monitores acionadores em execução, será possível controlar o número máximo de aplicativos que podem ser executados a qualquer momento.

2. Executar os aplicativos iniciados em segundo plano

   Se você escolher executar aplicativos em segundo plano, o WebSphere MQ não impõe nenhuma restrição no número de aplicativos que podem ser executados.

Para executar o aplicativo iniciado no segundo plano em sistemas UNIX and Linux , deve-se colocar um & (e comercial) no final do *EnvrData* da definição de processo.

Aplicativos CICS (nãoz/OS)

Um programa aplicativo nãoz/OS CICS que emite uma chamada MQCONN ou MQCONNX deve ser definido para CEDA como RESIDENT. Se você vincular novamente um aplicativo do servidor CICS como um cliente, poderá perder o suporte do ponto de sincronização.

Um programa aplicativo nãoz/OS CICS que emite uma chamada MQCONN ou MQCONNX deve ser definido para CEDA como RESIDENT. Para tornar o código residente o menor possível, é possível vincular-se a um programa separado para emitir a chamada MQCONN ou MQCONNX.

Se a variável de ambiente MQSERVER for usada para definir a conexão do cliente, ela deverá ser especificada no CICSENV CICSENV.CMD do CMD

WebSphere MQ aplicativos podem ser executados em um WebSphere MQ ambiente do servidor ou em um WebSphere MQ cliente sem alterar o código. No entanto, em um ambiente do servidor WebSphere MQ , o CICS pode agir como coordenador do ponto de sincronização e você usa EXEC CICS SYNCPOINT e EXEC CICS SYNCPOINT ROLLBACK em vez de MQCMIT e MQBACK Se um aplicativo CICS for simplesmente vinculado novamente como um cliente, o suporte do ponto de sincronização será perdido MQCMIT e MQBACK devem ser usados para o aplicativo em execução em um cliente MQI WebSphere MQ .

Preparando e executando aplicativos CICS e Tuxedo

Para executar aplicativos CICS e Tuxedo como aplicativos clientes, use bibliotecas diferentes daquelas usadas com aplicativos do servidor. O ID do usuário sob o qual o aplicativo é executado também é diferente.

Para preparar os aplicativos cliente MQI do CICS e Tuxedo para execução como WebSphere MQ , siga as instruções em Configurando um cliente transacional estendido.

Observe, no entanto, que as informações que lidam especificamente com a preparação de aplicativos CICS e Tuxedo, incluindo os programas de amostra fornecidos com WebSphere MQ, assumem que você está preparando aplicativos para execução em um sistema do servidor WebSphere MQ . Como resultado, as informações referem-se apenas às bibliotecas do WebSphere MQ que são destinadas para uso em um sistema do servidor Quando você estiver preparando seus aplicativos clientes, deve-se executar as ações a seguir:

- Use a biblioteca do sistema do cliente apropriada para as ligações de idioma que seu aplicativo usa. Por exemplo, para aplicativos gravados em C no AIX, HP-UXou Solaris, use a biblioteca libmqic em vez de libmqm. Em sistemas Windows , use a biblioteca mqic.lib em vez de mqm.lib
- Em vez das bibliotecas do sistema do servidor mostradas em Tabela 48 na página 376, para AIX, HP-UXe Solaris e Tabela 49 na página 377, para sistemas Windows , use as bibliotecas do sistema do cliente equivalentes.. Se uma biblioteca do sistema do servidor não estiver listada nestas tabelas, use a mesma biblioteca em um sistema do cliente.

Tabela 48. Bibliotecas do sistema cliente no AIX, HP-UXe Solaris

Biblioteca para um sistema do servidor WebSphere MQ	**Biblioteca equivalente a ser usada em um sistema do cliente WebSphere MQ**
libmqmxa	libmqcxa

Tabela 49. Bibliotecas do sistema do cliente em sistemas Windows	
Biblioteca para um sistema do servidor WebSphere MQ	**Biblioteca equivalente a ser usada em um sistema do cliente WebSphere MQ**
mqmxa.lib	mqcxa.lib
mqmtux.lib	mqcxa.lib
mqmenc.lib	mqcxa.lib
mqmcics4.lib	mqccics4.lib

O ID do usuário usado por um aplicativo cliente

Ao executar um aplicativo do servidor WebSphere MQ no CICS, ele normalmente alterna do usuário CICS para o ID do usuário da transação. No entanto, ao executar um aplicativo cliente MQI do WebSphere MQ no CICS, ele mantém a autoridade privilegiada do CICS.

Programas de amostra CICS e Tuxedo

Programas de amostra do CICS e Tuxedo para uso nos sistemas AIX, HP-UX, Solaris e Windows

Tabela 50 na página 377 lista os programas de amostra CICS e Tuxedo que são fornecidos para uso nos sistemas do cliente AIX, HP-UXe Solaris Tabela 51 na página 377 lista as informações equivalentes para sistemas clientes Windows . As tabelas também listam os arquivos que são usados para preparar e executar os programas. Para obter uma descrição dos programas de amostra, consulte “A amostra de transação CICS” na página 120 e “Amostras do TUXEDO” na página 158.

Tabela 50. Programas de Amostra para Sistemas Cliente AIX, HP-UXe Solaris		
Descrição	**Origem**	**Módulo executável**
Programa CICS	amqscic0.ccs	amqscicc
Arquivo de cabeçalho para o programa CICS	amqscih0.h	-
Programa cliente Tuxedo para colocar mensagens	amqstxpx.c	-
Programa cliente Tuxedo para obter mensagens	amqstxgx.c	-
Programa do servidor Tuxedo para os dois programas clientes	amqstxsx.c	-
Arquivo UBBCONFIG para os programas Tuxedo	ubbstxcx.cfg	-
Arquivo de tabela do campo para os programas Tuxedo	amqstxvx.flds	-
Visualizar arquivo de descrição para os programas Tuxedo	amqstxvx.v	-

Tabela 51. Programas de amostra para sistemas clientes Windows		
Descrição	**Origem**	**Módulo executável**
Transação CICS	amqscic0.ccs	amqscicc
Arquivo de cabeçalho para a transação CICS	amqscih0.h	-
Programa cliente Tuxedo para colocar mensagens	amqstxpx.c	-
Programa cliente Tuxedo para obter mensagens	amqstxgx.c	-
Programa do servidor Tuxedo para os dois programas clientes	amqstxsx.c	-
Arquivo UBBCONFIG para os programas Tuxedo	ubbstxcx.cfg	-

Tabela 51. Programas de amostra para sistemas clientes Windows (continuação)		
Descrição	**Origem**	**Módulo executável**
Arquivo de tabela do campo para os programas Tuxedo	amqstxvx.fld	-
Visualizar arquivo de descrição para os programas Tuxedo	amqstxvx.v	-
Makefile para os programas Tuxedo	amqstxmc.mak	-
Arquivo ENVFILE para os programas Tuxedo	amqstxen.env	-

Mensagem de erro AMQ5203, conforme modificado para aplicativos CICS e Tuxedo

Ao executar aplicativos CICS ou Tuxedo que usam um cliente transacional estendido, você pode ver mensagens de diagnóstico padrão. Uma delas foi modificada para uso com um cliente transacional estendido

As mensagens que você pode ver nos arquivos do log de erros do WebSphere MQ são documentadas em Mensagens de diagnóstico: AMQ4000-9999. A mensagem AMQ5203 foi modificada para uso com um cliente transacional estendido. Aqui está o texto da mensagem modificada:

AMQ5203: Ocorreu um erro ao chamar a interface XA.

Explanation

O número de erro é &2, em que um valor de 1 indica que o valor fornecido de sinalizadores de &1 era inválido, 2 indica que houve uma tentativa de usar bibliotecas encadeadas e não encadeadas no mesmo processo, 3 indica que houve um erro com o nome do gerenciador de filas fornecido '&3', 4 indica que o ID do gerenciador de recursos de &1 era inválido, 5 indica que foi feita uma tentativa de usar um segundo gerenciador de filas chamado '&3' quando outro gerenciador de filas já estava conectado, 6 indica que o Gerenciador de Transações foi chamado quando o aplicativo não estava conectado a um gerenciador de filas, 7 indica que a chamada de XA foi feita enquanto outra chamada estava em progresso, 8 indica que a sequência ' xa_info &4' na chamada xa_open continha um valor de parâmetro inválido para o nome do parâmetro '&5' e 9 indica que a sequência ' xa_info &4' na chamada xa_open está sem um parâmetro obrigatório, nome do parâmetro '&5'.

Resposta do usuário

Corrija o erro e tente a operação novamente.

Preparando e executando aplicativos Microsoft Transaction Server

Para preparar um aplicativo MTS para executar como um aplicativo cliente MQI do WebSphere MQ , siga estas instruções conforme apropriado para seu ambiente.

Para obter informações gerais sobre como desenvolver aplicativos do Microsoft Transaction Server (MTS) que acessam os recursos do WebSphere MQ , consulte a seção no MTS na Central de Ajuda do WebSphere MQ

Para preparar um aplicativo MTS para ser executado como um aplicativo cliente MQI do WebSphere MQ , execute um dos seguintes procedimentos para cada componente do aplicativo:

- Se o componente usar as ligações de linguagem C para o MQI, siga as instruções no “Preparando programas C no Windows” na página 466, mas vincule o componente à biblioteca mqicxa.lib em vez da mqic.lib.
- Se o componente usar as classes C++ do WebSphere MQ , siga as instruções em “Construindo programas C++ no Windows” na página 660 , mas vincule o componente à biblioteca imqx23vn.lib em vez de imqc23vn.lib.

- Se o componente usar as ligações de linguagem Visual Basic para o MQI, siga as instruções no “Preparando programas do Visual Basic no Windows” na página 469, mas quando definir o projeto Visual Basic, digite `MqType=3` no campo **Argumentos de compilação condicional**.
- Se o componente usar o WebSphere MQ Classes de Automação para ActiveX (MQAX), defina uma variável de ambiente, GMQ_MQ_LIB, com o valor `mqic32xa.dll` .

  É possível definir a variável de ambiente a partir de seu aplicativo ou ela pode ser definida de modo que seu escopo seja para todo o sistema. Entretanto, defini-la como para todo o sistema pode causar que qualquer aplicativo MQAX existente que não defina a variável de ambiente a partir do aplicativo se comporte incorretamente.

Preparando e executando aplicativos JMS do WebSphere MQ

É possível executar aplicativos JMS do WebSphere MQ no modo cliente, com WebSphere Application Server como seu gerenciador de transações. Você pode ver certas mensagens de aviso.

Para preparar e executar aplicativos JMS do WebSphere MQ no modo cliente, com WebSphere Application Server como seu gerenciador de transações, siga as instruções em “Usando classes do WebSphere MQ para JMS” na página 724.

Ao executar um aplicativo cliente JMS do WebSphere MQ , você pode ver as seguintes mensagens de aviso:

MQJE080
Unidades de licença insuficientes – execute setmqcap

MQJE081
O arquivo que contém as informações da unidade de licença está no formato errado – execute setmqcap

MQJE082
O arquivo que contém as informações da unidade de licença não pode ser localizado – execute setmqcap

Saídas de usuário, saídas de API e WebSphere MQ serviços instaláveis

É possível estender os recursos do gerenciador de filas usando saídas do usuário, saídas da API ou serviços instaláveis. Este tópico contém links para informações sobre como usar e desenvolver esses programas.

Para uma introdução sobre como você pode usar saídas de usuário, saídas de API e serviços instaláveis para estender as instalações do gerenciador de filas, consulte Ampliando as instalações do gerenciador.

Para obter informações sobre gravação e compilação de saídas e serviços instaláveis, consulte “Gravando e compilando saídas e serviços instaláveis” na página 379.

Conceitos relacionados

Programas de Saída de Canal para Canais MQI

Referências relacionadas

Referência de saída de API

Informações de referência da interface de serviços instaláveis

Gravando e compilando saídas e serviços instaláveis

É possível gravar e compilar saídas sem vincular a nenhuma biblioteca IBM WebSphere MQ no UNIX, Linux e Windows.

Sobre esta tarefa

Este tópico aplica-se apenas aos sistemas Windows, UNIX and Linux Para obter detalhes sobre a composição de saídas e serviços instaláveis para outras plataformas, consulte os tópicos específicos da plataforma relevante.

Se o IBM WebSphere MQ for instalado em um local não padrão, deve-se escrever e compilar suas saídas sem vinculação a qualquer biblioteca do IBM WebSphere MQ.

É possível escrever e compilar saídas nos sistemas Windows, UNIX and Linux sem vinculação a qualquer uma destas bibliotecas do IBM WebSphere MQ:

- mqmzf
- mqm
- mqmvx
- mqmvxd
- mqic
- mqutl

Saídas existentes vinculadas a essas bibliotecas continuam funcionando, contanto que nos sistemas UNIX and Linux o IBM WebSphere MQ esteja instalado no local padrão.

Procedimento

1. Inclua o arquivo de cabeçalho cmqec.h.

   Incluir esse arquivo de cabeçalho inclui automaticamente os arquivos de cabeçalho cmqc.h, cmqxc.h e cmqzc.h.
2. Escreva a saída de forma que as chamadas MQI e DCI sejam feitas por meio da estrutura MQIEP. Para obter mais informações sobre a estrutura MQIEP, consulte Estrutura MQIEP.
   - Serviços instaláveis
     - Use o parâmetro **Hconfig** para apontar para a chamada MQZEP.
     - Deve-se verificar se os primeiros 4 bytes de **Hconfig** correspondem ao **StrucId** da estrutura MQIEP antes de usar o parâmetro **Hconfig**.
     - Para obter mais informações sobre como escrever componentes de serviço instaláveis, consulte MQIEP.
   - Saídas de API
     - Use o parâmetro **Hconfig** para apontar para a chamada MQXEP.
     - Deve-se verificar se os primeiros 4 bytes de **Hconfig** correspondem ao **StrucId** da estrutura MQIEP antes de usar o parâmetro **Hconfig**.
     - Para obter mais informações sobre como escrever saídas de API, consulte “Escrevendo saídas de API” na página 394.
   - Saídas do canal
     - Use o parâmetro **pEntryPoints** da estrutura MQCXP para apontar para as chamadas MQI e DCI.
     - Deve-se verificar se o número da versão de MQCXP está na versão 8 ou superior antes de usar **pEntryPoints**.
     - Para obter mais informações sobre como escrever saídas de canal, consulte “Gravando programas de saída do canal” na página 405.
   - Saídas de conversão de dados
     - Use o parâmetro **pEntryPoints** da estrutura MQDXP para apontar para as chamadas MQI e DCI.
     - Deve-se verificar se o número da versão de MQDXP está na versão 2 ou superior antes de usar **pEntryPoints**.

- É possível usar o comando **crtmqcvx** e o arquivo de origem amqsvfc0.c para criar o código de conversão de dados que usa o parâmetro **pEntryPoints**. Consulte "Gravando uma saída de conversão de dados para o WebSphere MQ para Windows" na página 427 e "Gravando uma saída de conversão de dados para sistemas WebSphere MQ em UNIX and Linux" na página 423.
- Se houver saídas de conversão de dados existentes que foram geradas usando o comando **crtmqcvx**, deve-se gerar novamente a saída usando o comando atualizado.
- Para obter mais informações sobre como escrever saídas de conversão de dados, consulte "Escrevendo saídas de conversão de dados" na página 421.

• Saídas de pré-conexão
- Use o parâmetro **pEntryPoints** da estrutura MQNXP para apontar para chamadas MQI e DCI.
- Deve-se verificar se o número da versão MQNXP está na versão 2 ou superior antes de usar **pEntryPoints**.
- Para obter mais informações sobre como escrever saídas de pré-conexão, consulte "Fazendo referência a definições de conexão usando uma saída de pré-conexão de um repositório" na página 429.

• Saídas de publicação
- Use o parâmetro **pEntryPoints** da estrutura MQPSXP para apontar para as chamadas MQI e DCI.
- Deve-se verificar se o número da versão MQPSXP está na versão 2 ou superior antes de usar **pEntryPoints**.
- Para obter mais informações sobre como escrever saídas de publicação, consulte "Escrevendo e compilando saídas de publicação" na página 431.

• Saídas de carga de trabalho do cluster
- Use o parâmetro **pEntryPoints** da estrutura MQWXP para apontar para chamadas MQXCLWLN.
- Deve-se verificar se o número da versão de MQWXP está na versão 4 ou superior antes de usar **pEntryPoints**.
- Para obter mais informações sobre como escrever saídas de carga de trabalho de cluster, consulte "Gravando e Compilando Saídas de Carga de Trabalho do Cluster" na página 433.

Por exemplo, em uma saída do canal que chama MQPUT:

```
pChannelExitParms -> pEntryPoints -> MQPUT_Call(pChannelExitParms -> Hconn,
                                                Hobj,
                                                &md,
                                                &pmo,
                                                messlen,
                                                buffer,
                                                &CompCode,
                                                &Reason);
```

Exemplos adicionais podem ser vistos em "Programas de amostra do WebSphere MQ" na página 98.

3. Compile a saída:
   • Não vincule às bibliotecas do IBM WebSphere MQ .
   • Não inclua um RPath integrado em qualquer biblioteca do IBM WebSphere MQ em sua saída.
   • Para obter mais informações sobre como compilar sua saída, consulte um dos tópicos a seguir:
     - Saídas de API: "Compilando saídas de API" na página 396.
     - Saídas de canal, saídas de publicação, saídas de carga de trabalho do cluster: "Compilando programas de saída do canal no Windows, sistemas UNIX and Linux" na página 420.
     - Saídas de conversão de dados: "Escrevendo saídas de conversão de dados" na página 421.
4. Coloque a saída em um dos locais a seguir:
   • Um caminho de sua escolha que você qualifique totalmente ao configurar a saída

- O caminho de saída padrão, em um diretório de instalação específico. Por exemplo, *MQ_DATA_PATH*/exits/installation2.
- O caminho de saída padrão

  O caminho de saída padrão é *MQ_DATA_PATH*/exits para saídas de 32 bits e *MQ_DATA_PATH*/exits64 para saídas de 64 bits. É possível mudar esses caminhos no arquivo qm.ini ou mqclient.ini. Para obter mais informações, consulte Saída do caminho. No Windows e no Linux, é possível usar o WebSphere MQ Explorer para alterar o caminho:

  a. Clique com o botão direito no nome do gerenciador de filas
  b. Clique em **Propriedades...**
  c. Clique em **Saídas**
  d. No campo do caminho padrão de saídas, especifique o nome do caminho do diretório que retém o programa de saída.

Se uma saída for colocada em um diretório de instalação específico e no diretório de caminho padrão, a saída do diretório de instalação específico será usada pela instalação do WebSphere MQ nomeado no caminho.. Por exemplo, a saída é colocada em /exits/installation2 e em /exits, mas não em /exits/installation1. A WebSphere MQ installation2 usa a saída de /exits/installation2 A WebSphere MQ installation1 usa a saída do diretório /exits .

5. Se necessário, configure a saída:
   - Serviços instaláveis: “Configurando serviços e componentes” na página 390.
   - Saídas de API: “Configurando saídas de API” na página 399.
   - Saídas do canal: “Configurando saídas do canal” na página 421.
   - Saídas de publicação: “Configurando saídas de publicação” na página 432.
   - Saídas de pré-conexão: “Sub-rotina PreConnect do arquivo de configuração do cliente” na página 430.

Serviços e componentes instaláveis para UNIX, Linux e Windows

Esta seção apresenta os serviços instaláveis e as funções e os componentes associados a eles. A interface para essas funções é documentada para que você ou os fornecedores de software possam fornecer os componentes.

As principais razões para fornecer serviços instaláveis do WebSphere MQ são:

- Para fornecer a você a flexibilidade de escolher se deseja usar os componentes fornecidos pelos produtos WebSphere MQ ou substituir ou aumentá-los com outros.
- Para permitir que fornecedores participem, fornecendo componentes que podem usar novas tecnologias, sem fazer mudanças internas nos produtos WebSphere MQ .
- Para permitir que o WebSphere MQ explore novas tecnologias de forma mais rápida e mais barata e, portanto, forneça produtos mais cedo e a preços mais baixos

Serviços Instaláveis e *componentes de serviço* fazem parte da estrutura do produto WebSphere MQ . No centro desta estrutura está a parte do gerenciador de filas que implementa a função e as regras associadas ao Message Queue Interface (MQI). Essa parte central requer inúmeras funções de serviço, denominadas *serviços instaláveis*, para executar seu trabalho. Os serviços instaláveis são:

- Serviço de autorização
- Serviço de Nomes

Cada serviço instalável é um conjunto relacionado de funções implementadas usando um ou mais *componentes de serviços*. Cada componente é chamado usando uma interface publicamente disponível e corretamente arquitetada. Isso permite que fornecedores de software independentes e outros terceiros forneçam componentes instaláveis para aumentar ou substituir aqueles fornecidos pelos produtos do WebSphere MQ Tabela 52 na página 383 resume os serviços e componentes que podem ser usados.

Tabela 52. Resumo dos componentes de serviços instaláveis			
Serviço instalável	**Componente fornecido**	**Função**	**Requisitos**
Serviço de autorização	object authority manager (OAM)	Fornece verificação de autorização sobre os comandos e as chamadas MQI. Os usuários podem gravar seu próprio componente para aumentar ou substituir o OAM. Por exemplo, para verificar se um ID do usuário tem autoridade para abrir uma fila.	(Instalações de autorização de plataforma apropriadas são presumidas)
Serviço de Nomes	Nenhum	Fornece suporte para o gerenciador de filas para consultar o nome do gerenciador de filas que possui uma fila especificada. • Definido pelo usuário **Nota:** As filas compartilhadas devem ter seu atributo *Scope* configurado para CELL.	• Um gerenciador de nomes gravado pelo usuário ou terceiros

A interface de serviços instaláveis é descrita em Informações de referência da interface de serviços instaláveis.

Gravando um componente de serviço

Esta seção descreve o relacionamento entre os serviços, componentes, pontos de entrada e códigos de retorno.

Funções e componentes

Cada serviço consiste em um conjunto de funções relacionadas. Por exemplo, o serviço de nomes contém funções para:

- Consultar um nome de filas e retornar o nome do gerenciador de filas no qual a fila está definida
- Inserir um nome da fila no diretório de serviço
- Excluir um nome da fila do diretório do serviço

Ele também contém as funções de inicialização e finalização.

É fornecido um serviço instalável por um ou mais componentes de serviço. Cada componente pode executar algumas ou todas as funções que são definidas para esse serviço. Por exemplo, no WebSphere MQ para AIX, o componente de serviço de autorização fornecido, o OAM, executa todas as funções disponíveis Consulte “Interface de serviço de autorização” na página 387 para obter informações adicionais. O componente também é responsável por gerenciar quaisquer recursos subjacentes ou software (por exemplo, um diretório LDAP) que precisarem implementar o serviço. Os arquivos de configuração fornecem uma maneira padrão de carregar o componente e de determinar os endereços das rotinas funcionais que fornece.

O Figura 72 na página 384 mostra como os serviços e os componentes estão relacionados:

- Um serviço é definido para um gerenciador de filas por sub-rotinas em um arquivo de configuração.
- Cada serviço é suportado pelo código fornecido no gerenciador de filas. Os usuários não podem mudar esse código e, portanto, não podem criar seus próprios serviços.

- Cada serviço é implementado por um ou mais componentes; eles podem ser fornecidos com o produto ou gravados pelo usuário. Podem ser chamados diversos componentes para um serviço, cada um suportando diferentes recursos no serviço.
- Os pontos de entrada conectam os componentes de serviço ao código de suporte no gerenciador de filas.

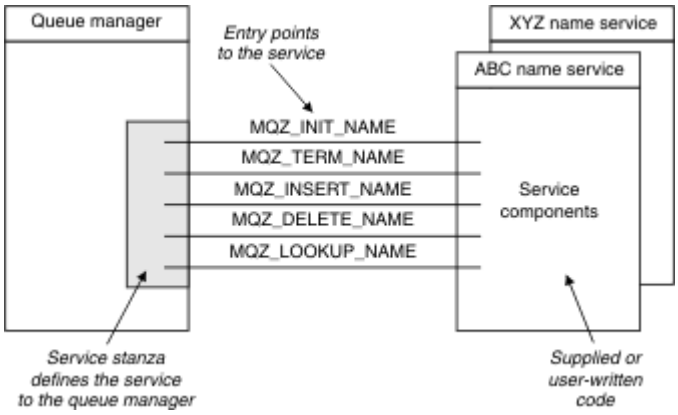


Figura 72. Entendendo serviços, componentes e pontos de entrada

Pontos de entrada

Cada componente de serviço é representado por uma lista de endereços de ponto de entrada das rotinas que suportam um determinado serviço instalável. O serviço instalável define a função a ser executada por cada rotina.

A ordenação dos componentes de serviço quando eles são configurados define a ordem na qual os pontos de entrada são chamados em uma tentativa de atender a uma solicitação para o serviço.

No arquivo de cabeçalho `cmqzc.h` fornecido, os pontos de entrada fornecidos para cada serviço possuem um prefixo MQZID_.

Se os serviços estiverem presentes, os serviços serão carregados em uma ordem predefinida. A lista a seguir mostra os serviços e a ordem na qual são inicializados.

1. `NameService`
2. `AuthorizationService`
3. `UserIdentifierService`

O serviço `AuthorizationService` é o único que está configurado por padrão. Configure o `NameService` e o `UserIdentifierService` manualmente, se desejar usá-los.

Serviços e os componentes de serviço possuem um mapeamento de um-para-um ou um-para-vários. Diversos componentes de serviço podem ser definidos para cada serviço. Nos sistemas UNIX and Linux, o valor de serviço da sub-rotina `ServiceComponent` deve corresponder ao valor do nome da sub-rotina de serviço no arquivo qm.ini. No Windows, o valor da chave de registro de Serviço do `ServiceComponent`deve corresponder ao valor da chave de registro de Nome e é definido como: `HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\`*qmname*`\` em que *qmname* é o nome do gerenciador de filas.

Para sistemas UNIX and Linux, os componentes de serviço são iniciados na ordem em que são definidos no arquivo qm.ini. No Windows, como o registro do Windows é usado, WebSphere MQ emite uma chamada **RegEnumKey** que retorna os valores em ordem alfabética. Portanto, no Windows , os serviços são chamados em ordem alfabética, conforme definidos no registro.

A ordem das definições de `ServiceComponent` é significativa. Essa ordem dita a ordem na qual os componentes são executados para um determinado serviço. Por exemplo, o `AuthorizationService` no Windows é configurado com o componente OAM padrão denominado `MQSeries.WindowsNT.auth.service`.. Componentes adicionais podem ser definidos para este serviço para substituir o OAM padrão. A menos que `MQCACF_SERVICE_COMPONENT` seja especificado,

o primeiro componente encontrado na ordem alfabética será usado para processar a solicitação e o nome para esse componente será usado.

Códigos de retorno

Os componentes de serviço fornecem códigos de retorno para o gerenciador de filas relatar em várias condições. Eles relatam o sucesso ou falha da operação e indicam se o gerenciador de filas continuará para o próximo componente de serviço. Um parâmetro *Continuation* separado transporta essa indicação.

Dados do componente

Um único componente de serviço pode requerer que os dados sejam compartilhados entre as suas várias funções. Serviços instaláveis fornecem uma área de dados opcional a ser passada em cada chamada de um componente de serviço. Essa área de dados é para uso exclusivo do componente de serviço. Ela é compartilhada por todas as chamadas de uma determinada função, mesmo se forem feitas a partir de processos ou espaços de endereço diferentes. É garantido que isso seja endereçável a partir do componente de serviço sempre que for chamado. Deve-se declarar o tamanho dessa área na sub-rotina *ServiceComponent*.

Inicialização e finalização de componentes
O uso das opções de inicialização e finalização de componentes.

Quando a rotina de inicialização do componente é chamada, ela deve chamar a função **MQZEP** do gerenciador de filas para cada ponto de entrada suportado pelo componente. **MQZEP** define um ponto de entrada para o serviço. Todos os pontos de saída indefinidos são presumidos como NULL.

Um componente é sempre chamado uma vez com a opção de inicialização primária, antes de ser chamado de qualquer outra maneira.

Um componente pode ser chamado com a opção de inicialização secundária em determinadas plataformas. Por exemplo, ele pode ser chamado uma vez para cada processo de sistema operacional, encadeamento ou tarefa pelo qual o serviço é acessado.

Se a inicialização secundária for usada:

- O componente pode ser chamado mais de uma vez para a inicialização secundária. Para cada chamada desse tipo, uma chamada correspondente para a finalização secundária é emitida quando o serviço não é mais necessário.

  Para serviços de nomenclatura, essa é a chamada MQZ_TERM_NAME.

  Para serviços de autorização, essa é a chamada MQZ_TERM_AUTHORITY.
- Os pontos de entrada devem ser especificados novamente (chamando MQZEP) toda vez que o componente for chamado para inicialização primária e secundária.
- Somente uma cópia de dados do componente é usada para o componente; não há uma cópia diferente para cada inicialização secundária.
- O componente não é chamado para qualquer outra chamada ao serviço (do processo de sistema operacional, encadeamento ou tarefa, conforme apropriado) antes da realização da inicialização secundária.
- O componente deve configurar o parâmetro *Version* para o mesmo valor para a inicialização primária e secundária.

O componente é sempre chamado com a opção de finalização primária uma vez, quando não for mais necessário. Nenhuma chamada adicional será feita para esse componente.

O componente é chamado com a opção de finalização secundária se tiver sido chamado para inicialização secundária.

Gerenciador de autoridade de objeto (OAM)
O componente de serviço de autorização fornecido com os produtos WebSphere MQ é chamado de Object Authority Manager (OAM).

Por padrão, o OAM está ativo e trabalha com os comandos de controle **dspmqaut** (autoridade de exibição),**dmpmqaut** (autoridade de dump) e **setmqaut** (autoridade de configuração ou reconfiguração).

A sintaxe desses comandos e como usá-los estão descritos em Os comandos de controle.

O OAM trabalha com a *entidade* de um diretor ou grupo.

- Nos sistemas UNIX and Linux:
  - o principal é um ID do usuário ou um ID associado a um programa aplicativo em execução em nome de um usuário.
  - O grupo é uma coleção de proprietários definida pelo sistema UNIX ou Linux
  - Autorizações podem ser concedidas ou revogadas apenas no nível do grupo. Um pedido para conceder ou revogar a autoridade de um usuário atualiza o grupo primário para esse usuário.
- Em sistemas Windows:
  - o principal é um ID do usuário do Windows ou um ID associado a um programa de aplicativo em execução em nome de um usuário.
  - O grupo é um grupo do Windows
  - Autorizações podem ser concedidas ou revogadas no nível do diretor ou do grupo.

Quando uma solicitação MQI é feita ou um comando é emitido, o OAM verifica a autorização da entidade associada à operação para ver se ela pode:

- Executar a operação solicitada.
- Acessar os recursos do gerenciador de filas especificado.

O serviço de autorização permite aumentar ou substituir a verificação de autoridade fornecida para gerenciadores de filas, gravando seu próprio componente de serviço de autorização.

Serviço de Nomes
O serviço de nomes é um serviço instalável que fornece suporte para o gerenciador de filas consultar o nome do gerenciador de filas que possui uma fila especificada. Nenhum outro atributo de fila pode ser recuperado de um serviço de nomes.

O serviço de nomes permite que um aplicativo abra filas remotas para saída como se fossem filas locais. Um serviço de nomes não é chamado para objetos diferentes de filas.

Nota: As filas remotas ***devem*** ter seu atributo *Scope* configurado para CELL.

Quando um aplicativo abre uma fila, ele procura o nome da primeira fila no diretório do gerenciador de filas. Se não a localizar lá, ele procura em todos os serviços de nomes que foram configurados, até que localize um que reconheça o nome da fila. Se nenhum reconhecer o nome, a abertura falhará.

O serviço de nomes retorna o gerenciador de filas proprietário dessa fila. O gerenciador de filas continua, então, com a solicitação MQOPEN como se o comando tivesse especificado a fila e nome do gerenciador de filas na solicitação original.

A interface de serviço de nomes (NSI) faz parte da estrutura WebSphere MQ .

Como o serviço de nomes funciona

Se uma definição de fila especificar o atributo *Scope* como gerenciador de filas, ou seja, SCOPE(QMGR) no MQSC, a definição de fila (juntamente com todos os atributos da fila) será armazenada somente no diretório do gerenciador de filas. Isso não pode ser substituído por um serviço instalável.

Se uma definição de fila especificar o atributo *Scope* como célula, ou seja, SCOPE(CELL) no MQSC, a definição de fila será novamente armazenada no diretório do gerenciador de filas, juntamente com todos os atributos da fila. No entanto, a fila e o nome do gerenciador de filas também são armazenados em um serviço de nomes. Se não houver nenhum serviço disponível que possa armazenar essas informações, uma fila com o *Scope* célula não poderá ser definida.

O diretório no qual as informações são armazenadas pode ser gerenciado pelo serviço ou o serviço pode usar um serviço subjacente, por exemplo, um diretório LDAP, para esse propósito. Em ambos os casos, as definições armazenadas no diretório devem persistir, mesmo após o componente e o gerenciador de filas serem finalizados, até serem excluídos explicitamente.

Nota:

1. Para enviar uma mensagem a uma definição de fila local de um host remoto (com um escopo de CELL) em um gerenciador de filas diferente em uma célula de diretório de nomenclatura, será necessário definir um canal.
2. Não é possível obter mensagens diretamente da fila remota, mesmo quando ela tem um escopo de CELL.
3. Nenhuma definição de fila remota é necessária ao enviar para uma fila com um escopo de CELL.
4. O serviço de nomenclatura define centralmente a fila de destino, embora você ainda precise de uma fila de transmissão para o gerenciador de filas de destino e um par de definições de canal. Além disso, a fila de transmissão no sistema local deve ter o mesmo nome que o gerenciador de filas proprietário da fila de destino, com o escopo de célula, no sistema remoto.

   Por exemplo, se o gerenciador de filas remotas tiver o nome QM01, a fila de transmissão no sistema local também deve ter o nome QM01.

Interface de serviço de autorização
O serviço de autorização fornece pontos de entrada para uso pelo gerenciador de filas.

Os pontos de entrada são os seguintes:

MQZ_AUTHENTICATE_USER
Autentica um ID do usuário e senha e pode configurar campos de contexto de identidade.

MQZ_CHECK_AUTHORITY
Verifica se uma entidade possui autoridade para desempenhar uma ou mais operações em um objeto especificado.

MQZ_CHECK_PRIVILEGED
Verifica se um usuário especificado é um usuário privilegiado.

MQZ_COPY_ALL_AUTHORITY
Copia todas as autorizações atuais que existem para um objeto referenciado para outro objeto.

MQZ_DELETE_AUTHORITY
Exclui todas as autorizações associadas a um objeto especificado.

MQZ_ENUMERATE_AUTHORITY_DATA
Recupera todos os dados de autoridade que correspondem aos critérios de seleção especificados.

MQZ_FREE_USER
Libera recursos alocados associados.

MQZ_GET_AUTHORITY
Obtém a autoridade que uma entidade tem para acessar um objeto especificado.

MQZ_GET_EXPLICIT_AUTHORITY
Obtém a autoridade que um grupo denominado possui para acessar um objeto especificado (mas sem a autoridade adicional do grupo **nobody**) ou a autoridade que o grupo principal do proprietário nomeado possui para acessar um objeto especificado.

MQZ_INIT_AUTHORITY
Inicializa componente de serviço de autorização.

MQZ_INQUIRE
Consulta a funcionalidade suportada do serviço de autorização.

MQZ_REFRESH_CACHE
Atualizar todas as autorizações.

MQZ_SET_AUTHORITY
Define a autoridade que uma entidade tem para um objeto especificado.

MQZ_TERM_AUTHORITY
Finaliza o componente de serviço de autorização.

Além disso, no WebSphere MQ para Windows, o serviço de autorização fornece os seguintes pontos de entrada para uso pelo gerenciador de filas:

- **MQZ_CHECK_AUTHORITY_2**
- **MQZ_GET_AUTHORITY_2**
- **MQZ_GET_EXPLICIT_AUTHORITY_2**
- **MQZ_SET_AUTHORITY_2**

Esses pontos de entrada suportam o uso do Windows Security Identifier (NT SID).

Esses nomes são definidos como **typedef**s, no arquivo de cabeçalho `cmqzc.h`, que pode ser usado para prototipar as funções do componente.

A função de inicialização (**MQZ_INIT_AUTHORITY**) deve ser o ponto de entrada principal do componente. As outras funções são chamadas por meio do endereço do ponto de entrada que a função de inicialização incluiu no vetor do ponto de entrada do componente.

Interface de serviço de nomes
Um nome do serviço fornece pontos de entrada para uso pelo gerenciador de filas.

Os pontos de entrada a seguir são fornecidos:

MQZ_INIT_NAME
Inicializar o nome do componente de serviço.

MQZ_TERM_NAME
Finalizar o nome do componente de serviço.

MQZ_LOOKUP_NAME
Consultar o nome do gerenciador de filas para a fila especificada.

MQZ_INSERT_NAME
Inserir uma entrada que contém o nome do gerenciador de filas proprietário para a fila especificada no diretório usado pelo serviço.

MQZ_DELETE_NAME
Excluir a entrada para a fila especificada do diretório usado pelo serviço.

Se houver mais de um serviço de nomes configurado:

- Para consulta, a função MQZ_LOOKUP_NAME é chamada para cada serviço na lista até que o nome da fila seja resolvido (a menos que algum componente indique que a procura deve parar).
- Para inserir, a função MQZ_INSERT_NAME é chamada para o primeiro serviço na lista que suporta essa função.
- Para excluir, a função MQZ_DELETE_NAME é chamada para o primeiro serviço na lista que suporta essa função.

Não tenha mais de um componente que suporte as funções inserir e excluir. Entretanto, um componente que suporta somente consulta é viável e pode ser usado, por exemplo, como o último componente na lista para resolver qualquer nome que não seja conhecido por nenhum outro componente de serviço de nomes para um gerenciador de filas no qual o nome possa ser definido.

Na linguagem de programação C, os nomes são definidos como tipos de dados de função usando a instrução typedef. Eles podem ser usados para criar protótipo das funções de serviço, para assegurar que os parâmetros estejam corretos.

O arquivo de cabeçalho que contém todo o material específico para serviços instaláveis é `cmqzc.h` para a linguagem C.

Além da função de inicialização (MQZ_INIT_NAME), que deve ser o ponto de entrada principal do componente, funções são chamadas pelo endereço do ponto de entrada que a função de inicialização incluiu, usando a chamada MQZEP.

Usando diversos componentes de serviço
É possível instalar mais de um componente para um serviço. Isso permite que os componentes forneçam somente implementações parciais do serviço e dependam de outros componentes para fornecer as funções restantes.

Exemplo de uso de vários componentes

Suponha que você crie dois um componentes de serviços de nomes chamados `ABC_name_serv` e `XYZ_name_serv`.

`ABC_name_serv`
Esse componente suporta a inserção ou a exclusão de um nome do diretório de serviço, mas não suporta a procura de um nome de fila.

`XYZ_name_serv`
Esse componente suporta a procura de um nome de fila, mas não suporta a inserção ou a exclusão de um nome do diretório de serviço.

O componente `ABC_name_serv` retém um banco de dados de nomes de filas e usa dois algoritmos simples para inserir ou excluir um nome do diretório de serviço.

O componente `XYZ_name_serv` usa um algoritmo simples que retorna um nome do gerenciador de filas fixo para qualquer nome de fila com o qual ele é chamado. Ele não mantém um banco de dados de nomes de filas e, portanto, não suporta as funções de inserção e exclusão.

Os componentes são instalados no mesmo gerenciador de filas. As sub-rotinas *`ServiceComponent`* são ordenadas para que o componente `ABC_name_serv` seja chamado primeiro. Quaisquer chamadas para inserir ou excluir uma fila em um diretório de componentes são manipuladas pelo componente `ABC_name_serv`; é o único que implementa essas funções. No entanto, uma chamada de consulta que o componente `ABC_name_serv` não pode resolver é transmitida para o componente somente de consulta, `XYZ_name_serv`. Esse componente fornece um nome do gerenciador de filas a partir de seu algoritmo simples.

Omitir pontos de entrada ao usar vários componentes

Se decidir usar vários componentes para fornecer um serviço, será possível projetar um componente de serviço que não implementa determinadas funções. A estrutura de serviços instaláveis não coloca restrições sobre o que é possível omitir. No entanto, para serviços instaláveis específicos, omissão de um ou mais funções pode ser logicamente inconsistente com o propósito do serviço.

Exemplo de pontos de entrada usados com vários componentes

Tabela 53 na página 389 mostra um exemplo do serviço de nomes instalável para o qual os dois componentes foram instalados. Cada um suporta um conjunto diferente de funções associadas a esse determinado serviço instalável. Para a função de inserção, o ponto de entrada do componente ABC é chamado primeiro. Os pontos de entrada que não foram definidos para o serviço (usando **MQZEP**) são considerados NULL. Um ponto de entrada para inicialização é fornecido na tabela, mas isso não é necessário porque a inicialização é executada pelo ponto de entrada principal do componente.

Quando o gerenciador de filas precisar usar um serviço instalável, ele usa os pontos de entrada definidos para esse serviço (as colunas em Tabela 53 na página 389). Tomando cada componente em sua vez, o gerenciador de filas determina o endereço da rotina que implementa a função necessária. Em seguida, chama a rotina, se ela existir. Se a operação for bem-sucedida, quaisquer resultados e informações de status serão usados pelo gerenciador de filas.

Tabela 53. Exemplo de pontos de entrada para um serviço instalável

Número da função	**Componente de serviço de nomes ABC**	**Componente de serviço de nomes XYZ**
MQZID_INIT_NAME (Initialize)	ABC_initialize()	XYZ_initialize()

Tabela 53. Exemplo de pontos de entrada para um serviço instalável (continuação)		
Número da função	**Componente de serviço de nomes ABC**	**Componente de serviço de nomes XYZ**
MQZID_TERM_NAME (Terminate)	ABC_terminate()	XYZ_terminate()
MQZID_INSERT_NAME (Insert)	ABC_Insert()	NULL
MQZID_DELETE_NAME (Delete)	ABC_Delete()	NULL
MQZID_LOOKUP_NAME (Lookup)	NULL	XYZ_Lookup()

Se a rotina não existir, o gerenciador de filas repete esse processo para o próximo componente na lista. Além disso, se a rotina existir, mas retornar um código indicando que ela não pôde executar a operação, a tentativa continuará com o próximo componente disponível. Rotinas em componentes de serviço podem retornar um código que indica que não deve ser feita nenhuma tentativa adicional para executar a operação.

Configurando serviços e componentes

Configure os componentes de serviço usando os arquivos de configuração do gerenciador de filas, exceto nos sistemas Windows , em que cada gerenciador de filas tem sua própria sub-rotina no Registro

1. Inclua sub-rotinas no arquivo de configuração do gerenciador de filas para definir o serviço para o gerenciador de filas e para especificar o local do módulo.

   Cada serviço usado deve ter uma sub-rotina *Service*, que define o serviço para o gerenciador de filas.

   Para cada componente dentro de um serviço, deve haver uma sub-rotina *ServiceComponent*. Isso identifica o nome e o caminho do módulo que contém o código para esse componente.

   Para obter mais informações, consulte "Formato da sub-rotina Service" na página 390 e "Formato da sub-rotina do componente de serviço" na página 391

   O componente do serviço de autorização, conhecido como Gerenciador de autoridade de objeto (OAM), é fornecido com o produto. Ao criar um gerenciador de filas, o arquivo de configuração do gerenciador de filas (ou o Registro em sistemas Windows) é atualizado automaticamente para incluir as sub-rotinas apropriadas para o serviço de autorização e para o componente padrão (o OAM). Para outros componentes, deve-se configurar o arquivo de configuração do gerenciador de filas manualmente.

   O código para cada componente de serviço é carregado no gerenciador de filas quando gerenciador de filas é iniciado, usando a ligação dinâmica, quando isso for suportado na plataforma.
2. Pare e reinicie o gerenciador de filas para ativar o componente.

Formato da sub-rotina Service

A sub-rotina Service contém o nome do serviço e o número de pontos de entrada definidos para o serviço.

O formato da sub-rotina é o seguinte:

```
Service:
  Name=<service_name>
  EntryPoints=<entries>
```

em que:

<service_name>
O nome do serviço. Isso é definido pelo serviço.

<entries>
O número de pontos de entrada definidos para o serviço. Inclui os pontos de entrada de inicialização e finalização.

Formato de sub-rotina de serviço de sistemas Windows
Em sistemas Windows , a sub-rotina *Service* inclui um atributo SecurityPolicy .

O formato da sub-rotina é:

```
Service:
  Name=<service_name>
  EntryPoints=<entries>
  SecurityPolicy=<policy>
```

em que:

<service_name>
O nome do serviço. Isso é definido pelo serviço.

<entries>
O número de pontos de entrada definidos para o serviço. Inclui os pontos de entrada de inicialização e finalização.

<policy>
NTSIDsRequired (o Identificador de Segurança do Windows) ou Default. Se você não especificar NTSIDsRequired, o valor Default será usado. Esse atributo é válido apenas se Name tiver um valor de AuthorizationService.

Consulte também “Configurando sub-rotinas de serviço de autorização: sistemas Windows” na página 392.

Formato da sub-rotina do componente de serviço

O formato da sub-rotina do componente de serviço é:

```
ServiceComponent:
  Service=<service_name>
  Name=<component_name>
  Module=<module_name>
  ComponentDataSize=<size>
```

em que:

<service_name>
O nome do serviço. Isso deve corresponder ao Name especificado em uma sub-rotina de serviço.

<component_name>
Um nome descritivo do componente de serviço. Ele deve ser exclusivo e conter apenas os caracteres válidos para os nomes de objetos do WebSphere MQ (por exemplo, nomes de filas). Este nome ocorre em mensagens do operador geradas pelo serviço. Recomendamos que seja usado um nome que comece com uma marca comercial da empresa ou sequência distintiva semelhante.

<module_name>
O nome do módulo a conter o código para esse componente.

<size>
O tamanho em bytes da área de dados do componente passado ao componente em cada chamada. Especifique zero se nenhum dado de componente for requerido.

Essas duas sub-rotinas podem ocorrer em qualquer ordem e as chaves de sub-rotina sob elas também podem ocorrer em qualquer ordem. Para qualquer uma dessas sub-rotinas, todas as chaves de sub-rotina devem estar presentes. Se uma chave de sub-rotina for duplicada, a última será usada.

No momento da inicialização, o gerenciador de filas processa cada entrada do componente de serviço no arquivo de configuração na sua vez. Em seguida, ele carrega o módulo componente do especificado, chamando o ponto de entrada do componente (que deve ser o ponto de entrada para a inicialização do componente), passando a ele um identificador de configuração.

Configurando as sub-rotinas do serviço de autorização: sistemas UNIX and Linux
Nos sistemas UNIX and Linux, cada gerenciador de filas tem seu próprio arquivo de configuração de gerenciador de filas.

Por exemplo, o caminho padrão e o nome do arquivo do arquivo de configuração do gerenciador de filas para o gerenciador de filas QMNAME é `/var/mqm/qmgrs/QMNAME/qm.ini`.

As sub-rotinas *Service* e *ServiceComponent* para o componente de autorização padrão são incluídas em `qm.ini` automaticamente, mas podem ser substituídas por `mqsnoaut`. Qualquer outra sub-rotina *ServiceComponent* deve ser incluída manualmente.

Por exemplo, as seguintes sub-rotinas no arquivo de configuração do gerenciador de filas definem dois componentes de serviço de autorização no WebSphere MQ para AIX. O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado.

```
Service:
   Name=AuthorizationService
   EntryPoints=13

ServiceComponent:
   Service=AuthorizationService
   Name=MQSeries.UNIX.auth.service
   Module=MQ_INSTALLATION_PATH/lib/amqzfu
   ComponentDataSize=0

ServiceComponent:
   Service=AuthorizationService
   Name=user.defined.authorization.service
   Module=/usr/bin/udas01
   ComponentDataSize=96
```

Figura 73. sub-rotinas de serviço de autorização UNIX and Linux em qm.ini

A sub-rotina do componente de serviço (`MQSeries.UNIX.auth.service`) define o componente de serviço de autorização padrão, o OAM. Se essa sub-rotina for removida e o gerenciador de filas reiniciado, o OAM será desativado e nenhuma verificação de autorização será realizada.

Configurando sub-rotinas de serviço de autorização: sistemas Windows
No WebSphere MQ para Windows , cada gerenciador de filas tem sua própria sub-rotina no registro

A sub-rotina *Service* e a sub-rotina *ServiceComponent* para o componente de autorização padrão são incluídas no registro automaticamente, mas podem ser substituídas usando `mqsnoaut`. Qualquer outra sub-rotina *ServiceComponent* deve ser incluída manualmente.

Também é possível incluir o atributo `SecurityPolicy` usando os serviços do WebSphere MQ O atributo `SsecurityPolicy` se aplicará apenas se o serviço especificado na sub-rotina *Service* for o serviço de autorização, ou seja, o OAM padrão O atributo `SecurityPolicy` permite especificar a política de segurança para cada gerenciador de filas. Os valores possíveis são:

Default
Especifique `Default` se desejar que a política de segurança padrão tenha efeito. Se um identificador de segurança do Windows (NT SID) não for transmitido ao OAM para um determinado ID do usuário, será feita uma tentativa de obter o SID apropriado procurando os bancos de dados de segurança relevantes.

NTSIDsRequired
Exige que um SID NT seja passado ao OAM ao executar as verificações de segurança.

Para obter informações sobre o formato da sub-rotina Service, consulte “Formato de sub-rotina de serviço de sistemas Windows” na página 391. Para obter mais informações gerais sobre a segurança, consulte Configurando a segurança no Windows, UNIX and Linux sistemas

A sub-rotina do componente de serviço, `MQSeries.WindowsNT.auth.service` define o componente de serviço de autorização padrão, o OAM. Se essa sub-rotina for removida e o gerenciador de filas reiniciado, o OAM será desativado e nenhuma verificação de autorização será realizada.

Configurando sub-rotinas do serviço de nomes: sistemas Unix e Linux
Insira sua breve descrição aqui; utilizado para primeiro parágrafo e resumo.

Os exemplos a seguir de sub-rotinas no arquivo de configuração do UNIX and Linux para o serviço de nomes especificam um componente de serviço de nomes fornecido pela empresa ABC (fictícia).

```
# Stanza for name service
 Service:
    Name=NameService
    EntryPoints=5

# Stanza for name service component, provided by ABC
 ServiceComponent:
    Service=NameService
    Name=ABC.Name.Service
    Module=/usr/lib/abcname
    ComponentDataSize=1024
```

Figura 74. Sub-rotinas de serviço de nomes em qm.ini (para sistemas UNIX and Linux)

Nota: Em sistemas Windows , as informações de sub-rotina do serviço de nomes são armazenadas no Registro

Atualizando o OAM após mudar a autorização de um usuário
No WebSphere MQ, é possível atualizar as informações do grupo de autorização do OAM imediatamente após alterar a associação do grupo de autorização de um usuário, refletindo as mudanças feitas no nível do sistema operacional, sem precisar parar e reiniciar o gerenciador de fila Para fazer isso, emita o comando **REFRESH SECURITY**.

Nota: Ao mudar autorizações com o comando `setmqaut`, o OAM implementa tais mudanças imediatamente.

Os gerenciadores de filas armazenam dados de autorização em uma fila local chamada SYSTEM.AUTH.DATA.QUEUE. Esses dados são gerenciados pelo `amqzfuma.exe` .

Referências relacionadas
REFRESH SECURITY

Escrevendo e compilando saídas de API

As saídas de API permitem gravar o código que altera o comportamento das chamadas de API do WebSphere MQ, como MQPUT e MQGET, e, em seguida, inserir o código imediatamente antes ou imediatamente depois dessas chamadas.

Nota: Não é suportado no WebSphere MQ para z/OS

Por que usar saídas de API?

Cada um de seus aplicativos tem uma tarefa específica a executar e seu código deve executar essa tarefa com a maior eficiência possível. Em um nível superior, talvez deseje aplicar padrões ou processos de negócios a um determinado gerenciador de filas para **todos** os aplicativos que usam esse gerenciador de filas. É mais eficiente fazer isso acima do nível de aplicativos individuais e, assim, não precisar mudar o código de cada aplicativo afetado.

Eis algumas sugestões de áreas nas quais as saídas de API podem ser úteis:

- Para *segurança*, é possível fornecer autenticação, verificando se os aplicativos estão autorizados a acessar uma fila ou gerenciador de filas. Também é possível policiar o uso de aplicativos da API, autenticando as chamadas de API individuais ou mesmo os parâmetros que usam.
- Para *flexibilidade*, é possível responder a mudanças rápidas em seu ambiente de negócios sem mudar os aplicativos que dependem dos dados nesse ambiente. Você poderia, por exemplo, ter saídas de API que respondem a mudanças nas taxas de juros, taxas de câmbio de moedas ou preço dos componentes em um ambiente de fabricação.

- Para *monitoramento* do uso de uma fila ou gerenciador de filas, é possível rastrear o fluxo de aplicativos e mensagens, registrar os erros nas chamadas de API, configurar trilhas de auditoria para propósitos contábeis ou coletar estatísticas de uso para propósitos de planejamento.

O que acontece quando uma saída de API é executada?

Depois de gravar um programa de saída e identificá-lo para o WebSphere MQ, o gerenciador de filas chama automaticamente seu código de saída nos pontos registrados

As rotinas de saída de API a serem executadas são identificadas em sub-rotinas nos sistemas IBM i, Windows, UNIX and Linux. Este tópico cobre as sub-rotinas nos arquivos de configuração mqs.ini e qm.ini.

A definição das rotinas pode ocorrer em três locais:

1. ApiExitComum, no arquivo mqs.ini , identifica rotinas, para todo o WebSphere MQ, aplicado quando os gerenciadores de fila são iniciados.. Elas podem ser substituídas por rotinas definidas para gerenciadores de filas individuais (consulte o item "3" na página 394 nesta lista).
2. ApiExitModelo, no arquivo mqs.ini , identifica rotinas, para todo o WebSphere MQ, copiado para o conjunto local ApiExit(consulte o item "3" na página 394 nesta lista) quando um novo gerenciador de filas é criado.
3. ApiExitLocal, no arquivo qm.ini, identifica as rotinas que se aplicam a um gerenciador de filas específico.

Quando um novo gerenciador de filas é criado, as definições ApiExitTemplate no mqs.ini são copiadas para as definições ApiExitLocal em qm.ini para o novo gerenciador de filas. Quando um gerenciador de filas é iniciado, as definições ApiExitCommon e ApiExitLocal são usadas. As definições ApiExitLocal substituem as definições ApiExitCommon se ambas identificarem uma rotina com o mesmo nome. O atributo Sequence descrito em "Configurando saídas de API" na página 399 determina a ordem na qual as rotinas definidas nas sub-rotinas são executadas.

Usando saídas de API em diversas instalações do WebSphere MQ

Assegure que as saídas de API gravadas para a versão anterior do WebSphere MQ sejam usadas para trabalhar com todas as versões porque as mudanças feitas nas saídas na versão 7.1 podem não funcionar com uma versão anterior. Para obter mais informações sobre as mudanças feitas nas saídas, consulte "Gravando e compilando saídas e serviços instaláveis" na página 379

As amostras fornecidas para as saídas de API amqsaem e amqsaxe refletem as mudanças necessárias ao gravar saídas. O aplicativo cliente deve assegurar que as bibliotecas do WebSphere MQ corretas que correspondem à instalação do gerenciador de fila com o qual o aplicativo está associado sejam vinculadas a ele antes da ativação do aplicativo

Escrevendo saídas de API

É possível escrever saídas para cada chamada de API usando a linguagem de programação C.

Saídas estão disponíveis para cada chamada de API da seguinte forma:

- MQCB, para registrar novamente um retorno de chamada para a manipulação de objetos especificada e controlar ativação e mudanças no retorno de chamada
- MQCTL, para executar ações de controle nos identificadores de objeto abertos para uma conexão
- MQCONN/MQCONNX, para fornecer uma manipulação de conexões do gerenciador de filas para uso em chamadas de API subsequentes
- MQDISC, para desconectar-se de um gerenciador de filas
- MQBEGIN, para iniciar uma unidade de trabalho (UOW) global
- MQBACK, para restaurar uma UOW
- MQCMIT, para confirmar uma UOW
- MQOPEN, para abrir um recurso do WebSphere MQ para acesso subsequente
- MQCLOSE, para fechar um recurso do WebSphere MQ que foi aberto anteriormente para acesso

- MQGET, para recuperar uma mensagem de uma fila que havia sido aberta anteriormente para acesso
- MQPUT1, para colocar uma mensagem em uma fila
- MQPUT, para colocar uma mensagem em uma fila que havia sido aberta anteriormente para acesso
- MQINQ, para consultar os atributos de um recurso do WebSphere MQ que foi aberto anteriormente para acesso
- MQSET, para configurar os atributos de uma fila que havia sido aberta anteriormente para acesso
- MQSTAT, para recuperar informações de status
- MQSUB, para registrar a assinatura de aplicativos para um determinado tópico
- MQSUBRQ, para fazer uma solicitação de uma assinatura

MQ_CALLBACK_EXIT fornece uma função de saída a ser executada antes e após o processamento de retorno de chamada. Para obter mais informações, veja Retorno de chamada - MQ_CALLBACK_EXIT.

Nas saídas de API, as chamadas assumem o formato geral:

```
MQ_call_EXIT (parameters, context, ApiCallParameters)
```

em que *call* é o nome da chamada MQI sem o prefixo MQ; por exemplo, PUT, GET. Os *parameters* controlam a função da saída, fornecendo primariamente a comunicação entre a saída e os blocos de controle externos MQAXP (a estrutura do parâmetro de saída de API) e MQAXC (estrutura de contexto de saída de API). *context* descreve o contexto no qual a saída de API foi chamada e *ApiCallParameters* representam os parâmetros para a chamada MQI.

Para ajudar a escrever sua saída de API, uma saída de amostra, amqsaxe0.c, é fornecida; essa saída gera as entradas para arquivo especificado. É possível usar essa amostra como seu ponto de início ao escrever saídas. Para obter mais informações sobre como usar a saída de amostra, consulte “O programa de amostra de saída de API” na página 115.

Para obter mais informações sobre as chamadas de saída de API, blocos de controle externo e tópicos associados, consulte Referência de saída de API.

Para obter informações gerais sobre como escrever, compilar e configurar uma saída, consulte “Gravando e compilando saídas e serviços instaláveis” na página 379.

Usando identificadores de mensagens em saídas de API

É possível controlar a quais propriedades de mensagens uma saída de API tem acesso. Propriedades estão associadas a um ExitMsgHandle. Propriedades configuradas em uma saída put são configuradas na mensagem que está sendo efetuado put, mas as propriedades recuperadas em uma saída get não são retornadas ao aplicativo.

Ao registrar uma função de saída MQ_INIT_EXIT usando a chamada MQI MQXEP com **Function** configurado para MQXF_INIT e **ExitReason** configurado como MQXR_CONNECTION, você passa uma estrutura MQXEPO como o parâmetro **ExitOpts**. A estrutura MQXEPO contém o campo ExitProperties, que especifica o conjunto de propriedades a ser disponibilizado para a saída. Ela é especificada como uma sequência de caracteres que representa o prefixo das propriedades, que corresponde a um nome de pasta MQRFH2.

Cada saída de API recebe uma estrutura MQAXP, que contém um campo ExitMsgHandle. Esse campo é configurado para um valor gerado pelo WebSphere MQ e é específico para uma conexão.. Portanto, o identificador permanece inalterado entre as saídas de API de tipos iguais ou diferentes na mesma conexão.

Em um MQ_PUT_EXIT ou MQ_PUT1_EXIT com um **ExitReason** de MQXR_BEFORE, ou seja, uma saída de API executada antes de efetuar put de uma mensagem, quaisquer propriedades (diferentes de propriedades do descritor de mensagens) associadas a ExitMsgHandle quando a saída for concluída serão configuradas na mensagem para a qual put está sendo efetuado. Para evitar que isso aconteça, configure ExitMsgHandle para MQHM_NONE. Também é possível fornecer um identificador de mensagem diferente.

Em um MQ_GET_EXIT, ExitMsgHandle tem as propriedades removidas e é preenchido com as propriedades especificadas no campo ExitProperties quando MQ_INIT_EXIT foi registrado, sem considerar as propriedades do descritor de mensagens. Essas propriedades não são disponibilizadas no aplicativo de get. Se o aplicativo de get tiver especificado um identificador de mensagens no campo MQGMO (opções de get message), quaisquer propriedades associadas a esse identificador, incluindo as propriedades do descritor de mensagens, estarão disponíveis para a saída de API. Para evitar que o ExitMsgHandle seja preenchido com as propriedades, configure-o para MQHM_NONE.

Um programa de amostra, amqsaem0.c, é fornecido para ilustrar o uso de identificadores de mensagens em saídas de API.

Compilando saídas de API

Após ter escrito uma saída, você a compila e vincula da seguinte forma.

Os exemplos a seguir mostram os comandos usados para o programa de amostra descrito em "O programa de amostra de saída de API" na página 115. Para plataformas diferentes de sistemas Windows , é possível localizar o código de saída da API de amostra no *MQ_INSTALLATION_PATH*/samp e a biblioteca compartilhada compilada e vinculada no *MQ_INSTALLATION_PATH*/samp/bin Para sistemas Windows , é possível localizar o código de saída da API de amostra em *MQ_INSTALLATION_PATH*\Tools\c\Samples. *MQ_INSTALLATION_PATH* Representa o diretório no qual o WebSphere MQ foi instalado

Nota para usuários:

1. A orientação sobre a programação de aplicativos de 64 bits é listada em Padrões de codificação em plataformas de 64 bits

Com a introdução de clientes Multicast, as saídas de API e as saídas de conversão de dados precisam ser capazes de executar no lado do cliente, pois algumas mensagens podem não passar pelo gerenciador de filas. As bibliotecas a seguir agora fazem parte dos pacotes do cliente, assim como dos pacotes do servidor:

Tabela 54. Bibliotecas que agora estão nos pacotes do cliente e do servidor

Sistema operacional	**Bibliotecas**
Windows	32 bits e 64 bits: mqm.dll e mqm.pdb
Linux & HP-UX	32 bits e 64 bits: libmqm.s e libmqm_r.so
AIX	32 bits e 64 bits: libmqm.a e libmqm_r.a
Solaris	32 bits e 64 bits: libmqm.so

Compilando saídas de API no Unix e sistemas Linux

Exemplos de como compilar saídas de API em sistemas UNIX e Linux

Em todas as plataformas, o ponto de entrada para o módulo é MQStart.

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

No AIX

Compile o código-fonte de saída de API emitindo um dos seguintes comandos:

Aplicativos de 32 bits

Não encadeado

```
cc -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Encadeado

```
xlc_r -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe_r \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Aplicativos de 64 bits

Não encadeado

```
cc -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Encadeado

```
xlc_r -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe_r \
    amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Na plataforma Itanium HP-UX

Aplicativos de 32 bits

Não encadeado

Compile o código-fonte da Saída de API:

```
c89 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Vincule o código-fonte da Saída de API

```
ld +b: -b amqsaxe.o +ee MQStart -o /var/mqm/exits/amqsaxe
rm amqsaxe.o
```

Encadeado

Compile o código-fonte da Saída de API:

```
c89 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Vincule o código-fonte da Saída de API

```
ld +b: -b amqsaxe.o +ee MQStart -o /var/mqm/exits/amqsaxe_r
rm amqsaxe.o
```

Aplicativos de 64 bits

Não encadeado

Compile o código-fonte da Saída de API:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Vincule o código-fonte da Saída de API

```
ld -b amqsaxe.o +ee MQStart -o  /var/mqm/exits64/amqsaxe
rm amqsaxe.o
```

Encadeado

Compile o código-fonte da Saída de API:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -IMQ_INSTALLATION_PATH/inc
```

Vincule o código-fonte da Saída de API

```
ld -b amqsaxe.o +ee MQStart -o  /var/mqm/exits64/amqsaxe_r
rm amqsaxe.o
```

EmLinux

Compile o código-fonte de saída de API emitindo um dos seguintes comandos:

Aplicativos de 31 bits

Não encadeado

```
 gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Encadeado

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -IMQ_INSTALLATION_PATH/inc
```

Aplicativos de 32 bits

Não encadeado

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Encadeado

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -IMQ_INSTALLATION_PATH/inc
```

Aplicativos de 64 bits

Não encadeado

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

Encadeado

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe_r amqsaxe0.c \
     -IMQ_INSTALLATION_PATH/inc
```

No Solaris

Compile o código-fonte de saída de API emitindo um dos seguintes comandos:

Aplicativos de 32 bits

Plataforma SPARC

```
cc -xarch=v8plus -KPIC -mt -G -o /var/mqm/exits/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/32 -lsocket -lnsl -ldl
```

Plataforma x86-64

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/32 -lsocket -lnsl -ldl
```

Aplicativos de 64 bits

Plataforma SPARC

```
cc -xarch=v9 -KPIC -mt -G -o /var/mqm/exits64/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/64 -lsocket -lnsl -ldl
```

Plataforma x86-64

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/amqsaxe \
   amqsaxe0.c -IMQ_INSTALLATION_PATH/inc \
   -R/usr/lib/64 -lsocket -lnsl -ldl
```

Em Sistemas Windows

Compile e vincule o programa de saída da API de amostra, `amqsaxe0.c`, no Windows

Um arquivo de manifesto é um documento XML opcional que contém a informação de versão, ou qualquer outra, que possa ser integrada a um aplicativo compilado ou DLL.

Se você não tiver esse documento, omita o parâmetro -manifest *manifest.file* no comando **mt**.

Adapte os comandos nos exemplos em Figura 75 na página 399 ou Figura 76 na página 399 para compilar e vincular `amqsaxe0.c` no Windows .. Os comandos trabalham com Microsoft Visual Studio 2005, 2008 ou 2010. Os exemplos assumem que o diretório WebSphere MQ `C:\Program Files\IBM\WebSphere MQ\tools\c\samples` é o diretório atual.

32 bits

```
cl /c /nologo /MD /Foamqsaxe0.obj amqsaxe0.c

link /nologo /dll /def:amqsaxe.def

amqsaxe0.obj \
     /manifest /out:amqsaxe.dll

mt -nologo -manifest amqsaxe.dll.manifest \
   -outputresource:amqsaxe.dll;2
```

Figura 75. Compilar e vincular `amqsaxe0.c` em 32 bits Windows

64 bits

```
cl /c /nologo /MD /Foamqsaxe0.obj amqsaxe0.c

link /nologo /dll /def:amqsaxe.def \
     /libpath:..\..\lib64 \


amqsaxe0.obj /manifest /out:amqsaxe.dll

mt -nologo -manifest amqsaxe.dll.manifest \
   -outputresource:amqsaxe.dll;2
```

Figura 76. Compilar e vincular `amqsaxe0.c` em 64 bits Windows

Conceitos relacionados

"O programa de amostra de saída de API" na página 115
A saída API de amostra gera um rastreio de MQI para um arquivo especificado pelo usuário com um prefixo definido na variável de ambiente MQAPI_TRACE_LOGFILE.

Configurando saídas de API

Configure IBM WebSphere MQ para ativar as saídas de API mudando as informações de configuração.

Para mudar as informações de configuração, deve-se mudar as sub-rotinas que definem as rotinas de saída e a sequência na qual elas são executadas. Essas informações podem ser mudadas das seguintes maneiras:

- Usando o IBM WebSphere MQ Explorer (Em Windows e Linux (plataformasx86 e x86-64))
- Usando o comando **amqmdain** (em Windows)
- Usando os arquivos mqs.ini e qm.ini diretamente (nos sistemas Windows, UNIX and Linux .

  O arquivo mqs.ini contém informações relevantes para todos os gerenciadores de filas em um nó específico. É possível localizá-lo no diretório `/var/mqm` no UNIX and Linux e no WorkPath especificado na chave `HKLM\SOFTWARE\IBM\WebSphere MQ` nos sistemas Windows.

  O arquivo qm.ini contém informações relevantes para um gerenciador de filas específico. Há um arquivo de configuração do gerenciador de filas para cada gerenciador de filas, retido na raiz da árvore de

diretórios ocupada pelo gerenciador de filas. Por exemplo, o caminho e o nome para um arquivo de configuração para um gerenciador de filas denominado QMNAME é:

Nos sistemas UNIX and Linux:

```
/var/mqm/qmgrs/QMNAME/qm.ini
```

Nos sistemas Windows:

```
C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\qm.ini
```

Antes de editar um arquivo de configuração, faça backup dele para que tenha uma cópia para a qual possa reverter se surgir a necessidade.

Você pode editar um arquivo de configuração:

– Automaticamente, usando os comandos que alteram a configuração dos gerenciadores de fila no nó
– Manualmente, usando um editor de texto padrão

Se você configurar um valor incorreto em um atributo do arquivo de configuração, o valor será ignorado e uma mensagem do operador será emitida para indicar o problema. (O efeito é o mesmo que perder totalmente o atributo).

Sub-rotinas para configuração

As sub-rotinas que devem ser mudadas são as seguintes:

ApiExitCommon

Definido em mqs.ini e no IBM WebSphere MQ Explorer na página de propriedades IBM WebSphere MQ , em Saídas

Quando algum gerenciador de filas for iniciado, os atributos nesta sub-rotina serão lidos e, em seguida, substituídos pelas saídas de API definidas em qm.ini.

ApiExitTemplate

Definido em mqs.ini e no IBM WebSphere MQ Explorer na página de propriedades IBM WebSphere MQ , em Saídas

Quando algum gerenciador de filas for criado, os atributos nesta sub-rotina serão copiados no arquivo qm.ini recém-criado na sub-rotina ApiExitLocal.

ApiExitLocal

Definido em qm.ini e no IBM WebSphere MQ Explorer na página de propriedades do gerenciador de filas, em Saídas.

Quando o gerenciador de filas é iniciado, as saídas de API definidas aqui substituem os padrões definidos em mqs.ini.

Atributos para as sub-rotinas

- Nomeie a saída de API usando o seguinte atributo:

  Name=ApiExit_name
  O nome descritivo da saída de API passada para ela no campo ExitInfoName da estrutura MQAXP.

  Este nome deve ser exclusivo, sem ultrapassar 48 caracteres, e conter apenas caracteres válidos para os nomes de objetos do IBM WebSphere MQ (por exemplo, nomes de fila).

- Identifique o módulo e o ponto de entrada do código de saída de API para execução usando os seguintes atributos:

  Function=function_name
  O nome do ponto de entrada da função no módulo que contém o código de saída de API. Este ponto de entrada é a função MQ_INIT_EXIT.

O comprimento deste campo está limitado a MQ_EXIT_NAME_LENGTH.

Module=module_name
O módulo que contém o código de saída de API.

Se esse campo contiver o nome de caminho completo do módulo, ele será utilizado dessa forma.

Se este campo contiver apenas o nome do módulo, o módulo será localizado usando o atributo `ExitsDefaultPath` no `ExitPath` em qm.ini.

Em plataformas que suportam bibliotecas encadeadas separadas, deve-se fornecer uma versão encadeada e não encadeada do módulo de saída de API. A versão encadeada deve ter um sufixo `_r`. A versão encadeada do stub de aplicativo IBM WebSphere MQ anexa `_r` implicitamente ao nome do módulo fornecido antes de ser carregada.

O comprimento deste campo é limitado ao comprimento máximo do caminho que a plataforma suporta.

- Opcionalmente, passe os dados com a saída que usa o seguinte atributo:

  Data=data_name
  Dados a serem passados para a saída de API no campo ExitData da estrutura MQAXP.

  Se você incluir este atributo, espaços em branco iniciais e finais serão removidos, a sequência restante será truncada para 32 caracteres e o resultado será passado para a saída. Se você omitir este atributo, o valor padrão de 32 espaços em branco é passado para a saída.

  O comprimento máximo deste campo é de 32 caracteres.

- Identifique a sequência desta saída em relação a outras saídas usando o seguinte atributo:

  Sequence=sequence_number
  A sequência na qual esta saída de API é chamada em relação a outras saídas de API. Uma saída com um baixo número de sequência é chamada antes de uma saída com um número de sequência mais alto. Não há necessidade para que a numeração de sequência de saídas seja contígua. Uma sequência de 1, 2, 3 possui o mesmo resultado que uma sequência de 7, 42, 1096. Se duas saídas tiverem o mesmo número de sequência, o gerenciador de filas decidirá qual chamar primeiro. É possível informar qual foi chamado após o evento colocando a hora ou um marcador no ExitChainArea indicado pelo ExitChainAreaPtr em MQAXP ou gravando seu próprio arquivo de log.

  Este atributo é um valor numérico não assinado.

Sub-rotinas de amostra

O arquivo mqs.ini de amostra contém as seguintes sub-rotinas:

ApiExitTemplate
Esta sub-rotina define uma saída com o nome descritivo `OurPayrollQueueAuditor`, nome do módulo `auditor` e sequência número 2. Um valor de dados de 123 é transmitido para a saída.

ApiExitCommon
Esta sub-rotina define uma saída com o nome descritivo `MQPoliceman`, nome do módulo `tmqp` e sequência número 1. Os dados transmitidos são uma Instrução (`CheckEverything`)

```
mqs.ini

   ApiExitTemplate:
      Name=OurPayrollQueueAuditor
      Sequence=2
      Function=EntryPoint
      Module=/usr/ABC/auditor
      Data=123
   ApiExitCommon:
      Name=MQPoliceman
      Sequence=1
      Function=EntryPoint
      Module=/usr/MQPolice/tmqp
      Data=CheckEverything
```

O seguinte arquivo qm.ini de amostra contém uma definição ApiExitLocal de uma saída com o nome descritivo `ClientApplicationAPIchecker`, nome do módulo `ClientAppChecker` e número de sequência 3.

```
qm.ini

   ApiExitLocal:
      Name=ClientApplicationAPIchecker
      Sequence=3
      Function=EntryPoint
      Module=/usr/Dev/ClientAppChecker
      Data=9.20.176.20
```

Programas de Saída de Canal para Canais de Mensagens

Esta coleção de tópicos contém informações sobre programas de saída do canal WebSphere MQ para canais do sistema de mensagens.

Message channel agents (MCAs) também podem chamar saídas de conversão de dados. Para obter mais informações sobre saídas de conversão de dados, veja "Escrevendo saídas de conversão de dados" na página 421.

Algumas dessas informações também se aplicam a saídas em canais MQI, que conectam clientes MQI do WebSphere MQ a gerenciadores de filas Para obter mais informações, veja Programas de saída de canal para canais MQI.

Os programas de saída do canal são chamados nos locais definidos no processamento executado pelos programas MCA.

Alguns desses programas de saída do usuário trabalharam em pares complementares. Por exemplo, se um programa de saída de usuário for chamado pelo MCA de envio para criptografar as mensagens para transmissão, o processo complementar deve estar funcionando na extremidade de recebimento para reverter o processo.

O Tabela 55 na página 402 mostra os tipos de saída do canal que ficam disponíveis para cada tipo de canal.

Tabela 55. Saídas de Canal Disponíveis para cada tipo de canal

Tipo de Canal	**Saída de mensagen**	**saída de nova tentativa de mensagem**	**Saída de recepção**	**Saída de Segurança**	**Saída de envio**	**saída de definição automática**
Canal Emissor	Sim		Sim	Sim	Sim	
Canal servidor	Sim		Sim	Sim	Sim	
canal do emissor de clusters	Sim		Sim	Sim	Sim	Sim
Canal receptor	Sim	Sim	Sim	Sim	Sim	Sim
Canal solicitador	Sim	Sim	Sim	Sim	Sim	
canal do receptor de clusters	Sim	Sim	Sim	Sim	Sim	Sim

Tabela 55. Saídas de Canal Disponíveis para cada tipo de canal (continuação)						
Tipo de Canal	**Saída de mensagen**	**saída de nova tentativa de mensagem**	**Saída de recepção**	**Saída de Segurança**	**Saída de envio**	**saída de definição automática**
canal de conexão do cliente			Sim	Sim	Sim	
canal de conexão do servidor			Sim	Sim	Sim	Sim

Se você pretende executar as saídas de canal em um cliente, não será possível usar a variável de ambiente MQSERVER. Em vez disso, crie e faça referência a uma tabela de definições de canal do cliente (CCDT) conforme descrito em Tabela de definição de canal do cliente .

Visão geral de processamento

Uma visão geral de como os MCAs usam programas de saída do canal.

Na inicialização, os MCAs trocam um diálogo de inicialização para sincronizar o processamento. Em seguida, alternam para uma troca de dados que inclui as saídas de segurança. Essas saídas devem terminar com sucesso para que a fase de inicialização seja concluída e para permitir que as mensagens sejam transferidas.

A fase de verificação de segurança é um loop, conforme mostrado em Figura 77 na página 403.

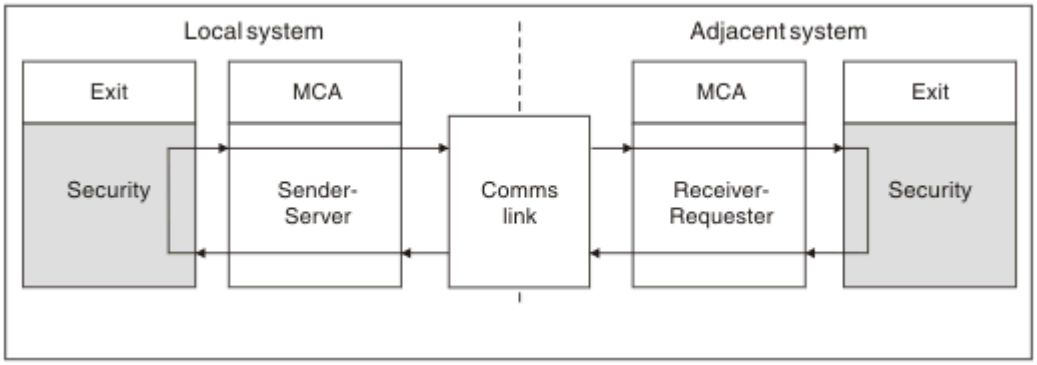


Figura 77. Loop da saída de segurança

Durante a fase de transferência de mensagem, o MCA de envio recebe as mensagens de uma fila de transmissão, chama a saída de mensagem, chama a saída de envio e, em seguida, envia a mensagem para o MCA de recebimento, conforme mostrado em Figura 78 na página 404.

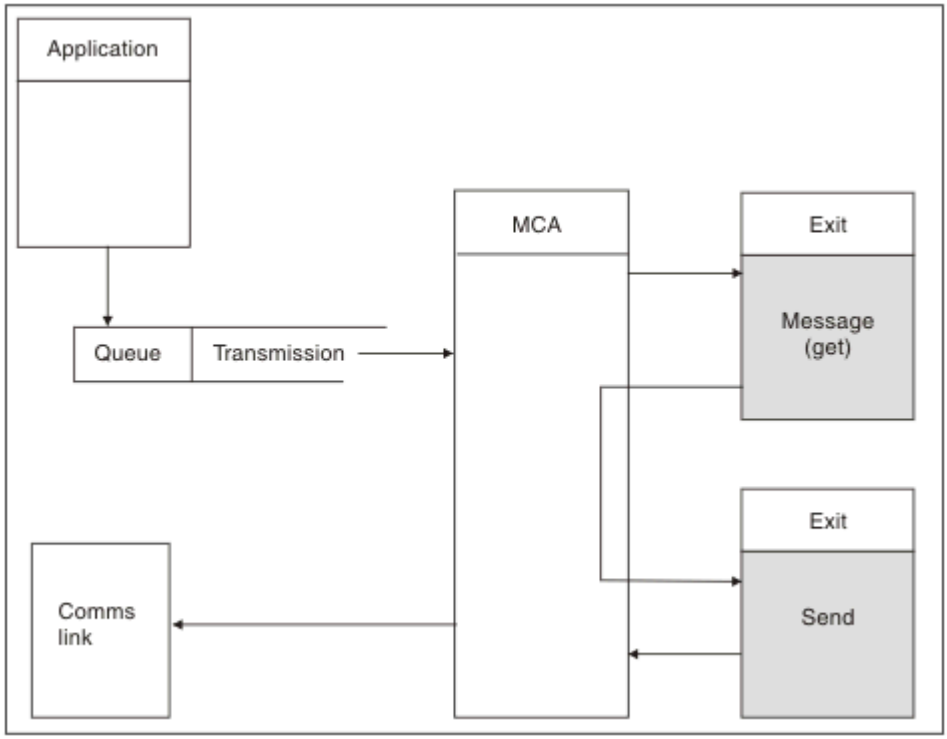


Figura 78. Exemplo de uma saída de envio na extremidade do emissor do canal de mensagens

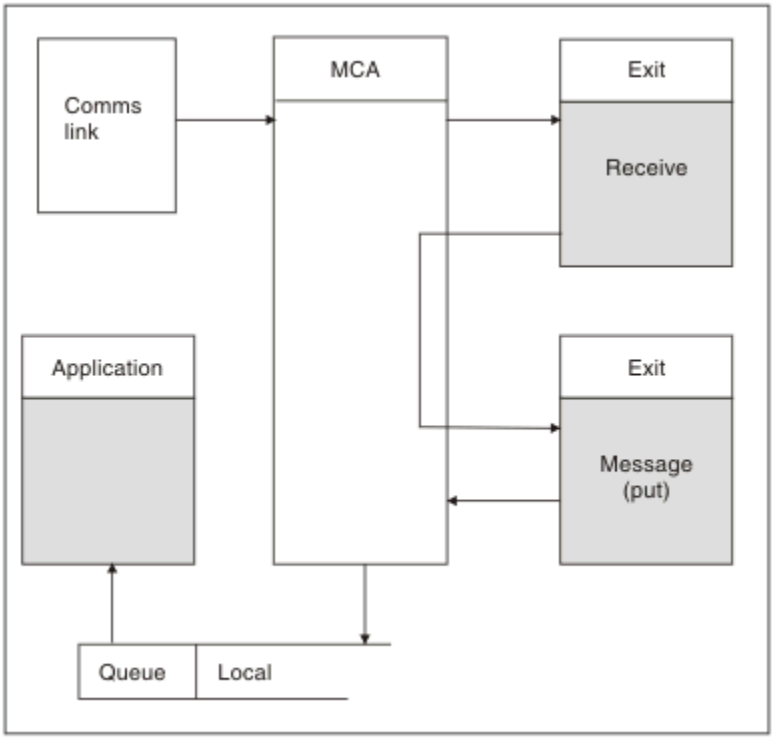


Figura 79. Exemplo de uma saída de recebimento no receptor final do canal de mensagens

O MCA de recebimento recebe uma mensagem do link de comunicações, chama a saída de recebimento, chama a saída de mensagem e, em seguida, coloca a mensagem na fila local, conforme mostrado em Figura 79 na página 404. (A saída de recebimento pode ser chamada mais de uma vez antes da saída de mensagem ser chamada.)

Gravando programas de saída do canal

É possível usar as seguintes informações para ajudá-lo a gravar programas de saída do canal.

As saídas de usuário e programas de saída do canal podem usar todas as chamadas MQI, exceto conforme o observado nas seções a seguir. Para MQ V7 e posterior, a estrutura MQCXP versão 7 e superior contém a manipulação de conexão hConn, que pode ser usada em vez de emitir o MQCONN. Para versões anteriores, para obter a manipulação de conexões, uma chamada MQCONN deverá ser emitida, embora um aviso MQRC_ALREADY_CONNECTED seja retornado porque o próprio canal está conectado ao gerenciador de filas.

Observe que a saída do canal deve ser thread-safe.

Para saídas nos canais de conexão do cliente, o gerenciador de filas ao qual a saída tenta se conectar depende de como a saída foi vinculada. Se a saída foi vinculada ao MQM do MQM.LIB e você não especificar um nome de gerenciador de filas na chamada MQCONN, a saída tenta se conectar ao gerenciador de filas padrão em seu sistema. Se a saída foi vinculada ao MQM do MQM.LIB e você especificar o nome do gerenciador de filas que foi transmitido para a saída por meio do campo QMgrName do MQCD, a saída tenta se conectar a esse gerenciador de filas. Se a saída tiver sido vinculada com MQIC.LIB ou qualquer outra biblioteca, a chamada MQCONN falhará especificando um nome do gerenciador de filas ou não.

É necessário evitar alterar o estado da transação associada ao hConn passado em uma saída de canal; não se deve usar os verbos MQCMIT, MQBACK ou MQDISC com o canal hConn e não é possível usar o verbo MQBEGIN especificando o canal hConn.

Se MQCONNX for usado especificando MQCNO_HANDLE_SHARE_BLOCK ou MQCNO_HANDLE_SHARE_NO_BLOCK para criar uma nova conexão do IBM WebSphere MQ, então será sua responsabilidade assegurar que a conexão seja corretamente gerenciada e desconectar do gerenciador de filas corretamente. Por exemplo, uma saída de canal que cria uma nova conexão ao gerenciador de filas em cada chamada sem desconectar resulta em um acúmulo de manipulações de conexão e um aumento no número de encadeamentos de agente.

Uma saída é executada no mesmo encadeamento que o MCA sozinho e usa a mesma manipulação de conexões. Portanto, ela é executada dentro do mesmo UOW que o MCA e quaisquer chamadas feitas no ponto de sincronização são confirmadas ou voltadas pelo canal ao final do lote.

Portanto, uma saída de mensagem de canal poderia enviar mensagens de notificação que são confirmadas apenas nessa fila quando o lote que contém a mensagem original for confirmado. Portanto, é possível emitir as chamadas MQI do ponto de sincronização a partir de uma saída de mensagem de canal.

Uma saída do canal pode mudar os campos no MQCD. No entanto, não há ação sobre essas mudanças, exceto nas circunstâncias listadas. Se um programa de saída de canal mudar um campo na estrutura de dados MQCD, o novo valor será ignorado pelo processo do canal IBM WebSphere MQ . No entanto, o novo valor permanece no MQCD e é passado a qualquer saída restante em uma sequência de saída e a qualquer conversa que compartilhando instância do canal. Para obter mais informações, consulte Mudando campos MQCD em uma saída de canal

Além disso, para programas escritos em C, a função de biblioteca C não reentrante não deve ser usada em um programa de saída de canal.

Se você usar diversas bibliotecas de saída de canal simultaneamente, podem surgir problemas em algumas plataformas UNIX and Linux se o código para duas saídas diferentes contiver funções nomeadas de forma idêntica. Quando uma saída do canal estiver carregada, o carregador dinâmico resolverá os nomes de função na biblioteca de saída para os endereços em que a biblioteca estiver carregada. Se duas bibliotecas de saída definirem funções separadas que por acaso possuem nomes idênticos, esse processo de resolução pode resolver incorretamente os nomes de função de uma biblioteca para usar as funções de outra. Se este problema ocorrer, especifique ao vinculador que ele deve exportar apenas a saída necessária e as funções MQStart, uma vez que essas funções não são afetadas. Outras funções devem receber visibilidade local para que não sejam usadas pelas funções fora de suas próprias bibliotecas de saída. Consulte a documentação do vinculador para obter informações adicionais.

Todas as saídas são chamadas com uma estrutura do parâmetro de saída do canal (MQCXP), uma estrutura de definição do canal (MQCD), um buffer de dados preparado, parâmetro de comprimento de dados e parâmetro de comprimento de buffer. O comprimento do buffer não deve ser excedido:

- Para saídas de mensagem, deve-se permitir que a maior mensagem necessária seja enviada através do canal, além do comprimento da estrutura MQXQH.
- Para as saídas de envio e recebimento, o maior buffer que deve ser permitido é o seguinte:

  LU6.2
  32 KB

  TCP:
  32 KB

  Nota: O comprimento máximo utilizável pode ser 2 bytes a menos que esse comprimento. Verifique o valor retornado em MaxSegmentLength para obter detalhes. Para obter informações adicionais sobre MaxSegmentLength, consulte MaxSegmentLength.

  NetBIOS:
  64 KB

  SPX:
  64 KB

  Nota: As saídas de recebimento nos canais emissores e as saídas de emissor nos canais receptores usam buffers de 2 KB para TCP.
- Para saídas de segurança, o recurso de enfileiramento distribuído aloca um buffer de 4000 bytes.

É permissível para a saída retornar um buffer alternativo juntamente com os parâmetros relevantes. Consulte “Programas de Saída de Canal para Canais de Mensagens” na página 402 para os detalhes da chamada.

Gravando programas de saída de canal em sistemas Windows, UNIX and Linux
É possível usar as informações a seguir para ajudar a gravar programas de saída de canal para sistemas Windows, UNIX and Linux .

Siga as instruções descritas em “Gravando e compilando saídas e serviços instaláveis” na página 379. Use as informações específicas da saída do canal a seguir, conforme apropriado:

A saída deve ser escrita em C e é uma DLL no Windows.

Defina uma rotina MQStart() simulada na saída e especifique `MQStart` como o ponto de entrada na biblioteca. Figura 80 na página 406 mostra como configurar uma entrada para seu programa:

```
#include <cmqec.h>

void MQStart() {;} /* dummy entry point - for consistency only */
void MQENTRY ChannelExit ( PMQCXP  pChannelExitParms,
                           PMQCD   pChannelDefinition,
                           PMQLONG pDataLength,
                           PMQLONG pAgentBufferLength,
                           PMQVOID pAgentBuffer,
                           PMQLONG pExitBufferLength,
                           PMQPTR  pExitBufferAddr)
{
... Insert code here
}
```

Figura 80. Código-fonte de amostra para uma saída do canal

Ao gravar saídas de canal para Windows usando Visual C + +, você deve gravar seu próprio arquivo DEF . Um exemplo de como fazer isso é mostrado em Figura 81 na página 407. Para obter informações adicionais sobre como escrever programas de saída do canal, consulte “Gravando programas de saída do canal” na página 405.

```
EXPORTS
ChannelExit
```

Figura 81. Arquivo DEF de amostra para Windows

Programas de saída de segurança do canal
É possível usar programas de saída de segurança para verificar se o parceiro na outra extremidade de um canal é genuíno. Isso é conhecido como autenticação. Para especificar que um canal deve usar uma saída de segurança, especifique o nome de saída no campo SCYEXIT da definição de canal.

Nota: A autenticação também pode ser atingida com registros de autenticação de canal. Registros de autenticação de canal fornecem grande flexibilidade para evitar acesso aos gerenciadores de filas por determinados usuários e canais e no mapeamento de usuários remotos para identificadores de usuários do IBM WebSphere MQ. Suporte a SSL e TLS também é fornecido pelo IBM WebSphere MQ para autenticar seus usuários e fornecer criptografia e verificações de integridade de dados para seus dados. Para obter mais informações sobre SSL e TLS, consulte o suporte do WebSphere MQ para SSL e TLS No entanto, se ainda requerer formas mais sofisticadas (ou diferentes) de processamento de segurança e outros tipos de verificações e estabelecimento de contexto de segurança, considere gravar as saídas de segurança.

Para saídas de segurança gravadas antes do IBM WebSphere MQ Version 7.1 vale a pena observar que versões anteriores do IBM WebSphere MQ consultavam o provedor de soquetes seguros subjacentes (por exemplo, GSKit) para determinar o Subject Distinguished Name (SSLPEER) e Issuer Distinguished Name (SSLCERTI) do certificado do parceiro remoto. No IBM WebSphere MQ Version 7.1, suporte foi incluído para um intervalo de novos atributos de segurança. Para acessar esses atributos, o IBM WebSphere MQ Version 7.1 obtém a codificação DER do certificado e a usa para determinar o Subject e Issuer DN. Os atributos Subject e Issuer DN aparecem nos atributos de status do canal a seguir:

- SSLPEER (PCF selector MQCACH_SSL_SHORT_PEER_NAME)
- SSLCERTI (PCF selector MQCACH_SSL_CERT_ISSUER_NAME)

Esses valores são retornados pelos comandos de status do canal, assim como pelos dados passados às saídas de segurança do canal listadas, conforme mostrado:

- MQCD SSLPeerNamePtr
- MQCXP SSLRemCertIssNamePtr

No IBM WebSphere MQ Version 7.1, um atributo SERIALNUMBER também é incluído no DN do assunto e contém o número de série do certificado do parceiro remoto. Além disso, alguns atributos DN são retornados em uma sequência diferente das liberações anteriores. Consequentemente, a composição dos campos SSLPEER e SSLCERTI está alterada no Version 7.1 de liberações anteriores e recomenda-se, portanto, que qualquer saída de segurança ou aplicativo dependente desses campos sejam examinados e atualizados.

Os filtros de nome do peer do WebSphere MQ existentes especificados por meio do campo SSLPEER de uma definição de canal não são afetados e continuarão operando da mesma maneira que em liberações anteriores. Isso ocorre porque o algoritmo correspondente do nome do peer do WebSphere MQ foi atualizado para processar filtros SSLPEER existentes sem qualquer necessidade de alterar as definições de canal. Essa mudança mais provavelmente afetará as saídas de segurança e os aplicativos que dependem dos valores de Subject DN e Issuer DN retornados pela interface de programação do PCF.

Uma saída de segurança pode ser escrita em C ou Java.

Programas da saída de segurança de canal são chamados nos locais a seguir no ciclo de processamento de um MCA:

- Na inicialização e na finalização do MCA.
- Imediatamente após a negociação de dados inicial ser concluída na inicialização do canal. A extremidade do receptor ou do servidor do canal pode iniciar uma troca de mensagem de segurança com a extremidade remota, fornecendo uma mensagem a ser entregue à saída de segurança na extremidade remota. Também pode recusar a fazer isso. O programa de saída é iniciado novamente para processar qualquer mensagem de segurança recebida da extremidade remota.

- Imediatamente após a negociação de dados inicial ser concluída na inicialização do canal. A extremidade do emissor ou do solicitante do canal processa uma mensagem de segurança recebida da extremidade remota ou inicia uma troca de segurança quando a extremidade remota não puder. O programa de saída é iniciado novamente para processar todas as mensagens de segurança subsequentes que possam ser recebidas.

Um canal do solicitante nunca é chamado com MQXR_INIT_SEC. O canal notifica o servidor de que tem o programa de saída de segurança e o servidor tem, então, a oportunidade de iniciar uma saída de segurança. Se ele não tiver um, informará o solicitante e um fluxo de comprimento zero será retornado para o programa de saída.

Nota: Evite enviar mensagens de segurança de comprimento zero.

Exemplos dos dados trocados por programas de saída de segurança estão ilustrados nas figuras Figura 82 na página 409 a Figura 85 na página 411. Esses exemplos mostram a sequência de eventos que ocorrem envolvendo a saída de segurança do receptor e a saída de segurança do emissor. Linhas sucessivas nas figuras representam a passagem de tempo. Em alguns casos, os eventos no receptor e no emissor não são correlacionados e, portanto, podem ocorrer ao mesmo tempo ou em momentos diferentes. Em outros casos, um evento em um programa de saída resulta em um evento complementar que ocorre posteriormente no outro programa de saída. Por exemplo, emFigura 82 na página 409:

1. O receptor e o emissor são chamados, cada um, com MQXR_INIT, mas essas chamadas não são correlacionadas e, portanto, podem ocorrer ao mesmo tempo ou em momentos diferentes.
2. O receptor é chamado, em seguida, com MQXR_INIT_SEC, mas retorna MQXCC_OK que não requer evento complementar na saída do emissor.
3. O emissor é chamado, em seguida, com MQXR_INIT_SEC. Isso não está correlacionado à chamada do receptor com MQXR_INIT_SEC. O emissor retorna MQXCC_SEND_SEC_MSG, que causa um evento complementar na saída do receptor.
4. O receptor é, então, chamado com MQXR_SEC_MSG e retorna MQXCC_SEND_SEC_MSG, que causa um evento complementar na saída do emissor.
5. O emissor é, então, chamado com MQXR_SEC_MSG e retorna MQXCC_OK que não requer evento complementar na saída do receptor.

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_OK	
	Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_OK
Message transfer begins	

Figura 82. Troca iniciada pelo emissor com acordo

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_OK	
	Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_OK	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_SUPPRESS_FUNCTION
Channel closes	
Invoked with MQXR_TERM Responds with MQXCC_OK	Invoked with MQXR_TERM Responds with MQXCC_OK

Figura 83. Troca iniciada pelo emissor sem acordo

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_OK	
Message transfer begins	
Invoked with MQXR_TERM Responds with MQXCC_OK	Invoked with MQXR_TERM Responds with MQXCC_OK

Figura 84. Troca iniciada pelo receptor com acordo

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_OK
Invoked with MQXR_SEC_MSG Responds with MQXCC_SUPPRESS_FUNCTION	
Channel closes	

Figura 85. Troca iniciada pelo receptor sem acordo

O programa da saída de segurança de canal é passado a um buffer de agente que contém dados de segurança, excluindo qualquer cabeçalho de transmissão, gerado pela saída de segurança. Esses dados podem ser quaisquer dados adequados para que a extremidade do canal seja capaz de executar a validação de segurança.

O programa de saída de segurança em ambas as extremidades de envio e de recebimento do canal de mensagens pode retornar um de dois códigos de resposta para qualquer chamada:

- Troca de segurança finalizada sem erros
- Suprimir o canal e fechar

Nota:

1. As saídas de segurança do canal geralmente funcionam em pares. Ao definir os canais apropriados, certifique-se de que os programas de saída compatíveis sejam denominados para ambas as extremidades do canal.
2. No IBM i, os programas de saída de segurança que foram compilados com "Usar autoridade adotada" (USEADPAUT = *YES) podem adotar a autoridade QMQM ou QMQMADM. Tome cuidado para que a saída não use esse recurso para constituir um risco de segurança ao seu sistema.
3. Em um canal SSL no qual a outra extremidade do canal fornece um certificado, a saída de segurança recebe o Nome distinto do sujeito deste certificado no campo MQCD acessado por SSLPeerNamePtr e o Nome distinto do emissor no campo MQCXP acessado por SSLRemCertIssNamePtr. Os usos desse nome podem ser:
   - Para restringir o acesso pelo canal SSL.
   - Para mudar MQCD.MCAUserIdentifier com base no nome.

Conceitos relacionados

Registros de Autenticação de Canal

Conceitos do Secure Sockets Layer (SSL) e Transport Layer Security (TLS)

Escrevendo uma saída de segurança

É possível escrever uma saída de segurança usando o código de estrutura básica da saída de segurança.

Figura 86 na página 412 ilustra como escrever uma saída de segurança.

```
void MQENTRY MQStart() {;}
void MQENTRY EntryPoint (PMQVOID pChannelExitParms,
                         PMQVOID pChannelDefinition,
                         PMQLONG pDataLength,
                         PMQLONG pAgentBufferLength,
                         PMQVOID pAgentBuffer,
                         PMQLONG pExitBufferLength,
                         PMQPTR  pExitBufferAddr)
{
  PMQCXP pParms = (PMQCXP)pChannelExitParms;
  PMQCD  pChDef = (PMQCD)pChannelDefinition;
  /* TODO: Add Security Exit Code Here */

}
```

Figura 86. Código de estrutura básica da saída de segurança

O MQStart padrão do Ponto de entrada do WebSphere MQ deve existir mas não é necessário para executar nenhuma função. O nome da função (EntryPoint, neste exemplo) pode ser mudado, mas a função deve ser exportada quando a biblioteca for compilada e vinculada. Como no exemplo anterior, os ponteiros pChannelExitParms devem ter cast efetuado para PMQCXP e pChannelDefinition deve ter cast efetuado para PMQCD. Para obter informações gerais sobre como chamar saídas do canal e sobre o uso de parâmetros, consulte MQ_CHANNEL_EXIT. Esses parâmetros são usados em uma saída de segurança da seguinte forma:

PMQVOID pChannelExitParms
entrada/saída

Ponteiro para a estrutura MQCXP – efetuar cast para PMQCXP para acessar campos. Essa estrutura é usada para comunicação entre a Saída e o MCA. Os campos a seguir no MQCXP são de interesse especial para Saídas de segurança:

ExitReason
Informa a Saída de segurança o estado atual na troca de segurança e é usado ao decidir qual ação tomar.

ExitResponse
A resposta ao MCA que determina o próximo estágio na troca de segurança.

ExitResponse2
Sinalizações de controle extra para gerir como o MCA interpreta a resposta da Saída de segurança.

ExitUserArea
16 bytes (máximo) de armazenamento que podem ser usados pela Saída de segurança para manter o estado entre chamadas.

ExitData
Contém os dados especificados no campo SCYDATA da definição de canal (32 bytes preenchidos à direita com espaços em branco).

PMQVOID pChannelDefinition
entrada/saída

Ponteiro para a estrutura MQCD – efetuar cast para PMQCD para acessar campos. Esse parâmetro contém a definição do canal. Os campos a seguir no MQCD são de interesse especial para Saídas de segurança:

ChannelName
O nome do canal (20 bytes preenchidos à direita com espaços em branco).

ChannelType
Um código que define o tipo de canal.

Identificador de usuário do MCA
Esse grupo de três campos é inicializado para o valor do campo MCAUSER especificado na definição do canal. Qualquer identificador de usuário especificado pela Saída de segurança nesses campos é usado para controle de acesso (não aplicável a canais SDR, SVR, CLNTCONN ou CLUSSDR).

MCAUserIdentifier
Primeiros 12 bytes do identificador preenchidos à direita com espaços em branco.

LongMCAUserIdPtr
Ponteiro para um buffer que contém o identificador de comprimento integral (não garantida finalização nula) tem prioridade sobre MCAUserIdentifier.

LongMCAUserIdLength
Comprimento da sequência apontada por LongMCAUserIdPtr – deve ser configurado se LongMCAUserIdPtr estiver configurado.

Identificador de usuário remoto
Se aplica somente a pares de canais CLNTCONN/SVRCONN. Se nenhuma Saída de segurança CLNTCONN for definida, então, esses três campos serão inicializados pelo MCA do cliente, portanto, eles podem conter um identificador de usuário do ambiente do cliente que pode ser usado por uma Saída de segurança SVRCONN para autenticação e ao especificar o Identificador de usuário do MCA. Se uma Saída de segurança CLNTCONN for definida, então, esses campos não serão inicializados e podem ser configuradas pelo Saída de segurança CLNTCONN ou mensagens de segurança podem ser usadas para passar um identificador de usuário do Cliente para o Servidor.

Identificador de usuário remoto
Primeiros 12 bytes do identificador preenchidos à direita com espaços em branco.

LongRemoteUserIdPtr
Ponteiro para um buffer que contém o identificador de comprimento integral (não garantida finalização nula) tem prioridade sobre RemoteUserIdentifier.

LongRemoteUserIdLength
Comprimento da sequência apontada por LongRemoteUserIdPtr – deve ser configurado se LongRemoteUserIdPtr estiver configurado.

PMQLONG pDataLength
entrada/saída

Ponteiro para MQLONG. Contém o comprimento de qualquer Saída de segurança contida no AgentBuffer após a chamada da Saída de segurança. Deve ser configurado por uma Saída de segurança para o comprimento de qualquer mensagem que está sendo enviada no AgentBuffer ou ExitBuffer.

PMQLONG pAgentBufferLength
entrada

Ponteiro para MQLONG. O comprimento dos dados contidos no AgentBuffer na chamada da Saída de segurança.

PMQVOID pAgentBuffer
entrada/saída

Na chamada da Saída de segurança, aponta para qualquer mensagem enviada da saída do parceiro. Se ExitResponse2 na estrutura MQCXP tiver a sinalização MQXR2_USE_AGENT_BUFFER configurada (padrão), então, uma Saída de segurança precisa configurar esse parâmetro para apontar para qualquer dado de mensagem que esteja sendo enviado.

PMQLONG pExitBufferLength
entrada/saída

Ponteiro para MQLONG. Esse parâmetro é inicializado para 0 na primeira chamada de uma Saída de segurança e o valor retornado é mantido entre as chamadas à Saída de segurança durante uma troca de segurança.

PMQPTR pExitBufferAddr
entrada/saída

Esse parâmetro é inicializado para um ponteiro nulo na primeira chamada de uma Saída de segurança e o valor retornado é mantido entre as chamadas à Saída de segurança durante uma troca de segurança. Se a sinalização MQXR2_USE_EXIT_BUFFER for configurada no ExitResponse2 na estrutura MQCXP, então, uma Saída de segurança precisa configurar esse parâmetro para apontar para qualquer dado de mensagem que esteja sendo enviado.

Diferenças em comportamento entre saídas de segurança definidas em pares de canais CLNTCONN/SVRCONN e outros pares de canais
As saídas de segurança podem ser definidas em todos os tipos de canal. No entanto, o comportamento das saídas de segurança definidas nos pares de canal CLNTCONN/SVRCONN é um pouco diferente das saídas de segurança definidas em outros pares de canal.

Uma saída de segurança em um canal CLNTCONN pode definir o Identificador de usuário remoto na definição de canal para processamento por uma saída de SVRCONN parceiro ou para autorização OAM se a Saída de segurança SVRCONN não estiver definida e o campo MCAUSER de SVRCONN não estiver configurado.

Se nenhuma Saída de segurança CLNTCONN estiver definida, o Identificador de usuário remoto na definição de canal é configurado para um identificador de usuário a partir do ambiente do cliente (que pode estar em branco) pelo cliente MCA.

Uma troca de segurança entre as Saídas de segurança definidas em um par de canais CLNTCONN e SVRCONN é concluída com êxito quando a Saída de segurança SVRCONN retorna um ExitResponse de MQXCC_OK. Uma troca de segurança entre outros pares de canal é concluída com êxito quando a Saída de segurança que iniciou a troca retorna um ExitResponse de MQXCC_OK.

No entanto, o código de ExitResponse MQXCC_SEND_AND_REQUEST_SEC_MSG pode ser usado para forçar a continuação da troca de segurança: se um ExitResponse de MQXCC_SEND_AND_REQUEST_SEC_MSG é retornado por uma Saída de segurança CLNTCONN ou SVRCONN, em seguida, a saída de parceiro deve responder enviando uma mensagem de segurança (não MQXCC_OK ou uma resposta nula) ou o canal é encerrado. Para Saídas de segurança definidas em outros tipos de canal, um ExitResponse de MQXCC_OK retornado em resposta a um MQXCC_SEND_AND_REQUEST_SEC_MSG da Saída de segurança do parceiro resulta na continuação da troca de segurança como se uma resposta nula fosse retornada e não na finalização do canal.

Saída de segurança SSPI
WebSphere MQ para Windows fornece uma saída de segurança que fornece autenticação para canais do WebSphere MQ usando a Security Services Programming Interface (SSPI). O SSPI fornece os recursos de segurança integrados do Windows

Essa saída de segurança destina-se ao cliente WebSphere MQ e ao servidor WebSphere MQ

Os pacotes de segurança são carregados a partir de security.dll ou secur32.dll. Esses DLLs são fornecidos com o seu sistema operacional.

A autenticação unilateral é fornecida no Windows, usando serviços de autenticação NTLM. A autenticação bidirecional é fornecida no Windows 2000, usando serviços de autenticação Kerberos .

O programa de saída de segurança é fornecido no formato de origem e de objeto. É possível usar o código de objeto no estado em que se encontra ou usar o código-fonte como um ponto de início para criar seus próprios programas de saída de usuário. Para obter mais informações sobre como usar o objeto ou o código de origem da saída de segurança SSPI, consulte "Usando a saída de segurança SSPI em sistemas Windows" na página 171

Programas de saída de envio e de recebimento do canal
É possível usar as saídas de envio e de recebimento para executar tarefas como compactação de dados e descompactação. É possível especificar uma lista de programas de saída de envio e de recebimento a ser executada em sucessão.

Programas de saída de envio e de recebimento do canal são chamados nos seguintes locais no ciclo de processamento de um MCA:

- Os programas de saída de envio e de recebimento do canal são chamados para inicialização na iniciação do MCA e para finalização na rescisão do MCA.
- O programa de saída de envio é chamado em um ou outro final do canal dependendo do final no qual uma transmissão para uma transferência de mensagem é enviada, imediatamente antes de uma transmissão ser enviada através do link. A nota 4 explica por que as saídas estão disponíveis em ambas as direções, ainda que os canais de mensagens enviem mensagens apenas em uma direção.
- O programa de saída de recebimento é chamado em um ou outro final do canal dependendo do final no qual uma transmissão para uma transferência de mensagem é recebida, imediatamente após uma transmissão ter sido obtida a partir do link. A nota 4 explica por que as saídas estão disponíveis em ambas as direções, ainda que os canais de mensagens enviem mensagens apenas em uma direção.

Pode haver várias transmissões para uma transferência de mensagem e pode haver várias iterações dos programas de saída de envio e recebimento antes de uma mensagem atingir a saída de mensagem no final receptor.

Aos programas de saída de envio e de recebimento do canal é passado um buffer de agente que contém dados de transmissão como enviados ou recebidos do link de comunicações. Para os programas de saída de envio, os primeiros 8 bytes do buffer são reservadas para uso pelo MCA e não devem ser mudados. Se o programa retornar um buffer diferente, então estes primeiros 8 bytes devem existir no novo buffer. O formato dos dados apresentados ao programa de saída não está definido.

Um bom código de resposta deve ser retornado pelos programas de saída de envio e recebimento. Qualquer outra resposta faz um fim anormal do MCA.

Nota: Não emita uma chamada MQGET, MQPUT ou MQPUT1 no ponto de sincronização a partir de uma saída de envio ou de recebimento.

Nota:

1. As saídas de envio e recebimento geralmente trabalham em pares. Por exemplo, uma saída de envio pode compactar os dados e uma saída de recebimento descompacte-los ou uma saída de envio pode criptografar os dados e uma saída de recebimento decriptografá-los. Ao definir os canais apropriados, certifique-se de que os programas de saída compatíveis sejam denominados para ambas as extremidades do canal.
2. Se a compactação estiver ativada para o canal, serão transmitidos dados compactados às saídas.
3. Saídas de envio e recebimento do canal podem ser chamadas para segmentos de mensagem diferentes dos segmentos dos dados do aplicativo, por exemplo, mensagens de status. Eles não são chamados durante o diálogo de inicialização nem durante a fase de verificação de segurança.
4. Embora canais de mensagens enviem mensagens apenas em uma direção, os dados do canal de controle, como pulsações e fim de processamentos em lote, flui em ambas as direções, e essas saídas estarão disponíveis em ambas as direções, também. No entanto, alguns dos fluxos de dados de inicialização do canal inicial são isentos de processamento por qualquer uma das saídas.
5. Há circunstâncias em que as saídas de envio e de recebimento podem ser invocadas fora de sequência; por exemplo, se estiver sendo executada uma série de programas de saída ou estiver também estiver executando saídas de segurança. Em seguida, quando a saída de recebimento é chamada pela primeira vez para processar dados, ela pode receber dados que não passaram pela saída de envio correspondente. Se a saída de recebimento apenas executasse a operação, por exemplo descompactação, sem antes verificar se era necessária, os resultados seriam inesperados.

   É necessário codificar suas saídas de envio e de recebimento de tal forma que a saída de recebimento possa verificar se os dados que está recebendo foram processados pela saída de envio correspondente. A maneira recomendada de fazer isso é codificar seus programas de saída de modo que:

   - A saída de envio configura o valor do nono byte de dados para 0 e desloca todos os dados juntos em 1 byte antes de executar a operação. (Os primeiros 8 bytes são reservados para uso pelo MCA.)
   - Se a saída de recebimento receber dados que possuem um 0 no byte 9, ela saberá que os dados vêm da saída de envio. Ela remove o 0, executa a operação complementar, e desloca os dados resultantes de volta em 1 byte.
   - Se a saída de recebimento receber dados que tenham algo diferente de 0 no byte 9, ela assumirá que a saída de envio não foi executada e enviará os dados de volta para o responsável pela chamada inalterados.

   Ao usar saídas de segurança, se o canal for encerrado pela saída de segurança, é possível que uma saída de envio seja chamada sem a saída de recebimento correspondente. Uma maneira de evitar esse problema é codificar a saída de segurança para configurar uma sinalização em MQCD.SecurityUserData ou MQCD.SendUserData, por exemplo, quando a saída decidir encerrar o canal. Em seguida, a saída de envio precisará verificar esse campo e processar os dados apenas se a sinalização não estiver configurada. Essa verificação evita que a saída de envio atere os dados sem necessidade e, portanto, impede qualquer conversão de erros que possa ocorrer se a saída de segurança tiver recebido dados alterados.

Programas de saída de envio do canal - reservando espaço
É possível usar saídas de envio e de recebimento para transformar os dados antes da transmissão. Programas de saída de envio do canal podem incluir seus próprios dados sobre a transformação reservando espaço no buffer de transmissão.

Esses dados são processados pelo programa de saída de recebimento e, em seguida, removidos do buffer. Por exemplo, você pode desejar criptografar os dados e incluir uma chave de segurança para decriptografia.

Como reservar espaço e usá-lo

Quando o programa de saída de envio é chamado para inicialização, configure o campo *ExitSpace* de MQXCP para o número de bytes a ser reservado. Consulte MQCXP para obter detalhes. *ExitSpace* pode

ser configurado somente durante a inicialização, ou seja, quando *ExitReason* tiver o valor MQXR_INIT. Quando a saída de envio é chamada imediatamente antes da transmissão, com *ExitReason* configurado para MQXR_XMIT, *ExitSpace* bytes são reservados no buffer de transmissão. O *ExitSpace* não é suportado no z/OS

A saída de envio não usa todo o espaço reservado. Ela pode usar menos que *ExitSpace* bytes ou, se o buffer de transmissão não estiver cheio, a saída pode usar mais do que a quantia reservada. Ao configurar o valor de *ExitSpace*, deve-se deixar pelo menos 1 KB para dados de mensagem no buffer de transmissão. O desempenho do canal pode ser afetado se o espaço reservado for usado para grandes quantias de dados.

O que acontece na extremidade de recebimento do canal

Os programas de saída de recebimento do canal devem ser configurados para serem compatíveis com as saídas de envio correspondentes. As saídas de recebimento devem saber o número de bytes no espaço reservado e devem remover os dados nesse espaço.

Várias saídas de envio

É possível especificar uma lista de programas de saída de envio e de recebimento a ser executada em sucessão. WebSphere MQ mantém um total do espaço reservado por todas as saídas de envio. Esse espaço total deve deixar pelo menos 1 KB para dados de mensagem no buffer de transmissão.

O exemplo a seguir mostra como o espaço é alocado para três saídas de envio, chamadas em sucessão:

1. Quando chamadas para inicialização:
   - A saída de envio A reserva 1 KB.
   - A saída de envio B reserva 2 KB.
   - A saída de envio C reserva 3 KB.
2. O tamanho de transmissão máximo é 32 KB e os dados do usuário têm 5 KB de comprimento.
3. A saída A é chamada com 5 KB de dados; até 27 KB estão disponíveis, pois 5 KB são reservados para as saídas B e C. A saída A soma 1 KB, a quantia que reservou.
4. A saída B é chamada com 6 KB de dados; até 29 KB estão disponíveis, pois 3 KB estão reservados para a saída C. A saída B soma 1 KB, menos do que os 2 KB que reservou.
5. A saída C é chamada com 7 KB de dados; até 32 KB estão disponíveis. A saída C inclui 10K, mais do que os 3 KB que reservou. Essa quantia é válida, porque a quantia total de dados, 17 KB, é menor que o máximo de 32 KB.

Programas de saída de mensagem do canal
É possível usar a saída de mensagem do canal para executar tarefas, como criptografia no link, validação ou substituição de IDs de usuário recebidos, conversão de dados de mensagem, utilização de diário e manipulação de mensagem de referência. É possível especificar uma lista de programas de saída de mensagem a serem executados em sucessão.

Programas de saída de mensagem do canal são chamados nos seguintes locais no ciclo de processamento do MCA:

- Na inicialização e na finalização do MCA
- Imediatamente após um MCA de envio ter emitido uma chamada MQGET
- Antes que o MCA de recebimento emita uma chamada MQPUT

É passado um buffer de agente à saída de mensagem que contém o cabeçalho da fila de transmissão, MQXQH, e o texto da mensagem do aplicativo, conforme recuperado da fila. (O formato de MQXQH é fornecido em MQXQH.) Se você usar mensagens de referência, ou seja, as mensagens que contêm somente um cabeçalho que aponte para algum outro objeto que deve ser enviado, a saída de mensagem reconhece o cabeçalho MQRMH. Ela identifica o objeto, recupera-o na maneira apropriada, anexa-o

ao cabeçalho e passa-o ao MCA para transmissão ao MCA de recebimento. No MCA de recebimento, outra saída de mensagem reconhece que essa mensagem é uma mensagem de referência, extrai o objeto e passo o cabeçalho para a fila de destino. Consulte “Mensagens de referência” na página 269 e “Executando as amostras Reference Message” na página 143 para obter mais informações sobre mensagens de referência e algumas saídas de mensagem de amostra que as manipulam.

As saídas de mensagem podem retornar as respostas a seguir:

- Envie a mensagem (saída GET). A mensagem pode ter sido mudada pela saída. (Isso retorna MQXCC_OK.)
- Coloque a mensagem na fila (saída PUT). A mensagem pode ter sido mudada pela saída. (Isso retorna MQXCC_OK.)
- Não processe a mensagem. A mensagem é colocada na fila de mensagens não entregues pelo MCA.
- Feche o canal.
- Código de retorno inválido, o que faz com que o MCA seja finalizado de forma anormal.

Nota:

1. As saídas de mensagem são chamadas uma vez para cada mensagem completa transferida, mesmo quando a mensagem for dividida em partes.
2. Em sistemas UNIX , se você fornecer uma saída de mensagem por qualquer motivo, a conversão automática de IDs do usuário para caracteres minúsculos não opera. Consulte Segurança de objetos em sistemas UNIX and Linux.
3. Uma saída é executada no mesmo encadeamento que o próprio MCA. Ela também é executada na mesma unidade de trabalho (UOW) que o MCA, pois usa a mesma manipulação de conexões. Portanto, quaisquer chamadas feitas sob o ponto de sincronização são confirmadas ou restauradas pelo canal no fim do lote. Por exemplo, um programa de saída de mensagem do canal pode enviar mensagens de notificação para outro e essas mensagens são confirmadas na fila somente quando o lote que contém a mensagem original for confirmado.

   Portanto, é possível emitir as chamadas MQI do ponto de sincronização a partir de um programa de saída de mensagem do canal.

Conversão de mensagens fora da saída de mensagem
Antes de chamar a saída de mensagem, o MCA de recebimento executa algumas conversões na mensagem. Este tópico descreve os algoritmos usados para executar as conversões.

Quais cabeçalhos são processados

Uma rotina de conversão é executada no MCA do receptor antes que a saída de mensagem seja chamada. A rotina de conversão começa com o cabeçalho MQXQH no início da mensagem. A rotina de conversão então processa os cabeçalhos encadeados que seguem MQXQH, executando a conversão conforme necessário. Os cabeçalhos encadeados podem se estender além do deslocamento contido no parâmetro HeaderLength dos dados de MQCXP que são passados à saída de mensagem do receptor. Os cabeçalhos a seguir são convertidos no local:

- MQXQH (nome do formato "`MQXMIT `")
- MQMD (esse cabeçalho faz parte do MQXQH e não tem nenhum nome de formato)
- MQMDE (nome do formato "`MQHMDE `")
- MQDH (nome do formato "`MQHDIST `")
- MQWIH (nome do formato "`MQHWIH `")

Os cabeçalhos a seguir não são convertidos, mas são passados à medida que o MCA continua a processar os cabeçalhos encadeados:

- MQDLH (nome do formato "`MQDEAD `")
- quaisquer cabeçalhos com nomes de formato que começam com os três caracteres 'MQH' (por exemplo, "`MQHRF `") que não são mencionados de outra forma

Como os cabeçalhos são processados

O parâmetro Format de cada cabeçalho WebSphere MQ é lido pelo MCA. O parâmetro Format é 8 bytes dentro do cabeçalho, que são 8 caracteres de byte único que contém um nome.

O MCA então interpreta os dados após cada cabeçalho como sendo do tipo denominado. Se o Formato for o nome de um tipo de cabeçalho elegível para conversão de dados WebSphere MQ , ele será convertido. Se for outro nome indicando dados não MQ (por exemplo, MQFMT_NONE ou MQFMT_STRING), o MCA para o processamento dos cabeçalhos.

Qual é o HeaderLength de MQCXP?

O parâmetro HeaderLength nos dados de MQCXP fornecido para uma saída de mensagem é o comprimento total dos cabeçalhos MQXQH (que inclui o MQMD), MQMDE e MQDH no início da mensagem. Esses cabeçalhos são encadeados usando os nomes e comprimentos de 'Format'.

MQWIH

Cabeçalhos encadeados podem se estender além do HeaderLength até a área de dados do usuário. O cabeçalho MQWIH, se estiver presente, é um desses cabeçalhos que aparece além do HeaderLength.

Se houver um cabeçalho MQWIH nos cabeçalhos encadeados, ele será convertido no local antes da saída de mensagem do receptor ser chamada.

Programa de saída de nova tentativa de mensagem do canal
A saída de nova tentativa de mensagem do canal é chamada quando uma tentativa de abrir a fila de destino não é bem-sucedida. É possível usar a saída para determinar sob quais circunstâncias tentar novamente, quantas vezes tentar novamente e com que frequência.

Essa saída também é chamada na extremidade de recebimento na inicialização e na finalização do MCA.

A saída de nova tentativa de mensagem do canal tem um buffer de agente passado que contém o cabeçalho da fila de transmissão, MQXQH, e o texto da mensagem do aplicativo conforme recuperado da fila. O formato de MQXQH é fornecido em Visão geral de MQXQH.

A saída é chamada para todos os códigos de razão; a saída determina para quais códigos de razão deseja que o MCA tente novamente, quantas vezes e em quais intervalos. (O valor do conjunto de contagem de novas tentativas de mensagem quando o canal foi definido é passado para a saída no MQCD, mas a saída pode ignorar esse valor.)

O campo MsgRetryCount em MQCXP é incrementado pelo MCA cada vez que a saída é chamada e a saída retorna MQXCC_OK com o tempo de espera contido no campo MsgRetryInterval de MQCXP ou MQXCC_SUPPRESS_FUNCTION. Novas tentativas continuam indefinidamente até a saída retornar MQXCC_SUPPRESS_FUNCTION no campo ExitResponse de MQCXP. Consulte MQCXP para obter informações sobre a ação executada pelo MCA para esses códigos de conclusão.

Se todas as novas tentativas forem mal-sucedidas, a mensagem será gravada na fila de mensagens não entregues. Se não houver nenhuma fila de mensagens não entregues disponível, o canal para.

Se você não definir uma saída de nova tentativa de mensagem para um canal e ocorrer uma falha que é provavelmente temporária, por exemplo, MQRC_Q_FULL, o MCA usa a contagem de nova tentativa de mensagem e os intervalos de nova tentativa de mensagem configurados de quando o canal foi definido. Se a falha for de natureza mais permanente e você não tiver definido um programa de saída para manipulá-la, a mensagem será gravada na fila de mensagens não entregues.

Programa de saída de autodefinição do canal
A saída de definição automática do canal pode ser usada quando uma solicitação é recebida para iniciar um canal receptor ou de conexão do servidor, mas nenhuma definição para esse canal existe (não para WebSphere MQ para z/OS). Também pode ser chamada em todas as plataformas para canais emissores de cluster e receptores de cluster para permitir modificação de definição para uma instância do canal.

A saída de definição automática do canal pode ser chamada em todas as plataformas exceto z/OS quando uma solicitação é recebida para iniciar um canal receptor ou de conexão do servidor, mas não

existe nenhuma definição de canal. É possível usá-la para modificar a definição padrão fornecida para um receptor definido automaticamente ou canal de conexão do servidor, SYSTEM.AUTO.RECEIVER ou SYSTEM.AUTO.SVRCON. Consulte Preparando canais para obter uma descrição de como as definições de canal podem ser criadas automaticamente.

A saída de autodefinição de canal também pode ser chamada quando uma solicitação for recebida para iniciar um canal emissor de clusters. Ela pode ser chamada para canais emissores de clusters e receptores de clusters para permitir modificação da definição para essa instância do canal. Nesse caso, a saída também se aplica ao WebSphere MQ para o z/OS Um uso comum da saída de autodefinição de canal é mudar os nomes de saídas de mensagens (MSGEXIT, RCVEXIT, SCYEXIT e SENDEXIT), porque nomes de saídas têm formatos diferentes em plataformas diferentes. Se nenhuma saída de definição automática de canal for especificada, o comportamento padrão no z/OS será examinar um nome de saída distribuído do formato *[path]/libraryname(function)* e usar até oito caracteres de função, se presente ou nome da biblioteca. No z/OS, um programa de saída de autodefinição de canal deve alterar os campos endereçados por MsgExitPtr, MsgUserDataPtr, SendExitPtr, SendUserDataPtr, ReceiveExitPtr e ReceiveUserDataPtr, em vez de MsgExit, MsgUserData, SendExit, Os próprios campos de Dados SendUser, ReceiveExit e ReceiveUser.

Para obter mais informações, consulte Definição automática de canais.

Como com outras saídas de canal, a lista de parâmetros é:

```
MQ_CHANNEL_AUTO_DEF_EXIT (ChannelExitParms, ChannelDefinition)
```

`ChannelExitParms` estão descritos em MQCXP. `ChannelDefinition` está descrito em MQCD.

MQCD contém os valores que são usados na definição de canal padrão, se eles não forem alterados pela saída. A saída pode modificar apenas um subconjunto dos campos; consulte MQ_CHANNEL_AUTO_DEF_EXIT. No entanto, a tentativa de mudar outros campos não causa um erro.

A saída de autodefinição de canal retorna uma resposta de MQXCC_OK ou MQXCC_SUPPRESS_FUNCTION. Se nenhuma dessas respostas for retornada, o MCA continua o processamento como se MQXCC_SUPPRESS_FUNCTION tivesse sido retornada. Ou seja, a autodefinição é abandonada, nenhuma nova definição de canal é criada e o canal não pode ser iniciado.

Compilando programas de saída do canal no Windows, sistemas UNIX and Linux

Use os exemplos a seguir para ajudar a compilar programas de saída do canal para sistemas Windows, UNIX and Linux .

Windows

Windows

O comando do compilador e do vinculador para programas de saída de canal no Windows:

```
cl.exe /Ic:\mqm\tools\c\include /nologo /c myexit.c
link.exe /nologo /dll myexit.obj /def:myexit.def /out:myexit.dll
```

sistemas UNIX e Linux

Linux UNIX

Nesses exemplos, `exit` é o nome da biblioteca e `ChannelExit` é o nome da função. No AIX o arquivo de exportação é chamado `exit.exp`. Esses nomes são usados pela definição de canal para fazer referência ao programa de saída usando o formato descrito em Definição de canal MQCD. Consulte também o parâmetro MSGEXIT do comando DEFINE CHANNEL.

Comandos do compilador e do vinculador de amostra para saídas do canal no AIX:

```
$ xlc_r -q64 -e MQStart -bE:exit.exp -bM:SRE -o /var/mqm/exits64/exit
exit.c -I/usr/mqm/inc
```

Comandos do compilador e do vinculador de amostra para saídas do canal no HP-UX

```
$ c89 +DD64 +z -c -D_HPUX_SOURCE -o exit.o exit.c -I/opt/mqm/inc
$ ld -b exit.o +ee MQStart +ee ChannelExit -o
/var/mqm/exits64/exit -L/usr/lib/pa20_64 -lpthread
$ rm exit.o
```

Comandos de amostra do compilador e do vinculador para saídas de canal em plataformas Linux, em que o gerenciador de filas é de 32 bits:

```
$ gcc -shared -fPIC -o /var/mqm/exits/exit exit.c -I/opt/mqm/inc
```

Comandos de amostra do compilador e do vinculador para saídas do canal em plataformas Linux, em que o gerenciador de filas é de 64 bits:

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
```

Comandos do compilador e do vinculador de amostra para saídas do canal no Solaris:

```
$ cc -xarch=v9 -mt -G -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
-R/usr/lib/64 -lsocket -lnsl -ldl
```

No cliente, uma saída de 32 bits ou 64 bits pode ser usada. Essa saída deve ser vinculada a mqic_r.

No AIX, todas as funções que são chamadas por IBM WebSphere MQ devem ser exportados Um arquivo de exportação de amostra para este make file:

```
#!
channelExit
MQStart
```

Configurando saídas do canal

Para chamar a saída do canal, deve-se denominá-la na definição de canal.

Saídas do canal devem ser denominadas na definição de canal. É possível realizar essa denominação quando definir os canais pela primeira vez ou é possível incluir as informações posteriormente usando, por exemplo, o comando ALTER CHANNEL do MQSC. Também é possível fornecer nomes de saída do canal na estrutura de dados do canal MQCD. O formato do nome da saída depende da plataforma IBM WebSphere MQ ; consulte MQCD ou Comandos de Script (MQSC) para obter informações.

Se a definição de canal não contiver um nome de programa de saída do usuário, a saída do usuário não será chamada.

A saída de autodefinição de canal é uma propriedade do gerenciador de filas, não do canal individual. Para que essa saída seja chamada, ela deve ser denominada na definição do gerenciador de filas. Para alterar uma definição do gerenciador de filas, use o comando ALTER QMGR do MQSC.

Escrevendo saídas de conversão de dados

Esta coleção de tópicos contém informações sobre como escrever saídas de conversão de dados.

Nota: Não suportado no MQSeries para VSE/ESA..

Ao executar um MQPUT, seu aplicativo cria o descritor de mensagens (MQMD) da mensagem. Como o WebSphere MQ precisa ser capaz de entender o conteúdo do MQMD, independentemente da plataforma na qual ele é criado, ele é convertido automaticamente pelo sistema

Dados do aplicativo, no entanto, não são automaticamente convertidos. Se os dados de caracteres estiverem sendo trocados entre plataformas em que os campos *CodedCharSetId* e *Encoding* diferem, por exemplo, entre ASCII e EBCDIC, o aplicativo deve providenciar a conversão da mensagem. A conversão de dados do aplicativo pode ser executada pelo gerenciador de filas em si ou por um programa de saída do usuário, referido como uma *saída de conversão de dados*. O gerenciador de filas pode executar a conversão de dados em si, usando uma de suas rotinas de conversão integradas, se os dados

do aplicativo estiverem em um dos formatos integrados (como MQFMT_STRING). Este tópico contém informações sobre o recurso de saída de conversão de dados que o WebSphere MQ fornece quando os dados do aplicativo não estão em um formato integrado.

O controle pode ser passado para a saída de conversão de dados durante uma chamada MQGET. Isso evita converter em diferentes plataformas antes de atingir o destino final. No entanto, se o destino final for uma plataforma que não suporte a conversão de dados no MQGET, deve-se especificar CONVERT(YES) no canal emissor que envia os dados para seu destino final. Isso assegura que o WebSphere MQ converta os dados durante a transmissão Nesse caso, sua saída de conversão de dados deve residir no sistema em que o canal emissor está definido.

A chamada MQGET é emitida diretamente pelo aplicativo. Configure os campos *CodedCharSetId* e *Encoding* no MQMD para o conjunto de caracteres e codificação necessários. Se seu aplicativo usar o mesmo conjunto de caracteres e codificação que o gerenciador de filas, configure *CodedCharSetId* para MQCCSI_Q_MGR e *Encoding* para MQENC_NATIVE. Após a chamada MQGET ser concluída, esses campos têm os valores apropriados para os dados de mensagem retornados. Eles podem ser diferentes dos valores necessários, se a conversão não tiver sido bem-sucedida. Seu aplicativo deve reconfigurar esses campos para os valores necessários antes de cada chamada MQGET.

As condições necessárias para que a saída de conversão de dados seja chamada são definidas para a chamada MQGET em MQGET.

Para obter uma descrição dos parâmetros que são passados à saída de conversão de dados e observações de uso detalhadas, consulte Conversão de dados para a chamada MQ_DATA_CONV_EXIT e a estrutura MQDXP.

Programas que convertem dados do aplicativo entre diferentes codificações da máquina e CCSIDs devem estar em conformidade com a interface de conversão de dados (DCI) do WebSphere MQ .

Com a introdução de clientes Multicast, as saídas de API e as saídas de conversão de dados precisam ser capazes de executar no lado do cliente, pois algumas mensagens podem não passar pelo gerenciador de filas. As bibliotecas a seguir agora fazem parte dos pacotes do cliente, assim como dos pacotes do servidor:

Tabela 56. Bibliotecas que agora estão nos pacotes do cliente e do servidor

Sistema operacional	**Bibliotecas**
Windows	32 bits e 64 bits: mqm.dll e mqm.pdb
Linux & HP-UX	32 bits e 64 bits: libmqm.s e libmqm_r.so
AIX	32 bits e 64 bits: libmqm.a e libmqm_r.a
Solaris	32 bits e 64 bits: libmqm.so

Chamando a saída de conversão de dados

Uma saída de conversão de dados é uma saída escrita pelo usuário que recebe controle durante o processamento de uma chamada MQGET.

A saída é chamada se o seguinte for verdadeiro:

- A opção MQGMO_CONVERT for especificada na chamada MQGET.
- Alguns ou todos os dados da mensagem não estiverem no conjunto de caracteres ou na codificação solicitado.
- O campo *Format* na estrutura MQMD associada à mensagem não for MQFMT_NONE.
- O *BufferLength* especificado na chamada MQGET não for zero.
- O comprimento dos dados da mensagem não for zero.
- A mensagem contiver dados que tenham um formato definido pelo usuário. O formato definido pelo usuário pode ocupar a mensagem inteira ou ser precedido por um ou mais formatos integrados. Por exemplo, o formato definido pelo usuário pode ser precedido por um formato MQFMT_DEAD_LETTER_HEADER. A saída for chamada para converter somente o formato definido

pelo usuário; o gerenciador de filas converte quaisquer formatos integrados que precedam o formato definido pelo usuário.

Uma saída escrita pelo usuário também pode ser chamada para converter um formato incorporado, mas isso acontece somente se as rotinas de conversão integradas não puderem converter o formato integrado com sucesso.

Há algumas outras condições, descritas integralmente na observações de uso da chamada MQ_DATA_CONV_EXIT em MQ_DATA_CONV_EXIT.

Consulte MQGET para obter detalhes da chamada MQGET. Saídas de conversão de dados não podem usar chamadas MQI, além de MQXCNVC.

Uma nova cópia da saída é carregada quando um aplicativo tenta recuperar a primeira mensagem que usa esse *Format* desde que o aplicativo conectou ao gerenciador de filas. Uma nova cópia também pode ser carregada em outros momentos se o gerenciador de filas tiver descartado uma cópia carregada anteriormente.

A saída de conversão de dados é executada em um ambiente como do programa que emitiu a chamada MQGET. Assim como aplicativos de usuário, o programa pode ser um MCA (agente do canal de mensagens) que está enviando mensagens a um gerenciador de filas de destino que não suporta a conversão de mensagens. O ambiente inclui espaço de endereço e perfil do usuário, conforme aplicável. A saída não pode comprometer a integridade do gerenciador de filas, porque ela não é executada no ambiente do gerenciador de filas.

Gravando uma saída de conversão de dados para sistemas WebSphere MQ em UNIX and Linux

Informações sobre as etapas a serem consideradas ao gravar programas de saída de conversão de dados para sistemas WebSphere MQ no UNIX and Linux

Siga estas etapas:

1. Nomeie seu formato da mensagem. O nome deve se ajustar no campo *Format* do MQMD e estar em maiúsculas, por exemplo, MYFORMAT. O nome do *Format* não deve ter espaços em branco à esquerda. Espaços em branco à direita são ignorados. O nome do objeto deve ter no máximo oito caracteres que não sejam espaços em branco, pois o *Format* tem somente oito caracteres de comprimento. Lembre-se de usar esse nome toda vez que enviar uma mensagem.

   Se a saída de conversão de dados for usada em um ambiente encadeado, o objeto carregável deve ser seguido por _r para indicar que é uma versão encadeada.
2. Crie uma estrutura para representar a sua mensagem. Consulte Sintaxe válida para obter um exemplo.
3. Execute essa estrutura por meio do comando `crtmqcvx` para criar um fragmento de código para sua saída de conversão de dados.

   As funções geradas pelo comando `crtmqcvx` usam macros que assumem que todas as estruturas são compactadas; emende-as se este não for o caso.
4. Copie o arquivo de origem da estrutura básica fornecido, renomeando-o para o nome de seu formato de mensagem configurado na etapa “1” na página 423. O arquivo de origem da estrutura básica e a cópia são somente leitura.

   O arquivo de origem da estrutura básica é chamado amqsvfc0.c.
5. No WebSphere MQ para AIX, um arquivo de exportação de esqueleto chamado amqsvfc.exp também é fornecido Copie esse arquivo, renomeando-o para MYFORMAT.EXP.
6. A estrutura básica inclui um arquivo de cabeçalho de amostra, amqsvmha.h, no diretório *MQ_INSTALLATION_PATH*/inc, em que *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado. Certifique-se de que seu caminho de inclusão aponte para esse diretório para captar esse arquivo.

   O arquivo amqsvmha.h contém macros que são usadas pelo código gerado pelo comando `crtmqcvx`. Se a estrutura a ser convertida contiver dados de caractere, essas macros chamarão MQXCNVC.
7. Localize as caixas de comentário a seguir no arquivo de origem e insira código conforme descrito:

a. Perto do fim do arquivo de origem, uma caixa de comentário inicia com:

```
/* Insert the functions produced by the data-conversion exit */
```

Aqui, insira o fragmento de código gerado na etapa “3” na página 423.

b. Perto do meio do arquivo de origem, uma caixa de comentário inicia com:

```
/* Insert calls to the code fragments to convert the format's */
```

Isso é seguido por uma chamada comentada para a função `ConverttagSTRUCT`.

Mude o nome da função para o nome da função incluída na etapa “7.a” na página 424. Remova os caracteres de comentário para ativar a função. Se houver várias funções, crie as chamadas para cada uma delas.

c. Perto do início do arquivo de origem, uma caixa de comentário inicia com:

```
/* Insert the function prototypes for the functions produced by */
```

Aqui, insira as instruções do protótipo de função para as funções incluídas na etapa “3” na página 423 acima...

8. Compile a sua saída como uma biblioteca compartilhada, usando MQStart como o ponto de entrada. Para isso, consulte “Compilação das saídas de conversão de dados em sistemas UNIX and Linux” na página 424.
9. Coloque a saída no diretório de saída. O diretório de saída padrão é `/var/mqm/exits` para sistemas de 32 bits e `/var/mqm/exits64`, para sistemas de 64 bits. É possível mudar esses diretórios no arquivo qm.ini ou mqclient.ini. Esse caminho pode ser configurado para cada gerenciador de filas e a saída será procurada somente nesse caminho ou caminhos.

Nota:

1. Se o `crtmqcvx` usar estruturas empacotados, todos os aplicativos WebSphere MQ deverão ser compilados dessa maneira
2. Os programas de saída de conversão de dados devem ser reentrantes.
3. MQXCNVC é a *única* chamada MQI que pode ser emitida a partir de uma saída de conversão de dados.

Compilação das saídas de conversão de dados em sistemas UNIX and Linux
Exemplos de como compilar uma saída de conversão de dados em sistemas UNIX and Linux.

Em todas as plataformas, o ponto de entrada para o módulo é MQStart.

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

AIX

Compile o código-fonte de saída emitindo um dos seguintes comandos:

Aplicativos de 32 bits

Não encadeado

```
cc -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Encadeado

```
xlc_r -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT_r \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Aplicativos de 64 bits

Não encadeado

```
cc -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Encadeado

```
xlc_r -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT_r \
    MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

HP-UX Plataforma Itanium

Compile e vincule o código-fonte de saída emitindo um dos seguintes conjuntos de comandos:

Aplicativos de 32 bits

Não encadeado

Compile o código-fonte de saída:

```
c89 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Vincule o objeto de saída:

```
ld +b: -b MYFORMAT.o +ee MQStart -o  \
    /var/mqm/exits/MYFORMAT -L/usr/lib/hpux32
rm MYFORMAT.o
```

Encadeado

Compile o código-fonte de saída:

```
c89 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Vincule o objeto de saída:

```
ld +b: -b MYFORMAT.o +ee MQStart -o  \
    /var/mqm/exits/MYFORMAT_r -L/usr/lib/hpux32 \
    -lpthread
rm MYFORMAT.o
```

Aplicativos de 64 bits

Não encadeado

Compile o código-fonte de saída:

```
 c89 +DD64 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Vincule o objeto de saída:

```
ld -b MYFORMAT.o +ee MQStart \
    -o  /var/mqm/exits64/MYFORMAT \
    -L/usr/lib/hpux64
rm MYFORMAT.o
```

Encadeado

Compile o código-fonte de saída:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -IMQ_INSTALLATION_PATH/inc
```

Vincule o objeto de saída:

```
ld -b MYFORMAT.o +ee MQStart  \
    -o  /var/mqm/exits64/MYFORMAT_r  \
    -L/usr/lib/hpux64 -lpthread
rm MYFORMAT.o
```

Linux

Compile o código-fonte de saída emitindo um dos seguintes comandos:

Aplicativos de 31 bits

Não encadeado

```
 gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c \
      -IMQ_INSTALLATION_PATH/inc
```

Encadeado

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

Aplicativos de 32 bits

Não encadeado

```
 gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c
      -IMQ_INSTALLATION_PATH/inc
```

Encadeado

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -IMQ_INSTALLATION_PATH/inc
```

Aplicativos de 64 bits

Não encadeado

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT MYFORMAT.c
      -IMQ_INSTALLATION_PATH/inc
```

Encadeado

```
 gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT_r MYFORMAT.c
      -IMQ_INSTALLATION_PATH/inc
```

Solaris

Compile o código-fonte de saída emitindo um dos seguintes comandos:

Aplicativos de 32 bits

Plataforma SPARC

```
cc -xarch=v8plus -KPIC -mt -G -o /var/mqm/exits/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/32 -lsocket -lnsl -ldl
```

Plataforma x86-64

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/32 -lsocket -lnsl -ldl
```

Aplicativos de 64 bits

Plataforma SPARC

```
cc -xarch=v9 -KPIC -mt -G -o /var/mqm/exits64/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/64 -lsocket -lnsl -ldl
```

Plataforma x86-64

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/MYFORMAT \
   MYFORMAT.c -IMQ_INSTALLATION_PATH/inc -R/usr/lib/64  -lsocket -lnsl -ldl
```

Gravando uma saída de conversão de dados para o WebSphere MQ para Windows

Informações sobre etapas a serem consideradas ao gravar programas de saída de conversão de dados para WebSphere MQ para Windows.

Siga estas etapas:

1. Nomeie seu formato da mensagem. O nome deve se ajustar no campo *Format* do MQMD. O nome do *Format* não deve ter espaços em branco à esquerda. Espaços em branco à direita são ignorados. O nome do objeto deve ter no máximo oito caracteres que não sejam espaços em branco, pois o *Format* tem somente oito caracteres de comprimento.

   Um arquivo .DEF chamado amqsvfcn.def também é fornecido no diretório de amostras, *MQ_INSTALLATION_PATH*\Tools\C\Samples. *MQ_INSTALLATION_PATH* é o diretório onde o WebSphere MQ está instalado. Tire uma cópia desse arquivo e o renomeie, por exemplo, para MYFORMAT.DEF. Certifique-se de que o nome do DLL que está sendo criado e o nome especificado em MYFORMAT.DEF sejam iguais. Sobrescreva o nome FORMAT1 em MYFORMAT.DEF com o novo nome do formato.

   Lembre-se de usar esse nome toda vez que enviar uma mensagem.
2. Crie uma estrutura para representar a sua mensagem. Consulte Sintaxe válida para obter um exemplo.
3. Execute essa estrutura por meio do comando `crtmqcvx` para criar um fragmento de código para sua saída de conversão de dados.

   As funções geradas pelo comando CRTMQCVX usam macros que são gravadas presumindo que todas as estruturas são compactadas; conserte-as se este não for o caso.
4. Copie o arquivo de origem de estrutura básica fornecido, amqsvfc0.c, renomeando-o para o nome do formato de mensagem que você configurou na etapa “1” na página 427.

   amqsvfc0.c está em *MQ_INSTALLATION_PATH*\Tools\C\Samples em que *MQ_INSTALLATION_PATH* é o diretório onde o WebSphere MQ está instalado. (O diretório de instalação padrão é `C:\Program Files\IBM\WebSphere MQ`.)

   O esqueleto inclui um arquivo de cabeçalho de amostra amqsvmha.h no diretório *MQ_INSTALLATION_PATH*\Tools\C\include Certifique-se de que seu caminho de inclusão aponte para esse diretório para captar esse arquivo.

   O arquivo amqsvmha.h contém macros que são usadas pelo código gerado pelo comando CRTMQCVX. Se a estrutura a ser convertida contiver dados de caractere, essas macros chamarão MQXCNVC.
5. Localize as caixas de comentário a seguir no arquivo de origem e insira código conforme descrito:

   a. Perto do fim do arquivo de origem, uma caixa de comentário inicia com:

   ```
   /* Insert the functions produced by the data-conversion exit */
   ```

   Aqui, insira o fragmento de código gerado na etapa “3” na página 427.

   b. Perto do meio do arquivo de origem, uma caixa de comentário inicia com:

   ```
   /* Insert calls to the code fragments to convert the format's */
   ```

   Isso é seguido por uma chamada comentada para a função `ConverttagSTRUCT`.

Mude o nome da função para o nome da função incluída na etapa “5.a” na página 427. Remova os caracteres de comentário para ativar a função. Se houver várias funções, crie as chamadas para cada uma delas.

c. Perto do início do arquivo de origem, uma caixa de comentário inicia com:

```
/* Insert the function prototypes for the functions produced by */
```

Aqui, insira as instruções de protótipo de função para as funções incluídas na etapa “3” na página 427.

6. Crie o arquivo de comando a seguir:

```
cl -I MQ_INSTALLATION_PATH\Tools\C\Include -Tp   \
MYFORMAT.C

MYFORMAT.DEF
```

em que *MQ_INSTALLATION_PATH* é o diretório no qual o WebSphere MQ está instalado

7. Emita o arquivo de comando para compilar sua saída como um arquivo DLL.
8. Coloque a saída no subdiretório de saída abaixo do diretório de dados WebSphere MQ . O diretório padrão para instalar suas saídas em sistemas de 32 bits é *MQ_DATA_PATH*\Exits e para sistemas de 64 bits é *MQ_DATA_PATH*\Exits64

O caminho usado para procurar pelas saídas de conversão de dados é fornecido no registro. A pasta de registro é:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\ClientExitPa
th\
```

e a chave de registro é: ExitsDefaultPath. Esse caminho pode ser configurado para cada gerenciador de filas e a saída será procurada somente nesse caminho ou caminhos.

Nota:

1. Se CRTMQCVX usar estruturas empacotadas, todos os aplicativos WebSphere MQ deverão ser compilados dessa maneira.
2. Os programas de saída de conversão de dados devem ser reentrantes.
3. MQXCNVC é a *única* chamada MQI que pode ser emitida a partir de uma saída de conversão de dados.

Sair e alternar arquivos de carregamento nos sistemas operacionais Windows

Os processos do gerenciador de filas IBM WebSphere MQ for Windows Version 7.5 têm 32 bits. Como resultado, quando usar aplicativos 64-bit, alguns tipos de saída e arquivos de carregamento do comutador XA também precisa ter uma versão 32-bit disponível para uso pelo gerenciador de filas. Se a versão 32-bit da saída ou arquivo de carregamento do comutador XA for necessário e não estiver disponível, então o comando falhar ou chamada de API relevante.

Dois atributos são suportados no qm.ini file para *ExitPath*. Eles são ExitsDefaultPath=*MQ_INSTALLATION_PATH*\exits e ExitsDefaultPath64=*MQ_INSTALLATION_PATH*\exits64, *MQ_INSTALLATION_PATH* Representa o diretório de alto nível no qual o WebSphere MQ está instalado Usá-los assegura que a biblioteca apropriada pode ser localizada. Se uma saída for usada em um cluster do WebSphere MQ , isso também garantirá que a biblioteca apropriada em um sistema remoto possa ser localizada

A tabela a seguir lista os diferentes tipos de Saída e Comutador carregar arquivos e notas se 32-bit ou 64-bit versões, ou ambos, são necessários, de acordo com se 32-bit ou 64-bit aplicativos estão sendo utilizados:

Tipos de arquivos	Aplicativos de 32 bits	Aplicativos de 64 bits
saída de cruzamento de API	32 bits	32-bit e 64-bit

Tipos de arquivos	Aplicativos de 32 bits	Aplicativos de 64 bits
Saída de conversão de dados	32 bits	64 bits
Saídas de canal do servidor (todos os tipos)	32 bits	32 bits
Saídas do Canal do Cliente (todos os tipos)	32 bits	64 bits
Saída de serviço instalável	32 bits	32 bits
Módulo de rastreio de serviço	32 bits	32-bit e 64-bit
Cluster do WLM de saída	32 bits	32 bits
Publicação/Assinatura de saída de roteamento	32 bits	32 bits
arquivos de carregamento do comutador do Banco	32 bits	32-bit e 64-bit
Bibliotecas AX do gerenciador de transações externo	32 bits	64 bits

Fazendo referência a definições de conexão usando uma saída de pré-conexão de um repositório

WebSphere MQ Os clientes MQI podem ser configurados para consultar um repositório para obter definições de conexão usando uma biblioteca de saída pré-conexão.

Apresentação

Um aplicativo cliente pode se conectar a um gerenciador de filas usando tabelas de definição de canal do cliente (CCDT). Geralmente, o arquivo CCDT está localizado em um servidor de arquivos de rede central e possui clientes que fazem referência a ele. Como é difícil gerenciar e administrar vários aplicativos clientes que fazem referência ao arquivo CCDT, uma abordagem flexível é armazenar as definições do cliente em um repositório global como um diretório LDAP, um Registro e Repositório do WebSphere ou qualquer outro repositório. Armazenar as definições de conexão do cliente em um repositório facilita o gerenciamento das definições de conexão do cliente e os aplicativos podem acessar as definições de conexão do cliente corretas e mais atuais.

Durante a execução da chamada MQCONN/X, o IBM WebSphere MQ MQI client carrega uma biblioteca de saída de pré-conexão especificada pelo aplicativo e chama uma função de saída para recuperar definições de conexão. As definições de conexão recuperadas são, então, usadas para estabelecer a conexão com um gerenciador de filas. Os detalhes da biblioteca de saída e da função a ser chamada são especificados no arquivo de configuração mqclient.ini.

Sintaxe

`void MQ_PRECONNECT_EXIT` (*pExitParms*, *pQMgrName*, *ppConnectOpts*, *pCompCode*, *pReason*);

Parâmetros

pExitParms

Tipo: entrada/saída PMQNXP

A estrutura do parâmetro de saída **`PreConnection`**.

A estrutura é alocada e mantida pelo responsável pela chamada da saída.

pQMgrName

Tipo: entrada/saída PMQCHAR

Nome do gerenciador de filas.

Na entrada, este parâmetro é a sequência de filtros fornecida para a chamada de API do MQCONN por meio do parâmetro **QMgrName**. Esse campo pode ficar em branco, ser explícito ou conter determinados caracteres curingas. O campo é mudado pela saída. O parâmetro é NULL quando a saída é chamada com MQXR_TERM.

ppConnectOpts
Tipo: entrada/saída ppConnectOpts

Opções que controlam a ação de MQCONNX.

Esse é um ponteiro para uma estrutura de opções de conexão MQCNO que controla a ação da chamada de API MQCONN. O parâmetro é NULL quando a saída é chamada com MQXR_TERM. O cliente MQI sempre fornece uma estrutura MQCNO para a saída, mesmo que ela não tenha sido fornecida originalmente pelo aplicativo. Se um aplicativo fornecer uma estrutura MQCNO, o cliente fará uma duplicata para passá-la para a saída em que é modificado. O cliente retém a propriedade do MQCNO.

Um MQCD referenciado por meio do MQCNO tem precedência sobre qualquer definição de conexão fornecida por meio da matriz. O cliente usa a estrutura MQCNO para se conectar ao gerenciador de filas e os outros são ignorados.

pCompCode
Tipo: entrada/saída PMQLONG

Código de conclusão.

Ponteiro para um MQLONG que recebe o código de conclusão de saídas. Deve ser um dos valores a seguir:

- `MQCC_OK` - Conclusão bem-sucedida
- `MQCC_WARNING` - Aviso (conclusão parcial)
- `MQCC_FAILED` - Falha na chamada

pReason
Tipo: entrada/saída PMQLONG

Razão que qualifica pCompCode.

Ponteiro para um MQLONG que recebe o código de razão de saída. Se o código de conclusão for MQCC_OK, o único valor válido será:

- MQRC_NONE - (0, x'000') Nenhuma razão para o relatório.

Se o código de conclusão for MQCC_FAILED ou MQCC_WARNING, a função de saída poderá configurar o campo de código de razão como qualquer valor MQRC_* válido.

Chamada C

```
void MQ_PRECONNECT_EXIT (&ExitParms, &QMgrName, &pConnectOpts, &CompCode, &Reason);
```

Parameter

```
PMQNXP  pExitParms   /*PreConnect exit parameter structure*/
PMQCHAR pQMgrName    /*Name of the queue manager*/
PPMQCNO ppConnectOpts/*Options controlling the action of MQCONNX*/
PMQLONG pCompCode    /*Completion code*/
PMQLONG pReason      /*Reason qualifying pCompCode*/
```

Sub-rotina PreConnect do arquivo de configuração do cliente

Use a sub-rotina PreConnect para configurar a saída PreConnect no arquivo `mqclient.ini`.

Os atributos a seguir podem ser incluídos na sub-rotina PreConnect:

Data=*< URL>*
URL do repositório no qual as definições de conexão estão armazenadas. Por exemplo, ao usar um servidor LDAP:

Data = ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com

Function=*<myFunc>*
Nome do ponto de entrada funcional na biblioteca que contém o código de saída PreConnect.

A definição de função se adere ao protótipo de saída PreConnect MQ_PRECONNECT_EXIT.

O comprimento máximo desse campo é MQ_EXIT_NAME_LENGTH.

Module=*< amqldapi>*
O nome do módulo que contém o código de saída da API.

Se este campo contiver o nome do caminho completo do módulo, ele será usado como está.

Sequence=*< sequence_number>*
A sequência na qual esta saída é chamada com relação a outras saídas. Uma saída com um baixo número de sequência é chamada antes de uma saída com um número de sequência mais alto. Não há necessidade para que a numeração de sequência de saídas seja contínua; uma sequência 1, 2, 3 possui o mesmo resultado que uma sequência de 7, 42, 1096. Este atributo é um valor numérico não assinado.

Diversas sub-rotinas PreConnect podem ser definidas no arquivo `mqclient.ini`. A ordem de processamento de cada saída é determinada pelo atributo de Sequência da sub-rotina.

Escrevendo e compilando saídas de publicação

É possível configurar uma saída de publicação no gerenciador de filas para mudar o conteúdo de uma mensagem publicada antes que seja recebida pelos assinantes. Também é possível mudar o cabeçalho da mensagem ou não entregar a mensagem a uma assinatura.

Saídas de publicação não suportadas no z/OS.

É possível usar a saída de publicação para inspecionar e alterar mensagens entregues aos assinantes:

- Examine o conteúdo de uma mensagem publicada para cada assinante
- Modifique os conteúdos de uma mensagem publicada para cada assinante
- Altere a fila na qual uma mensagem é colocada
- Pare a entrega de uma mensagem para um assinante

Escrevendo uma saída de publicação

Use as etapas em "Gravando e compilando saídas e serviços instaláveis" na página 379, para ajudá-lo a escrever e compilar sua saída.

O provedor da saída de publicação define o que a saída faz. A saída, no entanto, deve estar em conformidade com as regras definidas em MQPSXP.

WebSphere MQ não fornece uma implementação do ponto de entrada MQ_PUBLISH_EXIT. Ele fornece uma declaração typedef em linguagem C. Use o typedef para declarar os parâmetros para uma saída escrita pelo usuário corretamente. O exemplo a seguir ilustra como usar a declaração typedef:

```
#include "cmqec.h"

MQ_PUBLISH_EXIT MyPublishExit;

void MQENTRY MyPublishExit( PMQPSXP pExitParms,
                            PMQPBC  pPubContext,
                            PMQSBC  pSubContext )
{
  /* C language statements to perform the function of the exit */
}
```

A saída de publicação é executada no processo do gerenciador de filas, como resultado das operações a seguir:

- Uma operação de Publicação em que uma mensagem é entregue a um ou mais assinantes
- Uma operação de Publicação em que uma ou mais mensagens retidas são entregues
- Uma operação de Solicitação de assinatura em que uma ou mais mensagens retidas são entregues

Se a saída de publicação for chamada para uma conexão, na primeira vez em que for chamado, um código *ExitReason* de MQXR_INIT será configurado. Antes que a conexão seja desconectada depois de usar uma saída de publicação, a saída será chamada com um código *ExitReason* de MQXR_TERM.

Se a saída de publicação for configurada, mas não puder ser carregada quando o gerenciador de filas for iniciado, as operações de publicar/assinar mensagens serão inibidas para o gerenciador de filas. Deve-se corrigir o problema ou reiniciar o gerenciador de filas antes que o sistema de mensagens de publicar/assinar seja reativado.

Cada conexão WebSphere MQ que requer a saída de publicação pode falhar ao carregar ou inicializar a saída. Se a saída falhar ao carregar ou inicializar, as operações de publicar/assinar que requerem a saída de publicação serão desativadas para essa conexão. As operações falham com o WebSphere MQ código de razão MQRC_PUBLISH_EXIT_ERROR..

O contexto no qual a saída de publicação é chamada é a conexão por um aplicativo ao gerenciador de filas. Uma área de dados do usuário é mantida pelo gerenciador de filas para cada conexão que está executando operações de publicação. A saída pode reter informações na área de dados do usuário para cada conexão.

Uma saída de publicação pode usar algumas chamadas MQI. Ela pode usar somente as chamadas MQI que manipulam as propriedades de mensagens. As chamadas são:

- MQBUFMH
- MQCRTMH
- MQDLTMH
- MQDLTMP
- MQMHBUF
- MQINQMP
- MQSETMP

Se a saída de publicação mudar o gerenciador de filas de destino ou o nome da fila, nenhuma nova verificação de autoridade será executada.

Compilando uma saída de publicação

A saída de publicação é uma biblioteca dinamicamente carregada; ela pode ser considerada como uma saída de canal. Para obter informações sobre a compilação das saídas, consulte “Gravando e compilando saídas e serviços instaláveis” na página 379.

Saída de publicação de amostra

O programa de saída de amostra é chamado amqspse0.c. Ele grava uma mensagem diferente em um arquivo de log dependendo de se a saída foi chamada para inicializar, publicar ou finalizar as operações. Também demonstra o uso do campo da área do usuário de saída para alocar e liberar armazenamento de forma apropriada.

Configurando saídas de publicação

Deve-se configurar certos atributos para configurar uma saída de publicação.

No Windows e no Linux , é possível usar o WebSphere MQ Explorer para definir os atributos. Os atributos são definidos na página de propriedades do gerenciador de filas, em Publicar/Assinar.

Para configurar a saída de publicação no arquivo qm.ini nos sistemas UNIX e Linux , crie uma sub-rotina chamada PublishSubscribe A sub-rotina PublishSubscribe tem os atributos a seguir:

PublishExitPath=[path]|*module_name*
Nome e caminho do módulo que contém o código de saída da publicação. O comprimento máximo desse campo é MQ_EXIT_NAME_LENGTH. O padrão é nenhuma saída da publicação.

PublishExitFunction=*function_name*
Nome do ponto de entrada da função no módulo que contém o código de saída de publicação. O comprimento máximo desse campo é MQ_EXIT_NAME_LENGTH.

PublishExitData=*string*
Se o gerenciador de filas estiver chamando uma saída de publicação, ele passará uma estrutura MQPSXP como entrada. Os dados especificados usando o atributo *PublishExitData* são fornecidos no campo *ExitData* da estrutura. A sequência pode ter até MQ_EXIT_DATA_LENGTH caracteres de comprimento. O padrão é de 32 caracteres em branco.

Gravando e Compilando Saídas de Carga de Trabalho do Cluster

Escreva um programa de saída de carga de trabalho do cluster para customizar o gerenciamento de carga de trabalho de clusters. Você pode levar em consideração o custo de usar um canal em diferentes horários do dia ou o conteúdo da mensagem ao rotear as mensagens. Esses são fatores que não são considerados pelo algoritmo de gerenciamento de carga de trabalho padrão.

Na maioria dos casos, o algoritmo de gerenciamento de carga de trabalho é suficiente para suas necessidades. No entanto, para que seja possível fornecer seu próprio programa de saída de usuário para customizar o gerenciamento de carga de trabalho, o WebSphere MQ inclui uma saída de usuário, a saída de carga de trabalho do cluster

Pode haver alguma informação específica sobre sua rede ou mensagens que poderia usar para influenciar o balanceamento de carga de trabalho. Você pode saber quais são os canais de alta capacidade ou as rotas de rede baratas ou pode desejar rotear as mensagens dependendo de seu conteúdo. Poderia decidir escrever um programa de saída de carga de trabalho de cluster ou usar um fornecido por um terceiro.

A saída de carga de trabalho do cluster é chamada ao acessar uma fila de clusters. Ela é chamada por MQOPEN, MQPUT1 e MQPUT.

O gerenciador de filas de destino selecionado no momento de MQOPEN será fixado se MQOO_BIND_ON_OPEN for especificado. Nesse caso, a saída será executada somente uma vez.

Se o gerenciador de filas de destino não for fixado no momento de MQOPEN, o gerenciador de filas de destino será escolhido no momento da chamada MQPUT. Se o gerenciador de filas de destino não estiver disponível ou falhar enquanto a mensagem ainda estiver na fila de transmissão, a saída será chamada novamente. Um novo gerenciador de filas de destino será selecionado. Se o canal de mensagens falhar enquanto a mensagem estiver sendo transferida e a mensagem for restaurada, um novo gerenciador de filas de destino será selecionado.

Em plataformas diferentes do z/OS, o gerenciador de filas carregará a nova carga de trabalho do cluster na próxima vez que o gerenciador de filas for iniciado.

Se a definição do gerenciador de filas não contiver um nome de programa de saída de carga de trabalho do cluster, a saída de carga de trabalho do cluster não será chamada.

Vários dados são passados para uma saída de carga de trabalho do cluster na estrutura do parâmetro de saída, MQWXP:

- A estrutura de definição de mensagem, MQMD.
- O parâmetro de comprimento da mensagem.
- Uma cópia da mensagem ou parte da mensagem.

Em plataformas nãoz/OS , se você usar CLWLMode=FAST, cada processo do sistema operacional carrega sua própria cópia da saída. Diferentes conexões no gerenciador de filas podem fazer com que diferentes cópias da saída sejam chamadas. Se a saída for executada no modo de segurança padrão, CLWLMode=SAFE, uma única cópia da saída será executada em seu próprio processo separado.

Escrevendo saídas de carga de trabalho do cluster

Para plataformas diferentes do z/OS, as saídas de carga de trabalho do cluster não devem usar as chamadas MQI Em outros aspectos, as regras para escrever e compilar programas de saída de carga de trabalho do cluster são semelhantes às regras que se aplicam aos programas de saída do canal. Siga as etapas em “Gravando e compilando saídas e serviços instaláveis” na página 379 e use o programa de amostra, “Saída de carga de trabalho do cluster de amostra” na página 434, para ajudar a escrever e compilar sua saída.

Para obter mais informações sobre saídas do canal, consulte “Gravando programas de saída do canal” na página 405.

Configurando saídas de carga de trabalho do cluster

Nomeie saídas de carga de trabalho do cluster na definição do gerenciador de filas especificando o atributo de saída de carga de trabalho do cluster no comando ALTER QMGR . Por exemplo:

```
ALTER QMGR CLWLEXIT(myexit)
```

Saída de carga de trabalho do cluster de amostra

O WebSphere MQ inclui um programa de saída de carga de trabalho do cluster de amostra É possível copiar a amostra e usá-la como base para seus próprios programas.

Em plataformas diferentes do z/OS

O programa de saída de carga de trabalho do cluster de amostra é fornecido em C e chamado amqswlm0.c. Ele pode ser localizado em:

Tabela 57. Local do programa de saída de carga de trabalho do cluster de amostra (não z/OS).

Plataforma	**Caminho de arquivo..**
AIX, HP-UX, Sun Solaris	*MQ_INSTALLATION_PATH*/samp
Windows	*MQ_INSTALLATION_PATH*\Tools\c\Samples

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

Esta saída da amostra roteia todas as mensagens para um gerenciador de filas específico, a menos que o gerenciador de filas se torne indisponível. Ela reage contra a falha do gerenciador de filas ao rotear mensagens para outro gerenciador de filas.

Indique qual gerenciador de filas ao qual você deseja que as mensagens sejam enviadas. Forneça o nome do canal do receptor de clusters no atributo CLWLDATA na definição do gerenciador de filas. Por exemplo:

```
ALTER QMGR CLWLDATA('my-cluster-name.my-queue-manager')
```

Para ativar a saída, forneça seu caminho completo e nome no atributo CLWLEXIT:

Nos sistemas UNIX and Linux:

```
ALTER QMGR CLWLEXIT('path/amqswlm(clwlFunction)')
```

No Windows:

```
ALTER QMGR CLWLEXIT('path\amqswlm(clwlFunction)')
```

Agora, em vez de usar o algoritmo de gerenciamento de carga de trabalho fornecido o WebSphere MQ chama essa saída para rotear todas as mensagens para o seu gerenciador de filas escolhido

Construindo um aplicativo IBM WebSphere MQ

Use estas informações para saber como construir um aplicativo IBM WebSphere MQ em diferentes plataformas.

Construindo seu aplicativo no AIX

As publicações do AIX descrevem como construir aplicativos executáveis a partir dos programas gravados.

Este tópico descreve as tarefas adicionais e as mudanças nas tarefas padrão que devem ser executadas ao construir aplicativos WebSphere MQ para AIX para execução em AIX. C, C++ e COBOL são suportados. Para obter informações sobre a preparação de seus programas C++, consulte Using C++.

As tarefas que você deve executar para criar um aplicativo executável usando WebSphere MQ para AIX variam com a linguagem de programação na qual seu código-fonte é gravado. Além de codificar as chamadas MQI em seu código-fonte, deve-se incluir as instruções de linguagem apropriadas para incluir os arquivos de inclusão do WebSphere MQ para AIX para a linguagem que você está usando Familiarize-se com o conteúdo desses arquivos. Consulte "IBM WebSphere MQ arquivos de definição de dados" na página 82 para obter uma descrição completa.

Ao executar aplicativos cliente ou servidor encadeados, configure a variável de ambiente AIXTHREAD_SCOPE=S

Preparando programas C no AIX

Este tópico contém informações sobre a vinculação de bibliotecas necessárias para preparar programas C no AIX

Programas C pré-compilados são fornecidos no diretório *MQ_INSTALLATION_PATH*/samp/bin. Use o compilador ANSI e execute os comandos a seguir. Para obter informações adicionais sobre a programação de aplicativos de 64 bits, consulte Padrões de codificação em plataformas de 64 bits.

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

Para aplicativos de 32 bits:

```
$ xlc_r -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm
```

em que `amqsput0` é um programa de amostra.

Para aplicativos de 64 bits:

```
$ xlc_r -q64 -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64
-lmqm
```

em que `amqsput0` é um programa de amostra.

Se você estiver usando o compilador C/C++ do VisualAge para programas C++ deverá incluir a opção `-q namemangling=v5` para obter todos os símbolos do WebSphere MQ resolvidos ao vincular as bibliotecas.

Se desejar usar os programas em uma máquina que tenha apenas o cliente MQI do WebSphere MQ para AIX instalado, recompile os programas para vinculá-los à biblioteca do cliente (`-lmqic`) em vez disso.

Vinculando bibliotecas

São necessárias as seguintes bibliotecas:

- Vincule seus programas à biblioteca apropriada fornecida pelo WebSphere MQ.

  Em um ambiente não encadeado, vincule-se a uma das seguintes bibliotecas:

Arquivo de bilioteca	Tipo de programa/saída
libmqm.a	Servidor para C
libmqic.a & libmqm.a	Cliente para C

Em um ambiente encadeado, vincule-se a uma das seguintes bibliotecas:

Arquivo de bilioteca	Tipo de programa/saída
libmqm_r.a	Servidor para C
libmqic_r.a & libmqm_r.a	Cliente para C

Por exemplo, para construir um aplicativo WebSphere MQ simples encadeado a partir de uma única unidade de compilação, execute os comandos a seguir.

Para aplicativos de 32 bits:

```
$ xlc_r -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm_r
```

em que `amqsput0` é um programa de amostra.

Para aplicativos de 64 bits:

```
$ xlc_r -q64 -o amqsputc_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64
-lmqm_r
```

em que `amqsput0` é um programa de amostra.

Se desejar usar os programas em uma máquina que tenha apenas o cliente MQI do WebSphere MQ para AIX instalado, recompile os programas para vinculá-los à biblioteca do cliente (`-lmqic`) em vez disso.

Nota:

1. Se estiver gravando um serviço instalável (consulte Administrando para obter informações adicionais), será necessário vincular à biblioteca `libmqmzf.a` em um aplicativo não encadeado e à biblioteca `libmqmzf_r.a` em um aplicativo encadeado.
2. Se você estiver produzindo um aplicativo para coordenação externa por um gerenciador de transações compatível com XA, como IBM TXSeries, Encinaou BEA Tuxedo, será necessário vincular-se ao `libmqmxa.a` (ou `libmqmxa64.a` se seu gerenciador de transações tratar o tipo 'long' como 64 bits) e bibliotecas `libmqz.a` em um aplicativo não encadeado e às bibliotecas `libmqmxa_r.a` (ou `libmqmxa64_r.a`) e `libmqz_r.a` em um aplicativo encadeado.
3. É necessário vincular aplicativos confiáveis às bibliotecas encadeadas do WebSphere MQ . No entanto, apenas um encadeamento em um aplicativo confiável em sistemas WebSphere MQ em UNIX and Linux pode ser conectado por vez
4. Você deve vincular bibliotecas do WebSphere MQ antes de quaisquer outras bibliotecas do produto.

Preparando programas COBOL em AIX

Use estas informações ao preparar programas COBOL no AIX usando o IBM COBOL Set e o Micro Focus COBOL.

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

- Copy books COBOL de 32 bits são instalados no seguinte diretório:

  ```
  MQ_INSTALLATION_PATH/inc/cobcpy32
  ```

  e links simbólicos são criados em:

```
MQ_INSTALLATION_PATH/inc
```

- Copy books COBOL de 64 bits são instalados no diretório a seguir:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

Nos exemplos a seguir configure a variável de ambiente **COBCPY** para:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

para aplicativos de 32 bits e:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

para aplicativos de 64 bits.

É necessário vincular seu programa a um dos arquivos de biblioteca a seguir:

Arquivo de bilioteca	Tipo de programa/saída
libmqmcb.a	Servidor para COBOL (aplicativo não encadeado)
libmqmcb_r.a	Servidor para COBOL (aplicativo encadeado)
libmqicb.a	Cliente para COBOL (aplicativo não encadeado)
libmqicb_r.a	Cliente para COBOL (aplicativo encadeado)

É possível usar o compilador IBM COBOL Set ou o compilador Micro Focus COBOL, dependendo do programa:

- Programas iniciados por amqm são adequados para o compilador Micro Focus COBOL e
- Programas iniciados por amq0 são adequados para ambos os compiladores.

Preparando programas COBOL usando IBM COBOL Set for AIX

Programas COBOL de amostra são fornecidos com o IBM WebSphere MQ. Para compilar esse programa, insira o comando apropriado na lista a seguir:

Aplicativo do servidor não encadeado de 32 bits

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb -qLIB \
-I<COBCPY>
```

Aplicativo cliente não encadeado de 32 bits

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb -qLIB \
-I<COBCPY>
```

Aplicativo do servidor encadeado de 32 bits

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -I<COBCPY>
```

Aplicativo cliente encadeado de 32 bits

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -I<COBCPY>
```

Aplicativo do servidor não encadeado de 64 bits

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqmcb \
-qLIB -I<COBCPY>
```

Aplicativo cliente não encadeado de 64 bits

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqicb \
-qLIB -I<COBCPY>
```

Aplicativo do servidor encadeado de 64 bits

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -I<COBCPY>
```

Aplicativo cliente encadeado de 64 bits

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -I<COBCPY>
```

Preparando programas COBOL usando o Micro Focus COBOL

Configure as variáveis de ambiente antes de compilar seu programa da seguinte forma:

```
export COBCPY=<COBCPY>
export LIBPATH=MQ_INSTALLATION_PATH/lib:$LIBPATH
```

Para compilar um programa COBOL de 32 bits usando o Micro Focus COBOL, insira:

- Servidor para COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb
```

- Cliente para COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb
```

- Servidor encadeado para COBOL

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r
```

- Cliente encadeado para COBOL

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r
```

Para compilar um programa COBOL de 64 bits usando o Micro Focus COBOL, insira:

- Servidor para COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb
```

- Cliente para COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb
```

- Servidor encadeado para COBOL

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r
```

- Cliente encadeado para COBOL

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r
```

em que amqminqx é um programa de amostra

Consulte a documentação do Micro Focus COBOL para obter uma descrição das variáveis de ambiente que precisa configurar.

Preparando programas de aplicativos CICS no AIX

Use estas informações ao preparar programas CICS em AIX.

Os módulos do comutador XA são fornecidos para permitir que você vincule CICS com IBM WebSphere MQ:

Tabela 58. Código essencial para programas de aplicativo do CICS no AIX: rotina de inicialização de XA

Descrição	**C (origem)**	**C (exec) - incluir em seu XAD.Stanza**
Rotina de inicialização de XA	amqzscix.c	*amqzsc* - CICS para AIX

Use a versão pré-construída do IBM WebSphere MQ switch load file *amqzsc* , que é fornecido com o produto.

Sempre vincule suas transações C à IBM WebSphere MQ biblioteca *libmqm_r.athread-safe.*, e suas transações COBOL com a biblioteca COBOL *libmqmcb_r.a.*

É possível localizar mais informações sobre o suporte a transações do CICS no Administrando.

Suporte TXSeries CICS

O IBM WebSphere MQ on AIX suporta TXSeries CICS usando a interface XA Assegure que os aplicativos CICS estejam vinculados à versão encadeada das bibliotecas IBM WebSphere MQ .

É possível executar programas CICS usando o IBM COBOL Set para o AIX ou Micro Focus COBOL. As seções ações descrevem as diferenças entre a execução de programas do CICS no IBM COBOL Set para o AIX e Micro Focus COBOL.

Grave WebSphere MQ programas que são carregados na mesma região do CICS em C ou COBOL. Não é possível fazer uma combinação de chamadas MQI C e COBOL na mesma região do CICS A maioria das chamadas MQI na segunda linguagem usada falha com um código de razão de MQRC_H0BJ_ERR0R.

Preparando programas COBOL do CICS usando IBM COBOL configurado para AIX

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

Para usar o IBM COBOL, siga estas etapas:

1. Exporte a variável de ambiente a seguir:

```
export LDFLAGS="-qLIB -bI:/usr/lpp/cics/lib/cicsprIBMCOB.exp  \
                -IMQ_INSTALLATION_PATH/inc -I/usr/lpp/cics/include  \
                -e _iwz_cobol_main  \
```

   em que LIB é uma diretriz do compilador.
2. Converta, compile e vincule o programa digitando:

```
cicstcl -l IBMCOB <yourprog>.ccp
```

Preparando programas CICS COBOL usando Micro Focus COBOL

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

Para usar Micro Focus COBOL, siga estas etapas:

1. Inclua o módulo de biblioteca de tempo de execução do IBM WebSphere MQ COBOL na biblioteca de tempo de execução usando o comando a seguir:

```
cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
            MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe_r
```

   Nota: Com `cicsmkcobol`, o IBM WebSphere MQ não permite fazer chamadas MQI na linguagem de programação C a partir de seu aplicativo em COBOL.

   Se seus aplicativos existentes tiverem quaisquer chamadas desse tipo, será recomendado que você mova estas funções dos aplicativos COBOL para sua própria biblioteca, por exemplo, `myMQ.so`. Depois de mover as funções, não inclua a biblioteca do IBM WebSphere MQ `libmqmcbrt.o` ao construir o aplicativo COBOL para CICS.

   Além disso, se seu aplicativo COBOL não fizer nenhuma chamada MQI COBOL, não vincule `libmqmz_r` com `cicsmkcobol`.

   Isso cria o arquivo de método de linguagem Micro Focus COBOL e ativa a biblioteca COBOL de tempo de execução do CICS para chamar IBM WebSphere MQ nos sistemas UNIX and Linux.

   Nota: Execute `cicsmkcobol` somente ao instalar um dos produtos a seguir:

   - Nova versão ou liberação do Micro Focus COBOL
   - Nova versão ou liberação do CICS para AIX
   - Nova versão ou liberação de qualquer produto de banco de dados suportado (apenas para transações COBOL)
   - Nova versão ou liberação do IBM WebSphere MQ .
2. Exporte a variável de ambiente a seguir:

```
COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
```

3. Converta, compile e vincule o programa digitando:

```
cicstcl -l COBOL -e <yourprog>.ccp
```

Preparando programas C do CICS

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

Construir programas CICS C usando os recursos padrão do CICS:

1. Exporte ***uma*** das variáveis de ambiente a seguir:
   - LDFLAGS = "-L/*MQ_INSTALLATION_PATH*lib -lmqm_r" exportar LDFLAGS
   - USERLIB = "-L*MQ_INSTALLATION_PATH*lib -lmqm_r" exportar USERLIB
2. Converta, compile e vincule o programa digitando:

```
cicstcl -l C amqscic0.ccs
```

Transação de amostra CICS C

A origem de C de amostra para uma transação AIX IBM WebSphere MQ é fornecida por AMQSCIC0.CCS. A transação lê mensagens da fila de transmissão SYSTEM.SAMPLE.CICS.WORKQUEUE no gerenciador de filas padrão e coloca-os na fila local com um nome de fila que está contido no cabeçalho de transmissão da mensagem Quaisquer falhas são enviadas à fila SYSTEM.SAMPLE.CICS.DLQ. Use o script de MQSC de amostra AMQSCIC0.TST para criar estas filas e filas de entrada de amostra.

Construindo seu aplicativo no HP Integrity NonStop Server

Estas informações descrevem as tarefas adicionais e as mudanças nas tarefas padrão, que deverão ser executadas ao construir o cliente IBM WebSphere MQ para os aplicativos HP Integrity NonStop Server para serem executados no HP Integrity NonStop Server.

C, COBOL e pTAL são suportados.

Cabeçalhos e Bibliotecas Públicas de OSS e Guardian

Fornece listas de cabeçalhos OSS e Guardian e bibliotecas públicas. São listados cabeçalhos OSS, bibliotecas de importações públicas e executáveis públicos do OSS, cabeçalhos Guardian, e bibliotecas de importações públicas e executáveis públicos do Guardian.

“Cabeçalhos de OSS” na página 441

“Executáveis Públicos de OSS e bibliotecas de importações públicas” na página 442

“cabeçalhos Guardian” na página 443

“Bibliotecas executáveis e de importação, públicas, do Guardian” na página 443

Cabeçalhos de OSS

Tabela 59. Cabeçalhos de OSS

Object	Local	Descrição
cmqbc.h	<mqinstall>/inc	IBM WebSphere MQ cabeçalho da linguagem C (OSS)
cmqc.h	<mqinstall>/inc	IBM WebSphere MQ cabeçalho da linguagem C (OSS)

Tabela 59. Cabeçalhos de OSS (continuação)		
Object	**Local**	**Descrição**
cmqfc.h	<mqinstall>/inc	IBM WebSphere MQ cabeçalho da linguagem C (OSS)
cmqec.h	<mqinstall>/inc	IBM WebSphere MQ cabeçalho da linguagem C (OSS)
cmqpsc.h	<mqinstall>/inc	IBM WebSphere MQ cabeçalho da linguagem C (OSS)
cmqxc.h	<mqinstall>/inc	IBM WebSphere MQ cabeçalho da linguagem C (OSS)
cmqzc.h	<mqinstall>/inc	IBM WebSphere MQ cabeçalho da linguagem C (OSS)
cmqcobol.cpy	<mqinstall>/inc	IBM WebSphere MQ copybook COBOL (OSS)
cmqbt.tal	<mqinstall>/inc	Cabeçalho do IBM WebSphere MQ pTAL (OSS)
cmqcft.tal	<mqinstall>/inc	Cabeçalho do IBM WebSphere MQ pTAL (OSS)
cmqpst.tal	<mqinstall>/inc	Cabeçalho do IBM WebSphere MQ pTAL (OSS)
cmqt.tal	<mqinstall>/inc	Cabeçalho do IBM WebSphere MQ pTAL (OSS)
cmqxt.tal	<mqinstall>/inc	Cabeçalho do IBM WebSphere MQ pTAL (OSS)

Executáveis Públicos de OSS e bibliotecas de importações públicas

Tabela 60. Executáveis Públicos de OSS e bibliotecas de importações públicas		
Object	**Local**	**Descrição**
libmqic.so	<mqinstall>/bin	Biblioteca executável pública do IBM WebSphere MQ (OSS não encadeado)
libmqic_r.so	<mqinstall>/bin	Biblioteca executável pública do IBM WebSphere MQ (OSS multiencadeado)
libmqic.so	<mqinstall>/lib	Biblioteca de importação pública IBM WebSphere MQ (OSS não encadeado)
libmqic_r.so	<mqinstall>/lib	Biblioteca de importação pública doIBM WebSphere MQ (OSS multiencadeado)
mqicb	<mqinstall>/lib	IBM WebSphere MQ biblioteca de importações públicas para COBOL (OSS)

cabeçalhos Guardian

Tabela 61. cabeçalhos Guardian

Object	Local	Descrição
cmqbch	<mqinstall>/inc/G	IBM WebSphere MQ cabeçalho linguagem C (Guardian)
cmqch	<mqinstall>/inc/G	IBM WebSphere MQ cabeçalho linguagem C (Guardian)
cmqfch	<mqinstall>/inc/G	IBM WebSphere MQ cabeçalho linguagem C (Guardian)
cmqech	<mqinstall>/inc/G	IBM WebSphere MQ cabeçalho linguagem C (Guardian)
cmqpsch	<mqinstall>/inc/G	IBM WebSphere MQ cabeçalho linguagem C (Guardian)
cmqxch	<mqinstall>/inc/G	IBM WebSphere MQ cabeçalho linguagem C (Guardian)
cmqzch	<mqinstall>/inc/G	IBM WebSphere MQ cabeçalho linguagem C (Guardian)
cmqcobol	<mqinstall>/inc/G	IBM WebSphere MQ copybook COBOL (Guardian)
cmqbt	<mqinstall>/inc/G	Cabeçalho do IBM WebSphere MQ pTAL (Guardian)
cmqcft	<mqinstall>/inc/G	Cabeçalho do IBM WebSphere MQ pTAL (Guardian)
cmqpst	<mqinstall>/inc/G	Cabeçalho do IBM WebSphere MQ pTAL (Guardian)
cmqt	<mqinstall>/inc/G	Cabeçalho do IBM WebSphere MQ pTAL (Guardian)
cmqxt	<mqinstall>/inc/G	Cabeçalho do IBM WebSphere MQ pTAL (Guardian)

Bibliotecas executáveis e de importação, públicas, do Guardian

Tabela 62. Bibliotecas executáveis e de importação, públicas, do Guardian

Object	Local	Descrição
mqic	<mqinstall>/bin/G	Biblioteca executável pública do IBM WebSphere MQ (Guardian)
mqicb	<mqinstall>/lib/G	Biblioteca de importação púbica do IBM WebSphere MQ para COBOL (Guardian)

Preparando programas C no HP Integrity NonStop Server

Este tópico contém informações a considerar quando você está preparando programas C no HP Integrity NonStop Server juntamente com exemplos dos comandos usados quando está construindo aplicativos quando está usando o compilador OSS C e quando está usando o compilador Guardian C.

Programas C pré-compilados são fornecidos no diretório MQ_INSTALLATION_PATH/opt/mqm/samp/bin. Para construir uma amostra a partir do código-fonte, use o compilador c89.

Deve-se vincular os seus programas com a biblioteca apropriada fornecida pelo IBM WebSphere MQ. A tabela a seguir lista as bibliotecas com as quais deve-se vincular quando estiver preparando programas C no HP Integrity NonStop Server.

Tabela 63. . Bibliotecas de links HP Integrity NonStop Server

Biblioteca	**Descrição**
libmqic.so	OSS não encadeado
libmqic_r.so	OSS multiencadeado
mqic	Guardian

Aplicativos IBM WebSphere MQ nativos multiencadeados devem usar o recurso Posix User Threads (PUT). Não há suporte para Standard Posix Threads (SPT) nesse produto.

Construindo aplicativos usando o compilador OSS C

Esta seção contém exemplos dos comandos usados para construir programas que são destinados ao OSS ou ao Guardian quando você estiver usando o compilador OSS.

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

O exemplo a seguir constrói um aplicativo OSS do cliente C não encadeado:

```
c89 -Wsystype=oss -o amqsputc amqsput0.c -IMQ_INSTALLATION_PATH/opt/mqm/inc
-LMQ_INSTALLATION_PATH/opt/mqm/lib -lmqic
```

O exemplo a seguir constrói um aplicativo OSS do cliente C multiencadeado:

```
c89 -Wsystype=oss -D_PUT_MODEL_ -o amqsputc amqsput0.c -IMQ_INSTALLATION_PATH/opt/mqm/inc
-LMQ_INSTALLATION_PATH/opt/mqm/lib -lmqic_r -lput
```

O exemplo a seguir constrói um aplicativo Guardian do cliente C:

```
c89 -Wsystype=guardian -o /G/vol/subvol/amqsputc amqsput0.c -IMQ_INSTALLATION_PATH/opt/mqm/inc
-LMQ_INSTALLATION_PATH/opt/mqm/lib/G -lmqic
```

Construindo aplicativos usando o compilador Guardian C

Esta seção contém exemplos dos comandos que são usados para construir programas destinados para o Guardian quando você estiver usando o compilador Guardian.

MQ_INSTALLATION_PATH representa o volume e o subvolume do Guardian nos quais o IBM WebSphere MQ está instalado.

O exemplo a seguir constrói um aplicativo Guardian do cliente C:

```
CCOMP /in AMQSPUT0/ AMQSPUTC;&
       runnable,systype guardian,nolist,&
       ssv0 "$system.system",&
       ssv1 "MQINSTALLATION_SUBVOL",&
       ld(-LMQINSTALLATION_SUBVOL -lmqic)
```

Preparando programas COBOL

Este tópico contém informações a considerar quando você está preparando programas C para o cliente de IBM WebSphere MQ para HP Integrity NonStop Server. Ele contém exemplos dos comandos que você usa quando está construindo aplicativos quando está usando o compilador OSS ECOBOL e quando está usando o compilador Guardian ECOBOL.

Para construir uma amostra de COBOL a partir do código de origem, use o compilador ECOBOL.

A tabela a seguir lista as bibliotecas que são necessárias quando você está preparando programas COBOL no HP Integrity NonStop Server. Deve-se vincular os seus programas com a biblioteca apropriada fornecida pelo IBM WebSphere MQ.

Tabela 64. . Bibliotecas de links HP Integrity NonStop Server	
Biblioteca	**Descrição**
libmqic.so	OSS não encadeado
mqic	Guardian

Ao executar um aplicativo COBOL que se conecta a um gerenciador de filas, deve-se primeiro configurar a variável *SAVE-ENVIRONMENT* para ON. Para configurar a variável *SAVE-ENVIRONMENT* para ON:

- Para OSS, insira o comando a seguir:

```
export SAVE-ENVIRONMENT=ON
```

- Para Guardian, insira o comando a seguir:

```
param SAVE-ENVIRONMENT ON
```

Se a variável *SAVE-ENVIRONMENT* não for configurada como ON, quando o aplicativo tentar se conectar a um gerenciador de filas, ele falhará com o código da razão 2058 (080A) (RC2058): MQRC_Q_MGR_NAME_ERROR.

Construindo aplicativos usando o compilador OSS ECOBOL

Esta seção contém exemplos dos comandos que são usados para construir programas que são destinados para o OSS ou Guardian quando você está usando o compilador OSS ECOBOL.

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

O exemplo a seguir constrói um aplicativo cliente OSS COBOL:

```
ecobol -Wsystype=oss
       -Wcobol="ansi;port"
       -Wcobol="consult MQ_INSTALLATION_PATH/opt/mqm/lib/mqicb"
       -Wcopylib=MQ_INSTALLATION_PATH/opt/mqm/inc/cmqcobol.cpy
       -LMQ_INSTALLATION_PATH/opt/mqm/lib -lmqic
       -o amq0put0
       MQ_INSTALLATION_PATH/opt/mqm/samp/amq0put0.cbl
```

O exemplo a seguir constrói um aplicativo Guardian cliente COBOL:

```
ecobol -Wsystype=guardian
       -Wcobol="ansi;port;save all"
       -Wcobol="consult MQ_INSTALLATION_PATH/opt/mqm/lib/mqicb"
       -Wcopylib=MQ_INSTALLATION_PATH/opt/mqm/inc/cmqcobol.cpy
       -LMQ_INSTALLATION_PATH/opt/mqm/lib/G -lmqic
       -o amq0put0
       MQ_INSTALLATION_PATH/opt/mqm/samp/amq0put0.cbl
```

Construindo aplicativos usando o compilador Guardian ECOBOL

Esta seção contém exemplos dos comandos que são usados para construir programas que são destinados para o Guardian quando você está usando o compilador Guardian ECOBOL.

MQ_INSTALLATION_SUBVOL representa o volume do Guardian e subvolume no qual o IBM WebSphere MQ está instalado.

O exemplo a seguir constrói um aplicativo Guardian cliente COBOL:

```
ECOBOL /in MQSPUTL/ MQSPUT,MQINSTALLATION_SUBVOL.cmqcobol;
       call-shared;ansi;port;save all;nolist;runnable;
       consult MQINSTALLATION_SUBVOL.mqicb;
       eld(-LMQINSTALLATION_SUBVOL -lmqic)
```

Preparando programas pTAL

Aprenda a construir programas pTAL para o cliente do IBM WebSphere MQ na plataforma HP Integrity NonStop Server

Para construir uma amostra pTAL de código de origem, use o compilador EPTAL.

Nota:

- Os aplicativos pTAL IBM WebSphere MQ devem usar uma rotina principal que é gravada em qualquer uma das linguagens C ou COBOL.
- Os aplicativos pTAL podem ser construídos somente em Guardian.

A tabela a seguir lista a biblioteca que é necessária quando você está preparando programas pTAL em HP Integrity NonStop Server. Deve-se vincular os seus programas com a biblioteca apropriada fornecida pelo IBM WebSphere MQ.

Tabela 65. . Biblioteca de links HP Integrity NonStop Server

Biblioteca	**Descrição**
mqic	Guardian

Construindo aplicativos usando o compilador Guardian EPTAL

Esta seção contém exemplos dos comandos que são usados para construir programas que são destinados para o Guardian quando você está usando o compilador Guardian EPTAL.

MQINSTALLATION_SUBVOL representa o volume do Guardian e subvolume no qual o IBM WebSphere MQ está instalado.

Os aplicativos pTAL IBM WebSphere MQ devem usar uma rotina principal que é gravada em qualquer uma das linguagens C ou COBOL.

O exemplo a seguir constrói um aplicativo cliente Guardian pTAL:

```
ASSIGN SSV0, $SYSTEM.SYSTEM
ASSIGN SSV1, MQINSTALLATION_SUBVOL

EPTAL /in MQINSTALLATION_SUBVOL.MQSPUTT/ MQSPUTO;nolist

CCOMP /in MQINSTALLATION_SUBVOL.MQSPTMC/ MQSPUT;
       runnable,systype guardian,extensions,nolist,
       ssv0 "$system.system",
       ssv1 "MQINSTALLATION_SUBVOL",
       eld(MQSPUTO -LMQINSTALLATION_SUBVOL -lmqic)
```

Construindo seu Aplicativo no HP-UX

Estas informações descrevem as tarefas adicionais e as mudanças nas tarefas padrão que você deve executar ao construir o WebSphere MQ para HP-UX aplicativos para executar em HP-UX.

C, C++ e COBOL são suportados. Para obter informações sobre a preparação de seus programas C++, consulte Using C++.

As tarefas que devem ser executadas para criar um aplicativo executável usando WebSphere MQ para HP-UX variam com a linguagem de programação na qual seu código de origem é gravado. Além de codificar as chamadas MQI em seu código fonte, deve-se incluir as instruções de linguagem apropriadas para incluir os arquivos de inclusão do WebSphere MQ para HP-UX para a linguagem que você está

usando. Familiarize-se com o conteúdo desses arquivos. Consulte “IBM WebSphere MQ arquivos de definição de dados” na página 82 para obter uma descrição completa.

Em todo este tópico, usamos o caractere de barra invertida (\) para dividir os comandos longos em mais de uma linha. Não insira esse caractere; insira cada comando como uma única linha.

Preparando Programas C no HP-UX

Este tópico contém informações a serem consideradas ao preparar programas C no HP-UX; com exemplos para a plataforma IA64 (IPF).

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

Trabalhe em seu ambiente normal. Programas C pré-compilados são fornecidos no diretório *MQ_INSTALLATION_PATH*/samp/bin.

Para obter informações adicionais sobre a programação de aplicativos de 64 bits, consulte Padrões de codificação em plataformas de 64 bits.

Para usar SSL, os clientes MQI do WebSphere MQ no HP-UX devem ser construídos usando encadeamentos do POSIX

Alguns exemplos a serem considerados são:

- “Plataforma IA64 (IPF)” na página 447
- “Vinculando bibliotecas” na página 449

Plataforma IA64 (IPF)

Exemplos de construção de amqsput0, cliexit e srvexit na plataforma IA64(IPF).

O exemplo a seguir constrói o programa de amostra amqsput0 como um aplicativo cliente em um ambiente de 32 bits não encadeado:

```
c89 -Wl,+b,: +e -D_HPUX_SOURCE -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqic
```

O exemplo a seguir constrói o programa de amostra amqsput0 como um aplicativo cliente em um ambiente de 32 bits encadeado:

```
c89 -mt -Wl,+b,: +e -D_HPUX_SOURCE -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqic_r -lpthread
```

O exemplo a seguir constrói o programa de amostra amqsput0 como um aplicativo cliente em um ambiente de 64 bits não encadeado:

```
c89 +DD64 +e -D_HPUX_SOURCE -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic
```

O exemplo a seguir constrói o programa de amostra amqsput0 como um aplicativo cliente em um ambiente de 64 bits encadeado:

```
c89 -mt +DD64 +e -D_HPUX_SOURCE -o amqsputc_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic_r -lpthread
```

O exemplo a seguir constrói o programa de amostra amqsput0 como um aplicativo do servidor em um ambiente de 32 bits não encadeado:

```
c89 -Wl,+b,: +e -D_HPUX_SOURCE -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqm
```

O exemplo a seguir constrói o programa de amostra amqsput0 como um aplicativo do servidor em um ambiente de 32 bits encadeado:

```
c89 -mt -Wl,+b,: +e -D_HPUX_SOURCE -o amqsput_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqm_r -lpthread
```

O exemplo a seguir constrói o programa de amostra amqsput0 como um aplicativo do servidor em um ambiente de 64 bits não encadeado:

```
c89 +DD64 +e -D_HPUX_SOURCE -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm
```

O exemplo a seguir constrói o programa de amostra amqsput0 como um aplicativo do servidor em um ambiente de 64 bits encadeado:

```
c89 -mt +DD64 +e -D_HPUX_SOURCE -o amqsput_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
  -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm_r -lpthread
```

O exemplo a seguir constrói uma saída de cliente cliexit em um ambiente de 32 bits não encadeado:

```
c89 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_32 -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqic
```

O exemplo a seguir constrói uma saída de cliente cliexit em um ambiente de 32 bits encadeado:

```
c89 -mt +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_32_r -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqic_r -lpthread
```

O exemplo a seguir constrói uma saída de cliente cliexit em um ambiente de 64 bits não encadeado:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld -b cliexit.o +ee MQStart -o /var/mqm/exits64/cliexit_64 \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic
```

O exemplo a seguir constrói uma saída de cliente cliexit em um ambiente de 64 bits encadeado:

```
c89 -mt +DD64 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -IMQ_INSTALLATION_PATH/inc
ld  -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_64_r \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic_r -lpthread
```

O exemplo a seguir constrói uma saída de servidor srvexit em um ambiente de 32 bits não encadeado:

```
c89 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_32 -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqm
```

O exemplo a seguir constrói uma saída de servidor srvexit em um ambiente de 32 bits encadeado:

```
c89 -mt +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -IMQ_INSTALLATION_PATH/inc
ld +b: -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_32_r -LMQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqm_r -lpthread
```

O exemplo a seguir constrói uma saída de servidor srvexit em um ambiente de 64 bits não encadeado:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c
-IMQ_INSTALLATION_PATHMQ_INSTALLATION_PATH/inc
```

```
ld -b srvexit.o +ee MQStart -o /var/mqm/exits64/srvexit_64 \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm
```

O exemplo a seguir constrói uma saída de servidor srvexit em um ambiente de 64 bits encadeado:

```
c89 -mt +DD64 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -IMQ_INSTALLATION_PATH/inc
ld  -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_64_r \
-LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm_r -lpthread
```

Vinculando bibliotecas

É necessário vincular seus programas à biblioteca apropriada fornecida pelo WebSphere MQ.

A tabela a seguir mostra qual biblioteca usar em diferentes ambientes

Plataforma de hardware	Ambiente encadeado ou não encadeado	Tipo de programa/saída	Arquivo de bilioteca
IA64 (IPF)	Encadeamento	Servidor e cliente para C	libmqm_r.so
IA64 (IPF)	Encadeamento	Cliente para C	libmqic_r.so
IA64 (IPF)	Não encadeado	Servidor e cliente para C	libmqm.so
IA64 (IPF)	Não encadeado	Cliente para C	libmqic.so

Nota:

1. Se estiver escrevendo um serviço instalável (consulte o Administrando para obter informações adicionais), você precisa vincular à biblioteca `libmqmzf.sl`.
2. Se você estiver produzindo um aplicativo para coordenação externa por um gerenciador de transações compatível com XA, como IBM TXSeries Encinaou BEA Tuxedo, será necessário vincular-se ao `libmqmxa.sl` (ou `libmqmxa64.sl` se seu gerenciador de transações tratar o tipo 'long' como 64 bits) e bibliotecas `libmqz.sl` em um aplicativo não encadeado e às bibliotecas `libmqmxa_r.sl` (ou `libmqmxa64_r.sl`) e `libmqz_r.sl` em um aplicativo encadeado.
3. Você deve vincular bibliotecas do WebSphere MQ antes de quaisquer outras bibliotecas do produto.

Preparando programas COBOL no HP-UX

Aprenda sobre como preparar programas COBOL no HP-UX, usando o Micro Focus Server Express com o WebSphere MQ na plataforma IA64 (IPF) e executando programas no ambiente do cliente MQI do WebSphere MQ .

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

Notas para usuários

1. Copy books COBOL de 32 bits são instalados no seguinte diretório:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   e links simbólicos são criados em:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. Copy books COBOL de 64 bits são instalados no diretório a seguir:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. Nos seguintes exemplos, configure COBCPY para:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

para aplicativos de 32 bits e:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

para aplicativos de 64 bits.

Compile o programa usando o compilador Micro Focus. Os arquivos de cópia que declaram as estruturas estão em *MQ_INSTALLATION_PATH*/inc:

```
$ export LIB=MQ_INSTALLATION_PATH/lib:$LIB
$ export COBCPY="<COBCPY>"
```

Compilando programas de 32 bits:

```
$ cob32 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb Server for COBOL
$ cob32 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb Client for COBOL
$ cob32 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r Threaded Server for COBOL
$ cob32 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r Threaded Client for COBOL
```

Compilando programas de 64 bits:

```
$ cob64 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb Server for COBOL
$ cob64 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb Client for COBOL
$ cob64 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r Threaded Server for COBOL
$ cob64 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r Threaded Client for COBOL
```

em que `amqsput` é um programa de amostra

Certifique-se de ter especificado os tamanhos de pilha de tempo de execução adequados; 16 KB é o mínimo recomendado.

É necessário vincular seus programas à biblioteca apropriada fornecida pelo WebSphere MQ. A tabela a seguir mostra qual biblioteca usar em diferentes ambientes

Plataforma de hardware	**Tipo de programa/saída**	**Arquivo de bilioteca**
IA64 (IPF)	Servidor para COBOL	libmqmcb.so
IA64 (IPF)	Cliente para COBOL	libmqicb.so
IA64 (IPF)	Aplicativos encadeados	libmqmcb_r.so

Usando o Micro Focus Server Express com o WebSphere MQ na plataforma IA64 (IPF)

Consulte “Modelos de Espaço de Endereço suportados pelo WebSphere MQ para HP-UX em IA64 (IPF)” na página 452 para obter detalhes sobre o uso do Micro Focus Server Express em conjunto com o WebSphere MQ na plataforma HP/IPF.

Programas para executar no ambiente do cliente MQI do WebSphere MQ

Se estiver usando LU 6.2 para conectar o cliente de MQI a um servidor, vincule seu aplicativo a libsna.a, parte do produto SNAplusAPI. Use as opções `-lV3` e `-lstr` em seu comando compile e link.

- A opção `-lV3` fornece ao programa acesso à biblioteca de sinalização AT & T (o SNAplusAPI usa sinais AT & T)
- A opção `-lstr` vincula seu programa ao componente de fluxos

Preparando Programas CICS no HP-UX

Aprenda a construir programas de transação do CICS no HP-UX

Para construir a transação CICS de amostra, amqscic0.ccs, execute o comando a seguir:

```
$ export USERLIB="-lmqm_r"
$ cicstcl -l C amqscic0.ccs
```

Um módulo do comutador XA é fornecido para permitir que você vincule o CICS ao WebSphere MQ:

Tabela 66. Código essencial para aplicativos CICS (HP-UX)

Descrição	**C (origem)**	**C (exec)**
Rotina de inicialização de XA	amqzscix.c	amqzsc

É possível localizar mais informações sobre como suportar transações do CICS no Administrando

Suporte TXSeries CICS

WebSphere MQ no HP-UX suporta TXSeries CICS usando a interface XA. Assegure que os aplicativos CICS estejam vinculados à versão encadeada das bibliotecas do MQ .

Grave WebSphere MQ programas que são carregados na mesma região do CICS em C ou COBOL. Não é possível fazer uma combinação de chamadas MQI C e COBOL na mesma região do CICS A maioria das chamadas MQI na segunda linguagem usada falha com um código de razão de `MQRC_HOBJ_ERROR`.

Transação de amostra CICS C

A origem C de amostra para uma transação do CICS WebSphere MQ é fornecida pelo AMQSCIC0.CCS. A transação lê mensagens da fila de transmissão SYSTEM.SAMPLE.CICS.WORKQUEUE no gerenciador de fila padrão e coloca-os na fila local com o nome da fila que está contido no cabeçalho de transmissão da mensagem Quaisquer falhas são enviadas à fila SYSTEM.SAMPLE.CICS.DLQ. Use o script de MQSC de amostra AMQSCIC0.TST para criar estas filas e filas de entrada de amostra.

Preparando programas COBOL CICS usando Micro Focus COBOL

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

Para usar Micro Focus COBOL, siga estas etapas:

1. Inclua o módulo da biblioteca de tempo de execução COBOL do WebSphere MQ na biblioteca de tempo de execução usando o comando a seguir:

   ```
   cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
               MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe_r
   ```

   Nota: Com o `cicsmkcobol`, o WebSphere MQ não permite fazer chamadas MQI na linguagem de programação C a partir do aplicativo COBOL.

   Se seus aplicativos existentes tiverem quaisquer chamadas desse tipo, será recomendado que você mova estas funções dos aplicativos COBOL para sua própria biblioteca, por exemplo, `myMQ.so`. Depois de mover essas funções, não inclua a biblioteca WebSphere MQ `libmqmcbrt.o` ao construir o aplicativo COBOL para CICS.

Além disso, se seu aplicativo COBOL não fizer nenhuma chamada MQI COBOL, não vincule `libmqmz_r` com `cicsmkcobol`.

Isso cria o arquivo de método de linguagem COBOL do Micro Focus e permite que a biblioteca COBOL de tempo de execução do CICS chame o WebSphere MQ em sistemas UNIX and Linux

Nota: Execute `cicsmkcobol` somente ao instalar um dos produtos a seguir:

- Nova versão ou liberação do Micro Focus COBOL
- Nova versão ou liberação do CICS para HP-UX
- Nova versão ou liberação de qualquer produto de banco de dados suportado (apenas para transações COBOL)
- Nova versão ou liberação do WebSphere MQ

2. Exporte a variável de ambiente a seguir:

```
COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
```

3. Converta, compile e vincule o programa digitando:

```
cicstcl -l COBOL -e <yourprog>.ccp
```

Modelos de Espaço de Endereço suportados pelo WebSphere MQ para HP-UX em IA64 (IPF)

O HP-UX fornece vários modelos de espaço de endereço que podem ser explorados por aplicativos WebSphere MQ

O HP-UX suporta dois modelos de Espaço de Endereço:

- MGAS-Espaço de endereço global em grande parte (este é o padrão e é usado pelo WebSphere MQ)
- MPAS - Mostly private address space

Os aplicativos que se conectam ao WebSphere MQ podem usar os modelos de espaço de endereço MGAS ou MPAS Aplicativos construídos usando o modelo MPAS que se conectam ao WebSphere MQ usando memória compartilhada podem incorrer em um menor custo de desempenho devido à ineficiência no mapeamento das páginas de memória compartilhada usadas pelo WebSphere MQ no espaço de endereço virtual do programa MPAS.

Os aplicativos em COBOL construídos usando o Micro Focus Server Express usam o modelo MPAS por padrão.

É possível usar o programa **chatr** para verificar e mudar o modelo de endereçamento usado por um programa.

Se você encontrar problemas ao se conectar ao WebSphere MQ a partir de programas MPAS de 32 bits, considere usar o modelo de endereçamento MGAS ou construir seu aplicativo como um aplicativo MPAS de 64 bits em vez de um aplicativo MPAS de 32 bits.

Mais detalhes sobre os modelos de espaço de endereço MGAS e MPAS podem ser localizados na documentação do HP-UX

Construindo seu aplicativo no Linux

Estas informações descrevem as tarefas adicionais e as mudanças nas tarefas padrão que devem ser executadas ao construir aplicativos WebSphere MQ para Linux executar.

C e C++ são suportados. Para obter informações sobre a preparação de seus programas C++, consulte Using C++.

Preparando programas C no Linux

Programas C pré-compilados são fornecidos no diretório *MQ_INSTALLATION_PATH*/samp/bin . Para construir uma amostra usando o código-fonte, use o compilador gcc.

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

Trabalhe em seu ambiente normal. Para obter informações adicionais sobre a programação de aplicativos de 64 bits, consulte Padrões de codificação em plataformas de 64 bits.

Vinculando bibliotecas

A tabela a seguir lista as bibliotecas que são necessárias ao preparar programas C no Linux.

- É necessário vincular seus programas à biblioteca apropriada fornecida pelo WebSphere MQ.

  Em um ambiente não encadeado, vincule-se a uma das seguintes bibliotecas:

Arquivo de bilioteca	Tipo de programa/saída
libmqm.so	Servidor para C
libmqic.so & libmqm.so	Cliente para C

  Em um ambiente encadeado, vincule-se a uma das seguintes bibliotecas:

Arquivo de bilioteca	Tipo de programa/saída
libmqm_r.so	Servidor para C
libmqic_r.so & libmqm_r.so	Cliente para C

  Nota:

  1. Se você estiver gravando um serviço instalável (consulte o Administrando para obter informações adicionais), será necessário vincular a biblioteca libmqmzf.so.
  2. Se você estiver produzindo um aplicativo para coordenação externa por um gerenciador de transações compatível com XA, como IBM TXSeries Encinaou BEA Tuxedo, será necessário vincular-se ao libmqmxa.so (ou libmqmxa64.so se seu gerenciador de transações tratar o tipo 'long' como 64 bits) e bibliotecas libmqz.so em um aplicativo não encadeado e às bibliotecas libmqmxa_r.so (ou libmqmxa64_r.so) e libmqz_r.so em um aplicativo encadeado.
  3. Você deve vincular bibliotecas do WebSphere MQ antes de quaisquer outras bibliotecas do produto.

Criando aplicativos de 31 bits

Este tópico contém exemplos dos comandos usados para a construção de programas de 31 bits em vários ambientes.

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

Aplicativo cliente C, 31 bits, não encadeado

```
gcc -m31  -o famqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic
```

Aplicativo cliente C, de 31 bits, encadeado

```
gcc -m31  -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

Aplicativo do servidor C, de 31 bits, não encadeado

```
gcc -m31  -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

Aplicativo do servidor C, de 31 bits, encadeado

```
gcc -m31  -o amqsput_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Aplicativo cliente C + +, 31 bits, não encadeado

```
g++ -m31 -fsigned-char  -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl
-limqb23gl -lmqic
```

Aplicativo cliente C + +, de 31 bits, encadeado

```
g++ -m31 -fsigned-char  -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r
-limqb23gl_r -lmqic_r -lpthread
```

Aplicativo do servidor C + +, de 31 bits, não encadeado

```
g++ -m31 -fsigned-char  -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl
-limqb23gl -lmqm
```

Aplicativo do servidor C + +, de 31 bits, encadeado

```
g++ -m31 -fsigned-char  -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r
-limqb23gl_r -lmqm_r -lpthread
```

Saída cliente C, de 31 bits, não encadeada

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic
```

Saída cliente C, de 31 bits, encadeada

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/cliexit_32_r cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic_r -lpthread
```

Saída do servidor C, de 31 bits, não encadeada

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/srvexit_32 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqm
```

Saída do servidor C, de 31 bits, encadeada

```
gcc -m31 -shared -fPIC  -o /var/mqm/exits/srvexit_32_r srvexit.c
```

```
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqm_r -lpthread
```

Criando aplicativos de 32 bits

Este tópico contém exemplos dos comandos usados para a construção de programas de 32 bits em vários ambientes.

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

Aplicativo do cliente C, 32 bits, não encadeado

```
gcc -m32 -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic
```

Aplicativo do cliente C, 32 bits, encadeado

```
gcc -m32 -o amqsputc_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

Aplicativo do servidor C, 32 bits, não encadeado

```
gcc -m32 -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

Aplicativo do servidor C, 32 bits, encadeados

```
gcc -m32 -o amqsput_32_r amqsput0.c -IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Aplicativo do cliente C++, 32 bits, não encadeado

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib  -Wl,-rpath=/usr/lib
-limqc23gl -limqb23gl -lmqic
```

Aplicativo do cliente C++, 32 bits, encadeado

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib  -Wl,-rpath=/usr/lib
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Aplicativo do servidor C++, 32 bits, não encadeado

```
g++ -m32 -fsigned-char -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl -limqb23gl -lmqm
```

Aplicativo do servidor C++, 32 bits, encadeado

```
g++ -m32 -fsigned-char -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

Saída do cliente C, 32 bits, não encadeado

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic
```

Saída do cliente C, 32 bits, encadeado

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/cliexit_32_r cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
-lmqic_r -lpthread
```

Saída do servidor C, 32 bits, não encadeado

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/srvexit_32 srvexit.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

Saída do servidor C, 32 bits, encadeado

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/srvexit_32_r srvexit.c
IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib
lmqm_r -lpthread
```

Criando aplicativos de 64 bits

Este tópico contém exemplos dos comandos usados para a construção de programas de 64-bit bits em vários ambientes.

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

Aplicativo cliente C, 64 bits, não encadeado

```
gcc -m64  -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqic
```

Aplicativo cliente C, 64 bits, encadeado

```
gcc -m64  -o amqsputc_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqic_r
-lpthread
```

Aplicativo do servidor C, de 64 bits, não-encadeados

```
gcc -m64  -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm
```

Aplicativo do servidor C, de 64-bit bits, encadeados

```
gcc -m64  -o amqsput_64_r amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm_r
-lpthread
```

Aplicativo cliente C ++, 64 bits, não encadeado

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Aplicativo cliente C++, 64 bits, encadeado

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp
```

```
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Aplicativo do servidor C ++, de 64 bits, não encadeado

```
g++ -m64 -fsigned-char  -o imqsput_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -limqs23gl -limqb23gl -lmqm
```

Aplicativo do servidor C ++, de 64 bits, encadeados

```
g++ -m64 -fsigned-char  -o imqsput_64_r imqsput.cpp
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

Saída de cliente C, 64 bits, não encadeado

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/cliexit_64 cliexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqic
```

Saída cliente C, de 64-bit bits, encadeados

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/cliexit_64_r cliexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqic_r -lpthread
```

Saída do servidor C, de 64-bit bits, não encadeado

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/srvexit_64 srvexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqm
```

Saída do servidor C, de 64-bit bits, encadeados

```
gcc -m64 -shared -fPIC  -o /var/mqm/exits64/srvexit_64_r srvexit.c
-IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqm_r -lpthread
```

Preparando programas COBOL em Linux

Aprenda como preparar programas COBOL no Linux e como preparar programas COBOL usando Micro Focus COBOL.

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

1. Copy books COBOL de 32 bits são instalados no seguinte diretório:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   e links simbólicos são criados em:

```
MQ_INSTALLATION_PATH/inc
```

2. Em plataformas de 64 bit bits, copy books COBOL de 64 bits são instalados no seguinte diretório:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

3. Nos seguintes exemplos, configure COBCPY para:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

para aplicativos de 32 bits e:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

para aplicativos de 64 bits.

É necessário vincular o seu programa a um dos seguintes:

Arquivo de bilioteca	Tipo de programa/saída
libmqmcb.so	Servidor para COBOL
libmqicb.so	Cliente para COBOL
libmqmcb_r.so	Servidor para COBOL (aplicativo encadeado)
libmqicb_r.so	Cliente para COBOL (aplicativo encadeado)

Preparando programas COBOL usando o Micro Focus COBOL

Configure as variáveis de ambiente antes de compilar seu programa da seguinte forma:

```
export COBCPY=<COBCPY>
export LIB=MQ_INSTALLATION_PATHlib:$LIB
```

Para compilar um programa COBOL de 32 bits, quando suportado, usando o Micro Focus COBOL, insira:

```
  $ cob32 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb Server for COBOL
  $ cob32 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb Client for COBOL
  $ cob32 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r Threaded Server for COBOL
  $ cob32 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r Threaded Client for COBOL
```

Para compilar um programa COBOL de 64 bits usando o Micro Focus COBOL, insira:

```
  $ cob64 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb Server for COBOL
  $ cob64 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb Client for COBOL
  $ cob64 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r Threaded Server for COBOL
  $ cob64 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r Threaded Client for COBOL
```

em que `amqsput` é um programa de amostra

Consulte a documentação Micro Focus COBOL para uma descrição das variáveis de ambiente que você precisa.

Construindo seu aplicativo no Solaris

Essas informações descrevem as tarefas adicionais e as mudanças nas tarefas padrão que devem ser executadas ao construir o WebSphere MQ para aplicativos Solaris para execução no Solaris. .

As linguagens de programação COBOL, C e C++ são suportadas. Para obter informações sobre a preparação de seus programas C++, consulte Using C++.

Além de codificar as chamadas MQI em seu código fonte, deve-se incluir os arquivos de inclusão apropriados. Familiarize-se com o conteúdo desses arquivos. Consulte “IBM WebSphere MQ arquivos de definição de dados” na página 82 para obter uma descrição completa.

Em todo este tópico, o caractere de barra invertida (\) é usado para dividir os comandos longos em mais de uma linha. Não insira este caractere, insira cada comando como uma única linha.

Preparando Programas C no Solaris

Programas C pré-compilados são fornecidos no diretório *MQ_INSTALLATION_PATH*/samp/bin.

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

Para obter informações adicionais sobre a programação de aplicativos de 64 bits, consulte Padrões de codificação em plataformas de 64 bits.

Se desejar usar os programas em uma máquina que tenha apenas o cliente MQI do WebSphere MQ para Solaris instalado, compile os programas para vinculá-los à biblioteca do cliente (-lmqic).

Se você usar o compilador não suportado ?usr?ucb?cc, seu aplicativo poderá compilar e vincular com êxito. No entanto, quando você executa o aplicativo, ele falha quando tenta se conectar ao gerenciador de filas.

Nota: Clientes SSL e TLS do Solaris x86 de 32 bits configurados para operação compatível com FIPS 140-2 falham ao executar em sistemas Intel. Esta falha ocorre porque o arquivo de biblioteca GSKit-Crypto Solaris x86 32 bits compatível com FIPS 140-2 não carrega no Intel Chipset. Nos sistemas afetados, o erro AMQ9655 é relatado no log de erros do cliente. Para resolver esse problema, desative a conformidade com FIPS 140-2 ou recompile o aplicativo cliente de 64 bits porque o código de 64 bits não é afetado.

Vinculando bibliotecas

Você deve se vincular às bibliotecas do WebSphere MQ que são apropriadas para seu tipo de aplicativo:

Arquivos de biblioteca	Tipo de programa/saída
libmqm.so	Servidor para C
libmqic.so & libmqm.so	Cliente para C

Nota:

1. Se você estiver gravando um serviço instalável (para obter informações adicionais, consulte o Administrando), vincule à biblioteca libmqmzf.so.
2. Se você estiver produzindo um aplicativo para coordenação externa por um gerenciador de transações compatível com XA, como o IBM TXSeries Encinaou BEA Tuxedo, deve-se vincular ao libmqmxa.so (ou libmqmxa64.so se o gerenciador de transações tratar o tipo 'long' como 64 bits) e bibliotecas libmqz.so
3. Você deve vincular bibliotecas do WebSphere MQ antes de quaisquer outras bibliotecas do produto.

Construindo aplicativos em x86-64

Este tópico contém exemplos dos comandos usados para construir programas em diversos ambientes na plataforma x86-64.

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

Aplicativo do cliente C, 32 bits

```
cc -xarch=386 -mt -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
```

```
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqic -lsocket -lnsl -ldl
```

Aplicativo do cliente C, 64 bits

```
cc -xarch=amd64 -mt -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqic -lsocket
-lnsl -ldl
```

Aplicativo do servidor C, 32 bits

```
cc -xarch=386 -mt -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqm -lsocket -lnsl -ldl
```

Aplicativo do servidor C, 64 bits

```
cc -xarch=amd64 -mt -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm -lsocket
-lnsl -ldl
```

Aplicativo cliente C++, 32 bits

```
CC -xarch=386 -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as -lmqic -lsocket -lnsl -ldl
```

Aplicativo do cliente C++, 64 bits

```
CC -xarch=amd64 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Aplicativo do servidor C++, 32 bits

```
CC -xarch=386 -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as -lmqm
-lsocket -lnsl -ldl
```

Aplicativo do servidor C++, 64 bits

```
CC -xarch=amd64 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as -lmqm
-lsocket -lnsl -ldl
```

Saída do cliente C, 32 bits

```
cc -xarch=386 -mt -G -KPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
-R/usr/lib/32 -lmqic
-lsocket -lnsl -ldl
```

Saída do cliente C, 64 bits

```
cc -xarch=amd64 -mt -G -KPIC -o /var/mqm/exits64/cliexit_64 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqic
-lsocket -lnsl -ldl
```

Saída do servidor C, 32 bits

```
cc -xarch=386 -mt -G -KPIC -o /var/mqm/exits/srvexit_32 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
-R/usr/lib/32 -lmqm
-lsocket -lnsl -ldl
```

Saída do servidor C, 64 bits

```
cc -xarch=amd64 -mt -G -KPIC -o /var/mqm/exits64/srvexit_64 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqm
-lsocket -lnsl -ldl
```

Construindo aplicativos no SPARC

Este tópico contém exemplos dos comandos usados para construir programas em vários ambientes na plataforma SPARC.

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

Aplicativo do cliente C, 32 bits

```
cc -xarch=v8plus -mt -o amqsputc_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqic -lsocket -lnsl -ldl
```

Aplicativo do cliente C, 64 bits

```
cc -xarch=v9 -mt -o amqsputc_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqic
-lsocket -lnsl -ldl
```

Aplicativo do servidor C, 32 bits

```
cc -xarch=v8plus -mt -o amqsput_32 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqm -lsocket -lnsl -ldl
```

Aplicativo do servidor C, 64 bits

```
cc -xarch=v9 -mt -o amqsput_64 amqsput0.c -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm
-lsocket -lnsl -ldl
```

Aplicativo cliente C++, 32 bits

```
CC -xarch=v8plus -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic
-lsocket -lnsl -ldl
```

Aplicativo do cliente C++, 64 bits

```
CC -xarch=v9 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Aplicativo do servidor C++, 32 bits

```
CC -xarch=v8plus -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib
```

```
-RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as -lmqm
-lsocket -lnsl -ldl
```

Aplicativo do servidor C++, 64 bits

```
CC -xarch=v9 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as -lmqm
-lsocket -lnsl -ldl
```

Saída do cliente C, 32 bits

```
cc -xarch=v8plus -mt -G -KPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
-R/usr/lib/32 -lmqic
-lsocket -lnsl -ldl
```

Saída do cliente C, 64 bits

```
cc -xarch=v9 -mt -G -KPIC -o /var/mqm/exits64/cliexit_64 cliexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqic
-lsocket -lnsl -ldl
```

Saída do servidor C, 32 bits

```
cc -xarch=v8plus -mt -G -KPIC -o /var/mqm/exits/srvexit_32 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib
-R/usr/lib/32 -lmqm
-lsocket -lnsl -ldl
```

Saída do servidor C, 64 bits

```
cc -xarch=v9 -mt -G -KPIC -o /var/mqm/exits64/srvexit_64 srvexit.c
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64
-R/usr/lib/64 -lmqm
-lsocket -lnsl -ldl
```

Preparando programas COBOL no Solaris

Saiba como preparar programas COBOL no Solaris.

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

1. Copy books COBOL de 32 bits são instalados no seguinte diretório:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   e links simbólicos são criados em:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. Copy books COBOL de 64 bits são instalados no diretório a seguir:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. Nos seguintes exemplos, configure COBCPY para:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

para aplicativos de 32 bits e:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

para aplicativos de 64 bits.

Compile os programas usando o compilador Micro Focus. Os arquivos de cópia que declaram as estruturas estão em *MQ_INSTALLATION_PATH*/inc:

```
$ export LIB=MQ_INSTALLATION_PATH/lib:$LIB
$ export COBCPY="<COBCPY>"
```

Compilando programas de 32 bits:

- `$ cob32 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb`

  Servidor para COBOL
- `$ cob32 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb`

  Cliente para COBOL
- `$ cob32 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r`

  Servidor encadeado para COBOL
- `$ cob32 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r`

  Cliente encadeado para COBOL

Compilando programas de 64 bits:

- `$ cob64 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb`

  Servidor para COBOL
- `$ cob64 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb`

  Cliente para COBOL
- `$ cob64 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r`

  Servidor encadeado para COBOL
- `$ cob64 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r`

  Cliente encadeado para COBOL

em que *amqs0put0.cbl* é um programa de amostra.

Deve-se vincular seu programa com um dos seguintes:

- libmqmcb.so

  Servidor para COBOL
- libmqicb.so

  Cliente para COBOL

Preparando programas CICS no Solaris

Saiba como preparar programas CICS no Solaris.

Um módulo do comutador XA é fornecido para permitir que você vincule o CICS ao WebSphere MQ:

Tabela 67. Código essencial para aplicativos CICS (Solaris)		
Descrição	**C (origem)**	**C (exec)**
Rotina de inicialização de XA	amqzscix.c	amqzsc - TXSeries para Solaris

Sempre vincule suas transações ao encadeamento seguro WebSphere MQ biblioteca libmqm.so.

É possível localizar mais informações sobre como suportar transações do CICS no Administrando

Suporte TXSeries CICS

WebSphere MQ para Solaris suporta TXSeries CICS usando a interface XA.

Grave WebSphere MQ programas que são carregados na mesma região do CICS em C ou COBOL. Não é possível fazer uma combinação de chamadas MQI C e COBOL na mesma região do CICS A maioria das chamadas MQI na segunda linguagem usada falha com um código de razão de MQRC_HOBJ_ERROR.

Preparando programas COBOL CICS usando Micro Focus COBOL

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

Para usar Micro Focus COBOL, siga estas etapas:

1. Inclua o módulo da biblioteca de tempo de execução COBOL do WebSphere MQ na biblioteca de tempo de execução usando o comando a seguir:

   ```
   cicsmkcobol -L/usr/lib/dce -LMQ_INSTALLATION_PATH/lib  \
               MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe
   ```

   Nota: Com o `cicsmkcobol`, o WebSphere MQ não permite fazer chamadas MQI na linguagem de programação C a partir do aplicativo COBOL.

   Se seus aplicativos existentes tiverem quaisquer chamadas desse tipo, mova essas funções dos aplicativos COBOL para sua própria biblioteca, por exemplo, `myMQ.so`. Depois de mover essas funções, não inclua a biblioteca WebSphere MQ `libmqmcbrt.o` ao construir o aplicativo COBOL para CICS.

   Além disso, se seu aplicativo COBOL não fizer nenhuma chamada MQI COBOL, não vincule `libmqmz_r` com `cicsmkcobol`.

   Isso cria o arquivo de método de linguagem COBOL do Micro Focus e permite que a biblioteca COBOL de tempo de execução do CICS chame o WebSphere MQ em sistemas UNIX and Linux

   Nota: Execute `cicsmkcobol` somente ao instalar um dos produtos a seguir:

   - Nova versão ou liberação do Micro Focus COBOL
   - Nova versão ou liberação do TXSeries para Solaris
   - Nova versão ou liberação de qualquer produto de banco de dados suportado (apenas para transações COBOL)
   - Nova versão ou liberação do WebSphere MQ
2. Exporte a variável de ambiente a seguir:

   ```
   COBCPY=MQ_INSTALLATION_PATH/inc  export COBCPY
   ```
3. Converta, compile e vincule o programa digitando:

   ```
   cicstcl -l COBOL -e <yourprog>.ccp
   ```

Preparando programas CICS C

Construa programas CICS C usando os recursos padrão do CICS :

1. Exporte ***uma*** das variáveis de ambiente a seguir:
   - LDFLAGS = "-L*MQ_INSTALLATION_PATH*! lib -lmqm_r" exportar LDFLAGS
   - USERLIB = "-L*MQ_INSTALLATION_PATH*overlib -lmqm_r" exportação USERLIB
2. Converta, compile e vincule o programa digitando:

```
cicstcl -l C amqscic0.ccs
```

Transação de amostra CICS C

A origem C de amostra para uma transação do CICS WebSphere MQ é fornecida pelo AMQSCIC0.CCS. A transação lê mensagens da fila de transmissão SYSTEM.SAMPLE.CICS.WORKQUEUE no gerenciador de filas padrão e coloca-os na fila local com um nome de fila que está contido no cabeçalho de transmissão da mensagem Quaisquer falhas são enviadas à fila SYSTEM.SAMPLE.CICS.DLQ. Use o script de MQSC de amostra AMQSCIC0.TST para criar estas filas e filas de entrada de amostra.

Construindo seu aplicativo em sistemas Windows

As publicações de sistemas Windows descrevem como construir aplicativos executáveis a partir dos programas gravados.

Este tópico descreve as tarefas adicionais e as mudanças nas tarefas padrão que devem ser executadas ao construir WebSphere MQ para aplicativos Windows para execução em sistemas Windows . As linguagens de programação ActiveX, C, C++, COBOL e Visual Basic são suportadas. Para obter informações sobre como preparar seus programas ActiveX, consulte Usando a Interface do modelo de objeto componente (Classes de automação do WebSphere MQ for ActiveX). Para obter informações sobre a preparação de seus programas C++, consulte Using C++.

As tarefas que devem ser executadas para criar um aplicativo executável usando o WebSphere MQ para Windows variam com a linguagem de programação na qual seu código-fonte é gravado. Além de codificar as chamadas MQI em seu código fonte, deve-se incluir as instruções de linguagem apropriadas para incluir os arquivos de inclusão do WebSphere MQ para Windows para a linguagem que você está usando. Familiarize-se com o conteúdo desses arquivos. Consulte “IBM WebSphere MQ arquivos de definição de dados” na página 82 para obter uma descrição completa.

Construindo aplicativos de 64 bits no Windows

Aplicativos de 32 bits e de 64 bits são suportadas no IBM WebSphere MQ for Windows Version 7.5. Os arquivos executáveis e de biblioteca do IBM WebSphere MQ são fornecidos nos formatos de 32 bits e de 64 bits, use a versão apropriada dependendo do aplicativo com o qual está trabalhando.

Arquivos executáveis e bibliotecas

Ambas as versões de 32 bits e de 64 bits das bibliotecas do IBM WebSphere MQ são fornecidas nos locais a seguir:

Tabela 68. Local de bibliotecas do IBM WebSphere MQ

Versão da biblioteca	**Diretório que contém arquivos de biblioteca**
32 bits	*MQ_INSTALLATION_PATH*\Tools\Lib
64 bits	*MQ_INSTALLATION_PATH*\Tools\Lib64

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

Aplicativos de 32 bits continuam a funcionar normalmente após a migração. Os arquivos de 32 bits existem no mesmo diretório que em versões anteriores do produto.

Se desejar criar a versão de 64 bits, deve-se assegurar que seu ambiente esteja configurado para usar os arquivos de biblioteca em *MQ_INSTALLATION_PATH* \Tools\Lib64. Assegure que a variável de ambiente LIB não esteja configurada para procurar na pasta que contém as bibliotecas de 32 bits.

Preparando programas C no Windows

Trabalhar em seu típico ambiente do Windows ; WebSphere MQ para Windows não requer nada de especial.

Para obter informações adicionais sobre a programação de aplicativos de 64 bits, consulte Padrões de codificação em plataformas de 64 bits.

- Vincule seus programas às bibliotecas apropriadas fornecidas pelo WebSphere MQ:

Arquivo de bilioteca	Tipo de programa/saída
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	servidor para 32 bits C
MQ_INSTALLATION_PATH\Tools\Lib\mqic.lib	cliente para 32 bits C
MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib	cliente para 32 bits C com a coordenação de transação
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	servidor para 64 bits C
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	cliente para 64 bits C
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	cliente para 64 bits C com coordenação de transação

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

O comando a seguir fornece um exemplo de compilação do programa de amostra amqsget0 (usando o compilador Microsoft Visual C++).

Para aplicativos de 32 bits:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib
```

Para aplicativos de 64 bits:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib
```

Nota:

– Se você estiver gravando um serviço instalável (consulte o Administrando para obter informações adicionais), você precisa se vincular à biblioteca mqmzf.lib.

– Se você estiver produzindo um aplicativo para coordenação externa por um gerenciador de transações compatível com XA, como IBM TXSeries Encinaou BEA Tuxedo, será necessário se vincular à biblioteca mqmxa.lib ou mqmxa.lib .

  - Se você estiver gravando uma saída do CICS , vincule à biblioteca mqmcics4.lib .
  - Você deve vincular bibliotecas do WebSphere MQ antes de quaisquer outras bibliotecas do produto.
- As DLLs devem estar no caminho (PATH) que você especificou.
- Se você usar caracteres minúsculos sempre que possível, será possível mover do WebSphere MQ para Windows para WebSphere MQ em sistemas UNIX and Linux , em que o uso de minúsculos é necessário.

Preparando programas CICS e Transaction Server

A origem C de amostra para uma transação do CICS WebSphere MQ é fornecida pelo AMQSCIC0.CCS. Você o constrói usando os recursos padrão do CICS . Por exemplo, para TXSeries para Windows 2000:

1. Configure a variável de ambiente (insira o código a seguir em uma linha):

```
set CICS_IBMC_FLAGS=-I MQ_INSTALLATION_PATH\Tools\C\Include;
%CICS_IBMC_FLAGS%
```

2. Configure a variável de ambiente USERLIB:

```
set USERLIB=MQM.LIB;%USERLIB%
```

3. Converta, compile e vincule o programa de amostra:

```
cicstcl -l IBMC amqscic0.ccs
```

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

Isso é descrito no *Transaction Server for Windows NT Application Programming Guide (CICS) V4*.

É possível localizar mais informações sobre como suportar transações do CICS no Administrando

Preparando programas COBOL em Windows

Use estas informações para aprender a preparar programas COBOL no Windows e preparar os programas CICS e Transaction Server.

1. Os copy books COBOL de 32 bits são instalados no diretório a seguir: *MQ_INSTALLATION_PATH* \Tools\cobol\CopyBook.
2. Os copy books COBOL de 64 bits são instalados no diretório a seguir: *MQ_INSTALLATION_PATH* \Tools\cobol\CopyBook64
3. Nos exemplos a seguir, configure CopyBook para:

```
CopyBook
```

   para aplicativos de 32 bits e:

```
CopyBook64
```

   para aplicativos de 64 bits.

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

Para preparar programas COBOL em sistemas Windows, vincule seu programa a uma das bibliotecas a seguir fornecidas pelo IBM WebSphere MQ:

Arquivo de bilioteca	Tipo de programa ou saída
MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb	Servidor de 32 bits para IBM COBOL

Arquivo de bilioteca	Tipo de programa ou saída
MQ_INSTALLATION_PATH\Tools\Lib\mqmcb	Servidor de 32 bits de Micro Focus COBOL
MQ_INSTALLATION_PATH\Tools\Lib\mqiccbb	Clientes de 32 bits para IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib\mqiccb	Cliente de 32 bits para Micro Focus COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb	Servidor de 64 bits para IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb	Servidor de 64 bits para Micro Focus COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccbb	Cliente de 64 bits para IBM COBOL
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb	Cliente de 64 bits para Micro Focus COBOL

Quando você estiver executando um programa no ambiente do cliente MQI, certifique-se de que a biblioteca DOSCALLS apareça antes de qualquer biblioteca COBOL ou IBM WebSphere MQ.

É possível usar o compilador IBM COBOL Set ou o compilador Micro Focus COBOL, dependendo do programa:

- Programas que começam com `amqi` são adequados para o compilador IBM COBOL Set,
- Programas iniciados por `amqm` são adequados para o compilador Micro Focus COBOL e
- Programas iniciados por `amq0` são adequados para ambos os compiladores.

IBM e Micro Focus COBOL

Revincule quaisquer programas IBM WebSphere MQ Micro Focus COBOL de 32 bits usando `mqmcb.lib` ou `mqiccb.lib`, em vez de bibliotecas `mqmcbb` e `mqiccbb`.

Para compilar, por exemplo, o programa de amostra `amq0put0`, usando IBM VisualAge COBOL:

1. Configure a variável de ambiente SYSLIB para incluir o caminho para os copybooks COBOL IBM WebSphere MQ VisualAge (insira o código a seguir em uma linha):

   ```
   set SYSLIB=MQ_INSTALLATION_PATH\
   Tools\Cobol\Copybook\VAcobol;%SYSLIB%
   ```

2. Para uso no servidor IBM WebSphere MQ :

   ```
   cob2 amq0put0.cbl -qlib "MQ_INSTALLATION_PATH\
   Tools\Lib\mqmcbb.lib"
   ```

3. Para uso no cliente IBM WebSphere MQ :

   ```
   cob2 amq0put0.cbl -qlib "MQ_INSTALLATION_PATH\
   Tools\Lib\mqiccbb.lib"
   ```

   Nota: Embora deva-se usar a opção do compilador `CALLINT(SYSTEM)`, esse é o padrão para `cob2`.

Para compilar, por exemplo, o programa de amostra `amq0put0`, usando o Micro Focus COBOL:

1. Configure a variável de ambiente COBCPY para apontar para os copy books do IBM WebSphere MQ COBOL (insira o código a seguir em uma linha):

   ```
   set COBCPY=MQ_INSTALLATION_PATH\
   Tools\Cobol\Copybook
   ```

2. Compile o programa para fornecer a você um arquivo de objeto:

   ```
   cobol amq0put0 LITLINK
   ```

3. Vincule o arquivo de objeto para o sistema de tempo de execução.
   - Configure a variável de ambiente LIB para apontar para as bibliotecas COBOL bibliotecas do compilador.
   - Vincule o arquivo de objeto para uso no servidor IBM WebSphere MQ :

     ```
     cbllink amq0put0.obj mqmcb.lib
     ```

   - Ou vincule o arquivo de objeto para uso no cliente do IBM WebSphere MQ :

     ```
     cbllink amq0put0.obj mqiccb.lib
     ```

Preparando programas CICS e Transaction Server

Para compilar e vincular um programa TXSeries para Windows NT, V5.1 usando IBM VisualAge COBOL:

1. Configure a variável de ambiente (insira o código a seguir em uma linha):

   ```
   set CICS_IBMCOB_FLAGS=MQ_INSTALLATION_PATH\
   Cobol\Copybook\VAcobol;%CICS_IBMCOB_FLAGS%
   ```

2. Configure a variável de ambiente USERLIB:

   ```
   set USERLIB=MQMCBB.LIB
   ```

3. Converta, compile e vincule o programa:

   ```
   cicstcl -l IBMCOB myprog.ccp
   ```

Isso é descrito no *Transaction Server for Windows NT, V4 Application Programming* Guide.

Para compilar e vincular um programa CICS for Windows V5 usando o Micro Focus COBOL:

- Configure a variável INCLUDE:

  ```
  set
  INCLUDE=<drive>:\<programname>\ibm\websphere\tools\c\include;
          <drive>:\opt\cics\include;%INCLUDE%
  ```

- Configure a variável de ambiente COBCPY:

  ```
   setCOBCPY=<drive>:\<programname>\ibm\websphere\tools\cobol\copybook;
             <drive>:\opt\cics\include
  ```

- Configure as opções de COBOL:
  - `set`
  - `COBOPTS=/LITLINK /NOTRUNC`

e execute o código a seguir:

```
cicstran cicsmq00.ccp
cobol cicsmq00.cbl /LITLINK /NOTRUNC
cbllink -D -Mcicsmq00 -Ocicsmq00.cbmfnt cicsmq00.obj
%CICSLIB%\cicsprCBMFNT.lib user32.lib msvcrt.lib kernel32.lib mqmcb.lib
```

Preparando programas do Visual Basic no Windows

Use estas informações ao considerar o uso de programas do Visual Basic no Windows

Nota: Versões de 64 bits dos arquivos de módulo Visual Basic não são fornecidas.

Para preparar programas do Visual Basic no Windows:

1. Crie um novo projeto.
2. Inclua o arquivo do módulo fornecido, CMQB.BAS, no projeto.
3. Inclua outros arquivos de módulo fornecidos se você precisar deles:

CMQBB.BAS	suporte de MQAI
CMQCFB.BAS	Suporte a PCF
CMQXB.BAS	Suporte a saídas do canal
CMQPSB.BAS	Publicação/assinatura

Consulte “Codificação em Visual Basic” na página 88 para obter informações sobre o uso da chamada MQCONNXAny a partir do Visual Basic.

Chame o procedimento MQ_SETDEFAULTS antes de fazer quaisquer chamadas MQI no código do projeto. Esse procedimento configura as estruturas padrão que as chamadas MQI requerem.

Especifique se você está criando um servidor ou cliente WebSphere MQ antes de compilar ou executar o projeto, configurando a variável de compilação condicional *MqType*. Configure *MqType* em um projeto do Visual Basic para 1 para um servidor ou 2 para um cliente como a seguir:

1. Selecione o menu Projeto.
2. Selecione *Name* Propriedades (em que *Name* é o nome do projeto atual).
3. Selecione a guia Fazer na caixa de diálogo.
4. No campo Argumentos de compilação condicional, insira isso para um servidor:

```
MqType=1
```

ou isso para um cliente:

```
MqType=2
```

Saída de segurança SSPI

WebSphere MQ para Windows fornece uma saída de segurança para o cliente MQI do WebSphere MQ e o servidor WebSphere MQ . Este é um programa de saída de canal que fornece autenticação para canais do WebSphere MQ usando a Security Services Programming Interface (SSPI). O SSPI fornece os recursos de segurança integrados de sistemas Windows

Os pacotes de segurança são carregados a partir de security.dll ou secur32.dll. Esses DLLs são fornecidos com o seu sistema operacional.

Autenticação de via única é fornecida usando os serviços de autenticação NTLM. Autenticação de duas vias é fornecida usando os serviços de autenticação do Kerberos.

O programa de saída de segurança é fornecido no formato de origem e de objeto. É possível usar o código de objeto no estado em que se encontra ou usar o código-fonte como um ponto de início para criar seus próprios programas de saída de usuário.

Consulte também “Usando a saída de segurança SSPI em sistemas Windows” na página 171.

Introdução a saídas de segurança

Uma saída de segurança forma uma conexão segura entre os programas de saída de segurança, onde um programa destina-se a enviar MCA (Agente do Canal de Mensagem), e outro a receber MCA.

O programa que inicia a conexão segura, ou seja, o primeiro programa a obter controle após a sessão do MCA ser estabelecida, é conhecido como o *inicializador de contexto*. O programa parceiro é conhecido como o *aceitador de contexto*.

A tabela a seguir mostra alguns dos tipos de canais que são inicializadores de contexto e seus aceitadores de contexto associados.

Tabela 69. Inicializadores de contexto e seus aceitadores de contexto associados

Inicializador de contexto	**Aceitador de contexto**
MQCHT_CLNTCONN	MQCHT_SVRCONN
MQCHT_RECEIVER	MQCHT_SENDER
MQCHT_CLUSRCVR	MQCHT_CLUSSDR

O programa de saída de segurança tem dois pontos de entrada:

- **SCY_NTLM**

  Usa serviços de autenticação NTLM que fornecem autenticação unilateral. NTLM permite que servidores verifiquem as identidades de seus clientes. Não permite que os clientes verifiquem a identidade de um servidor nem que um servidor verifique a identidade de outro. A autenticação NTLM foi projetada para um ambiente de rede no qual servidores são considerados como genuínos.

- **SCY_KERBEROS**

  Usa serviços de autenticação mútua do Kerberos. O protocolo Kerberos não supõe que os servidores em um ambiente de rede sejam autênticos. As partes em ambas as extremidades de uma conexão de rede podem verificar a identidade da outra parte. Ou seja, os servidores podem verificar a identidade de clientes e de outros servidores, e os clientes podem verificar a identidade de um servidor.

O que a saída de segurança faz

Este tópico descreve o que os programas de saída do canal SSPI fazem.

Os programas de saída do canal fornecidos fornecem autenticação unidirecional ou de duas vias (mútua) de um sistema parceiro quando uma sessão está sendo estabelecida. Para um canal específico, cada programa de saída tem um *diretor* associado (semelhante a um ID do usuário, consulte “WebSphere MQ controle de acesso e proprietários do Windows” na página 472). Uma conexão entre dois programas de saída é uma associação entre os dois diretores.

Após a sessão subjacente ser estabelecida, uma conexão segura entre os programas de saída de segurança (uma para o MCA de envio e uma para o MCA de recebimento) é estabelecida. A sequência de operações é a seguinte:

1. Cada programa está associado a um diretor específico, por exemplo, como resultado de uma operação de login explícito.
2. O inicializador de contexto solicita uma conexão segura com o parceiro a partir do pacote de segurança (para Kerberos, o parceiro denominado) e recebe um token (chamado token1). O token é enviado, usando a sessão subjacente já estabelecida, ao programa parceiro.
3. O programa parceiro (o aceitador de contexto) passa o token1 para o pacote de segurança, que verifica se o inicializador de contexto é autêntico. Para NTLM, a conexão agora está estabelecida.
4. Para a saída de segurança fornecida pelo Kerberos (ou seja, para autenticação mútua), o pacote de segurança também gera um segundo token (chamado token2), que o aceitador de contexto retorna ao inicializador de contexto usando a sessão subjacente.
5. O inicializador de contexto usa o token2 para verificar se o aceitador de contexto é autêntico.

6. Nesse estágio, se ambos os aplicativos estiverem satisfeitos com a autenticidade do token do parceiro, a conexão segura (autenticada) será estabelecida.

WebSphere MQ controle de acesso e proprietários do Windows

O controle de acesso fornecido pelo WebSphere MQ é baseado no usuário e no grupo. A autenticação que o Windows fornece é baseada em principais, como usuário e nome do servicePrincipal(SPN). No caso de servicePrincipalName, pode haver muitos desses associados a um único usuário.

A saída de segurança SSPI usa os principais do Windows relevantes para autenticação Se a autenticação do Windows for bem-sucedida, a saída passará o ID do usuário associado ao proprietário do Windows para WebSphere MQ para controle de acesso.

Os principais do Windows que são relevantes para autenticação variam, dependendo do tipo de autenticação usado..

- Para autenticação NTLM, o principal do Windows para o Inicializador de Contexto é o ID do usuário associado ao processo que está em execução. Como essa autenticação é unilateral, o diretor associado ao Aceitador de contexto é irrelevante.
- Para autenticação Kerberos , em canais CLNTCONN, o principal do Windows é o ID do usuário associado ao processo que está em execução. Caso contrário, o proprietário do Windows será o Nome do servicePrincipalformado pela inclusão do prefixo a seguir no Nome do QueueManager.

```
ibmMQSeries/
```

Usando serviços de protocolo de acesso de diretório leve com o WebSphere MQ para Windows

Este tópico explica o que é um serviço de diretório e o papel desempenhado por um DAP (Directory Access Protocol). Isso também explica como os aplicativos WebSphere MQ podem usar um diretório Lightweight Directory Access Protocol (LDAP) usando um programa de amostra como um guia...

Nota: O programa de amostra foi projetado para alguém que já está familiarizado com o LDAP

Os tópicos a seguir fornecem mais informações sobre serviços de diretório, LDAP e uso de LDAP com WebSphere MQ.

- “Serviço de Diretório” na página 472
- “Protocolo LDAP” na página 473
- “Usando LDAP com WebSphere MQ” na página 473

Serviço de Diretório

Um diretório é um repositório de informações sobre objetos, que é organizado de forma que seja fácil localizar as informações sobre um objeto específico.

Um exemplo comum é uma lista telefônica, onde informações (endereço e número de telefone) são armazenadas sobre pessoas e empresas. Outro exemplo é uma lista de endereços para um sistema de e-mail, em que os endereços de e-mail e, opcionalmente, outras informações, como números de telefone, são armazenados para pessoas

Em sistemas de computador, diretórios podem armazenar informações sobre recursos de computador, como impressoras ou discos compartilhados. Por exemplo, você poderia usar um diretório para descobrir onde a impressora colorida mais próxima está localizada. Em um aplicativo WebSphere MQ um diretório pode ser usado para fornecer a associação entre um serviço de aplicativo (como processamento de contas a receber) e a fila a ser usada para mensagens que requerem esse serviço (possivelmente identificado por meio do nome da fila e seu nome do gerenciador de filas do host).

Os diretórios são implementados como sistemas cliente-servidor, em que o servidor de diretório contém todas as informações e responde a solicitações de clientes Os clientes poderiam ser programas de

interface com o usuário, que fornecem as informações diretamente ao usuário, ou programas de aplicativos que precisam localizar recursos para concluir seu trabalho. Um Serviço de Diretório inclui o servidor de diretórios, os programas administrativos e as bibliotecas clientes e programas que são necessários para configurar, atualizar e ler o diretório.

Protocolo LDAP

Existem muitos serviços de diretório, como Novell Directory Services, DCE Cell Directory Service, Banyan StreetTalk, Windows Directory Services, X.500e os serviços de catálogo de endereço associados a produtos de e-mail X.500 foi proposto como um padrão para serviços de diretório global pela International Standards Organization (ISO). Ele requer uma pilha de protocolos OSI para suas comunicações, e em grande parte por isso, seu uso tem sido restrito a grandes organizações e instituições acadêmicas. Um servidor de diretório X.500 comunica-se com seus clientes usando o Directory Access Protocol (DAP)

LDAP (Lightweight Directory Access Protocol) foi criado como uma versão simplificada do DAP. É mais fácil implementar, omite alguns dos recursos menos usados do DAP e executa sobre TCP/IP. Como resultado dessas mudanças, ele está sendo rapidamente adotado como o protocolo de acesso de diretório para a maioria dos propósitos, substituindo a infinidade de protocolos proprietários usados anteriormente.. Os clientes LDAP ainda podem acessar um servidor X.500 por meio de um gateway (oX.500 ainda requer a pilha de protocolos OSI) ou, cada vez mais, as implementações X.500 geralmente incluem suporte nativo para LDAP, bem como acesso DAP.

Os diretórios LDAP podem ser distribuídos e podem usar a replicação para permitir o acesso eficiente ao seu conteúdo

Para obter uma descrição mais completa do LDAP, consulte *Entendendo LDAP*, uma publicação IBM Redbooks .

Usando LDAP com WebSphere MQ

Nas configurações do WebSphere MQ , as informações que definem filas de mensagens e de transmissão são armazenadas localmente. Isso significa que em uma rede do WebSphere MQ as várias definições são distribuídas, sem nenhum diretório central dessas informações estar disponível para navegação. O sistema de mensagens remoto entre aplicativos WebSphere MQ geralmente é obtido por meio do uso de definições locais de filas remotas. O aplicativo primeiro emite uma chamada MQOPEN usando o nome especificado na definição local da fila remota.. Para colocar a mensagem na fila remota, o aplicativo então emite MQPUT, especificando o identificador retornado da chamada MQOPEN. A definição de fila remota fornece o nome da fila de destino, o gerenciador da fila de destino e, opcionalmente uma fila de transmissão Nesta técnica, o aplicativo deve saber no tempo de execução o nome especificado na definição de fila local

Uma variação no anterior evita o uso de definições locais de filas remotas. O aplicativo pode especificar o nome completo da fila de destino, que inclui o nome do gerenciador de filas remotas como parte do MQOPEN O aplicativo deve, portanto, saber esses dois nomes no tempo de execução O gerenciador de fila local deve ser configurado corretamente com a definição de fila local e com uma fila de transmissão adequadamente nomeada (ou padrão) e um canal associado que entrega para o destino.

No caso em que os gerenciadores de filas de origem e de destino são definidos como membros do mesmo cluster, os aspectos da fila de transmissão e do canal dos dois cenários anteriores podem ser ignorados. Se a fila de transmissão de destino for uma fila de clusters, uma definição local de uma fila remota também não será necessária No entanto, de forma semelhante aos casos anteriores descritos, o aplicativo ainda deve saber o nome da fila de destino..

Um serviço de diretório pode ser usado para remover essa dependência do aplicativo em nomes de filas (ou a combinação de nomes de filas e gerenciadores de filas). O mapeamento entre os critérios do aplicativo e os nomes de objetos do WebSphere MQ pode ser mantido em um diretório e atualizado dinamicamente, e independentemente de aplicativos No tempo de execução, o aplicativo WebSphere MQ que deseja enviar uma mensagem primeiro consulta o diretório usando critérios baseados em aplicativo, por exemplo, em que: service_name = "accounts receiable", recupera os nomes de objetos relevantes do WebSphere MQ e, em seguida, usa esses valores retornados na chamada MQOPEN.

Outro exemplo do uso de um diretório é para uma empresa que possui muitos depósitos ou escritórios pequenos, os clientes MQI do WebSphere MQ podem ser usados para enviar mensagens para servidores WebSphere MQ localizados nos escritórios maiores. Os clientes precisam saber o nome da máquina host, canal MQI e nome da fila para cada servidor para o qual eles enviam mensagens. Ocasionalmente, pode ser necessário mover um servidor WebSphere MQ para outra máquina; cada cliente que se comunica com o servidor precisa saber sobre a mudança. Um serviço de diretório LDAP poderia ser usado para armazenar os nomes das máquinas host (e os nomes de canais e filas) e os programas clientes poderiam recuperar as informações do diretório sempre que desejassem enviar uma mensagem para um servidor. Nesse caso, apenas o diretório precisará ser atualizado se um nome do host (ou nome do canal ou da fila) for alterado...

Vários destinos para uma mensagem de aplicativo poderiam ser armazenados em um diretório, com o escolhido sendo dependente de disponibilidade ou considerações de compartilhamento de carregamento

WebSphere MQ também pode usar um diretório LDAP para armazenar informações de autenticação para uso com Secure Sockets Layer (SSL). WebSphere MQ classes para Java também podem armazenar informações em um diretório LDAP.

Programa de amostra LDAP

O programa de amostra foi projetado para alguém que está familiarizado com o LDAP e provavelmente já o usa. Ele deve mostrar como os aplicativos WebSphere MQ podem usar um diretório LDAP.

Construindo o programa de amostra

Esse programa foi construído e testado somente no Windows usando TCP/IP (Protocolo de Controle de Transmissões / Protocolo da Internet) Além das considerações gerais mencionadas em “Preparando programas C no Windows” na página 466, observe os pontos a seguir:

- Este programa é projetado para ser executado como um programa cliente, portanto, ele deve estar vinculado ao MQIC do MQIC.LIB .
- Assim como os arquivos de cabeçalho e bibliotecas do WebSphere MQ , este programa deve ser construído usando arquivos de cabeçalho e bibliotecas do cliente LDAP.

  Por exemplo, usando o cliente IBM eNetwork , vincule o programa ao LIBLDAPSTATICE.LIB e LIBLBERSTATICSSL.LIB .

configurando o diretório

Antes que o programa de amostra possa ser executado, um Servidor de Diretório LDAP deve ser configurado com dados de amostra

O arquivo MQuser.ldif, no diretório `tools\c\samples` , contém alguns dados de amostra em LDIF (LDAP Data Interchange Format). É possível editar esse arquivo para atender às suas necessidades. Ele contém dados para uma empresa fictícia chamada MQuser que tem um Departamento de Transporte composto por três escritórios. Cada um desses escritórios possui uma máquina que executa um servidor do WebSphere MQ

No mínimo, deve-se editar as três linhas que contêm os nomes de hosts das máquinas que executam os servidores WebSphere MQ : linhas 18, 27 e 36:

```
host: LondonHost
  ...
host: SydneyHost
  ...
host: WashingtonHost
```

Deve-se mudar `LondonHost`, `SydneyHoste` `WashingtonHost` para os nomes de três de suas máquinas que executam servidores WebSphere MQ Também é possível alterar os nomes de canal e de fila se você desejar (a amostra usa nomes dos padrões do sistema) Talvez você também queira aumentar ou diminuir o número de escritórios nos dados de amostra.

Configurando o servidor de diretório Tivoli IBM

Consulte o Guia do Administrador IBM Tivoli Directory Server (ITDS) para obter informações sobre como instalar o diretório. No tópico `Installing and Configuring Server`, trabalhe nas seções `Installing Server` e `Basic Server Configuration`. Se necessário, leia o tópico `Administrator Interface` para se familiarizar com o funcionamento da interface.

No tópico `Configuring - How Do I`, siga as instruções para iniciar o administrador, em seguida, trabalhe na seção `Configure Database` e crie um banco de dados padrão. Ignore a seção `Configure replica` e, usando a seção `Work with Suffixes`, inclua um sufixo o=`MQuser`

Antes de incluir quaisquer entradas no banco de dados, deve-se estender o esquema de diretório incluindo algumas definições de atributo e uma definição de classe de objetos Isso é descrito no Guia do Administrador do IBM Tivoli Directory Server no capítulo `Reference Information` sob a seção `Directory Schema` Dois arquivos de amostra estão incluídos para ajudá-lo com isso: O arquivo `mq.at.conf` inclui as definições de atributo que devem ser incluídas no arquivo `?etc?slapd.at.conf`.. Faça isso incluindo o arquivo de amostra editando `slapd.at.conf` e incluindo uma linha:

```
include <pathname>/mq.at.conf
```

Como alternativa, é possível editar o arquivo `slapd.at.conf` e incluir o conteúdo do arquivo de amostra diretamente nele, ou seja: inclua as linhas:

```
# MQ attribute definitions
attribute mqChannel           ces   mqChannel        1000   normal
attribute mqQueueManager      ces   mqQueueManager   1000   normal
attribute mqQueue             ces   mqQueue          1000   normal
attribute mqPort              cis   mqPort           64     normal
```

Da mesma forma, para a definição de classe de objeto, é possível incluir o arquivo de amostra editando `etc?slapd.oc.conf` e incluir a linha:

```
include <pathname>/mq.oc.conf
```

ou é possível incluir o conteúdo do arquivo de amostra diretamente em `slapd.oc.conf,` ou seja, incluir as linhas:

```
# MQ object classdefinition
objectclass mqApplication
      requires
              objectClass,
              cn,
              host,
              mqChannel,
              mqQueue
      allows
              mqQueueManager,
              mqPort,
              description,
              l,
              ou,
              seeAlso
```

Agora você pode iniciar o servidor de diretórios (Administração, Servidor, Inicialização) e adicionar as entradas de amostra a ele. Para incluir as entradas de amostra, acesse a página Administração, Incluir entradas do administrador, digite o nome do caminho completo do arquivo de amostra `MQuser.ldif` e clique em Enviar.

O servidor de diretório agora está em execução e carregado com dados adequados para a execução do programa de amostra

Configurando o servidor de diretório Netscape

Usando a página Administração do Netscape Server, clique em **Criar Novo Netscape Directory Server**.

Agora você deve receber um formulário contendo informações de configuração. Altere o Sufixo do Diretório para **o = MQuser** e inclua uma senha para o Usuário Irrestrito Também é possível alterar quaisquer outras informações para adequar sua instalação. Clique em **OK**, e o diretório deverá ser criado com êxito Clique em **Retornar para Administração do servidor** e inicie o servidor de diretório Clique no nome do diretório para iniciar o Directory Server Administration Server para o novo diretório.

Antes de incluir quaisquer entradas no banco de dados, estenda o esquema de diretório, incluindo algumas definições de atributo e uma definição de classe de objeto Clique na guia **Esquema** da página do Directory Server. Agora é apresentado um formulário que permite incluir novos atributos. Inclua os seguintes atributos (deixe o OID do Atributo em branco para todos eles):

```
Attribute Name              Syntax
--------------              ------
mqChannel                   Case Exact String
mqQueueManager              Case Exact String
mqQueue                     Case Exact String
mqPort                      Integer
```

Inclua um novo objectClass clicando em **Criar ObjectClass** no painel lateral.. Insira **mqApplication** como o ObjectClass Name, selecione **applicationProcess** como o pai ObjectClass e deixe o **ObjectClass OID** em branco. Agora, inclua alguns atributos na objectClass Selecione **host**, **mqChannel**e **mqQueue** como atributos necessários e selecione **mqQueueManager** e **mqPort** como atributos permitidos. Pressione o botão **Criar Novo ObjectClass** para criar o objectClass

Para incluir os dados de amostra, clique na guia **Gerenciamento de banco de dados** e selecione **Incluir entradas** no painel lateral.. Insira o nome do caminho do arquivo de dados de amostra `<pathname>\MQuser.ldif`, insira a senha e clique em **OK**

O programa de amostra é executado como um usuário não autorizado e, por padrão, o Diretório do Netscape não permite que usuários não autorizados procurem o diretório Altere isso clicando na guia **Controle de Acesso** Insira a senha para o Usuário Irrestrito e clique em **OK** para carregar nas entradas de controle de acesso para o diretório Elas devem estar vazias no momento Pressione o botão **Nova ACI** para criar uma nova entrada de controle de acesso Na caixa de entrada exibida, clique em **Negar** (que está sublinhado) e, na caixa de diálogo resultante, altere-a para **Permitir**. Inclua um nome, por exemplo, **MQuser-access**e clique em **escolher um sufixo** para selecionar **o = MQuser**. Insira **o = MQuser** como o destino, insira a senha para o Usuário Irrestrito e clique em **Enviar**.

O servidor de diretório agora está em execução e carregado com dados adequados para a execução do programa de amostra

Executando o programa de amostra.

Agora você deve ter um LDAP Directory Server em execução e preenchido com os dados de amostra. Os dados especificam três máquinas host, todas elas devem estar executando servidores WebSphere MQ . Assegure que o gerenciador de filas padrão esteja em execução em cada máquina (a menos que você tenha alterado os dados de amostra para especificar um gerenciador de filas diferente).

Além disso, inicie o programa listener do WebSphere MQ em cada máquina; a amostra usa TCP/IP com o número da porta WebSphere MQ padrão, para que seja possível iniciar o listener com o comando:

```
runmqlsr -t tcp
```

Para testar a amostra, você também pode desejar executar um programa para ler as mensagens recebidas em cada servidor WebSphere MQ , por exemplo, você poderia usar o programa de amostra amqstrg:

```
amqstrg SYSTEM.DEFAULT.LOCAL.QUEUE
```

O programa de amostra usa três variáveis de ambiente, uma necessária e duas opcionais.. A variável necessária é LDAP_BASEDN, que especifica o Nome Distinto base para a procura de diretório. Para trabalhar com os dados de amostra, configure como `ou=Transport, o=MQuser`, por exemplo, em um prompt de comandos no tipo de sistema Windows :

```
set LDAP_BASEDN=ou=Transport, o=MQuser
```

As variáveis opcionais são LDAP_HOST e LDAP_VERSION. A variável LDAP_HOST especifica o nome do host no qual o servidor LDAP está em execução; ele é padronizado para o host local se não for especificado. A variável LDAP_VERSION especifica a versão do protocolo LDAP a ser usado e pode ser 2 ou 3. A maioria dos servidores LDAP agora suporta a versão 3 do protocolo; todos eles suportam a versão mais antiga 2. Essa amostra funciona igualmente bem com qualquer versão do protocolo e, se não for especificada, será padronizada para a versão 2.

Agora é possível executar a amostra, digitando o nome do programa seguido pelo nome do aplicativo WebSphere MQ para o qual você deseja enviar mensagens, no caso dos dados de amostra, os nomes do aplicativo são Londres, Sydney e Washington Por exemplo, para enviar mensagens para o aplicativo London:

```
amqsldpc London
```

Se o programa falhar ao conectar-se ao servidor WebSphere MQ , uma mensagem de erro apropriada será exibida Se ele se conectar com êxito, será possível iniciar a digitação de mensagens, cada linha digitada (finalizada por < return> ou < enter>) será enviada como uma mensagem separada, uma linha vazia encerrará o programa.

Design do programa.

O programa possui duas partes distintas: a primeira parte usa as variáveis de ambiente e o valor da linha de comandos para consultar um servidor de diretórios LDAP; a segunda parte estabelece a conexão do WebSphere MQ usando as informações retornadas do diretório e envia as mensagens...

As chamadas LDAP usadas na primeira parte do programa diferem um pouco dependendo se o LDAP versão 2 ou 3 está sendo usado e são descritas em detalhes pela documentação fornecida com as bibliotecas do cliente LDAP. Esta seção fornece uma breve descrição

A primeira parte do programa verifica se ele foi chamado corretamente e lê as variáveis de ambiente Em seguida, estabelece uma conexão com o servidor de diretório LDAP no host especificado:

```
if (ldapVersion == LDAP_VERSION3)
{
  if ((ld = ldap_init(ldapHost, LDAP_PORT)) == NULL)
     ...
}
else
{
  if ((ld = ldap_open(ldapHost, LDAP_PORT)) == NULL )
     ...
}
```

Quando uma conexão tiver sido estabelecida, o programa configura algumas opções no servidor com a chamada "ldap_set_option" e, em seguida, autentica-se para o servidor ligando-se a ele:

```
if (ldapVersion == LDAP_VERSION3)
{
  if (ldap_simple_bind_s(ld, bindDN, password) != LDAP_SUCCESS)
     ...
}
else
{
  if (ldap_bind_s(ld, bindDN, password, LDAP_AUTH_SIMPLE) !=
      LDAP_SUCCESS)
     ...
}
```

No programa de amostra, `bindDN` e `password` são configurados como NULL, o que significa que o programa se autentica como um usuário anônimo, ou seja, ele não possui nenhum direito de acesso especial e pode acessar apenas informações que estão disponíveis publicamente Na prática, a maioria das organizações restringem o acesso às informações que armazenam em diretórios para que apenas usuários autorizados possam acessá-las..

O primeiro parâmetro para a chamada de ligação `ld` é um identificador que é usado para identificar essa sessão LDAP específica em todo o restante do programa Após a autenticação, o programa procura no diretório entradas que correspondam ao nome do aplicativo:

```
rc = ldap_search_s(ld,                /* LDAP Handle             */
                   baseDN,            /* base distinguished name */
                   LDAP_SCOPE_ONELEVEL, /* one-level search       */
                   filterPattern,     /* filter search pattern   */
                   attrs,             /* attributes required     */
                   FALSE,             /* NOT attributes only     */
                   &ldapResult);      /* search result           */
```

Esta é uma chamada síncrona simples para o servidor que retorna os resultados diretamente Há outros tipos de procura mais apropriados para consultas complexas ou quando um grande número de resultados é esperado. O primeiro parâmetro da procura é o identificador `ld` que identifica a sessão. O segundo parâmetro é o nome distinto de base, que especifica onde no diretório a procura deve começar e o terceiro parâmetro é o escopo da procura, ou seja, quais entradas relativas ao ponto inicial são procuradas. Esses dois parâmetros juntos definem quais entradas no diretório são procuradas.. O próximo parâmetro, `filterPattern` especifica o que estamos procurando. O parâmetro `attrs` lista os atributos que queremos obter de volta do objeto quando o localizamos O próximo atributo diz se queremos apenas os atributos ou seus valores também; configurar isso como FALSE significa que queremos os valores de atributos.. O parâmetro final é usado para retornar o resultado

O resultado poderia conter muitas entradas de diretórios, cada uma com os atributos especificados e os seus valores Temos de extrair os valores que queremos do resultado. Neste programa de amostra, esperamos que apenas uma entrada seja localizada, portanto, olhamos apenas para a primeira entrada no resultado:

```
ldapEntry = ldap_first_entry(ld, ldapResult);
```

Essa chamada retorna um identificador que representa a primeira entrada e configuramos um loop for para extrair todos os atributos da entrada:

```
for (attribute = ldap_first_attribute(ld, ldapEntry, &ber);
     attribute != NULL;
     attribute = ldap_next_attribute(ld, ldapEntry, ber ))
{
```

Para cada um desses atributos, extraímos os valores associados a ele. Novamente, esperamos apenas um valor por atributo, portanto, usamos apenas o primeiro valor; determinamos qual atributo temos e armazenamos o valor na variável do programa apropriada:

```
values = ldap_get_values(ld, ldapEntry, attribute);
if (values != NULL && values[0] != NULL)
{
  if (stricmp(attribute, MQ_HOST_ATTR) == 0)
  {
    mqHost = strdup(values[0]);
    ...
```

Finalmente, nós arrumamos liberando a memória (ldap_value_free, ldap_memfree, ldap_msgfree) e fechamos a sessão *desligando* do servidor:

```
ldap_unbind(ld);
```

Verificamos que localizamos todos os valores do WebSphere MQ que precisamos do diretório e, em caso afirmativo, chamamos sendMessages() para se conectar ao servidor WebSphere MQ e enviar as mensagens do WebSphere MQ .

A segunda parte do programa de amostra é a rotina sendMessages() que contém todas as chamadas do WebSphere MQ . Isso é modelado no programa de amostra amqsput0 , as diferenças sendo que os parâmetros para o programa foram estendidos e MQCONNX é usado em vez da chamada MQCONN.

Desenvolvendo aplicativos para o IBM WebSphere MQ Telemetry

aplicativos de telemetria integram dispositivos de sentido e controle a outras fontes de informações disponíveis na Internet e nas empresas.

Desenvolva aplicativos para o IBM WebSphere MQ Telemetry usando padrões, exemplos trabalhados, programas de amostra, conceitos de programação e informações de referência. Use o daemon do IBM WebSphere MQ Telemetry para dispositivos para simplificar a conexão de vários pequenos dispositivos ao IBM WebSphere MQ.

Conceitos relacionados

WebSphere MQ Telemetry

Conceitos e Cenários de Telemetria para Monitoramento e Controle

Tarefas relacionadas

Instalando o WebSphere MQ Telemetry

Administrando o WebSphere MQ Telemetry

Resolução de problemas para WebSphere MQ Telemetry

Referências relacionadas

Referência do WebSphere MQ Telemetry

IBM WebSphere MQ Telemetry programas de amostra

Scripts de amostra são fornecidos para demonstrar o uso básico do aplicativo cliente MQ Telemetry Transport v3 . Use os scripts para publicar uma mensagem e assinar um tópico.

Antes de começar

Inicie o serviço de telemetria (MQXR) para executar os programas de amostra

O ID do usuário deve ser um membro do grupo de usuários mqm.

Execute o script SampleMQM primeiro, seguido pelo script MQTTV3Sample para executar uma publicação e assinatura.. Execute o script de amostra CleanupMQM para excluir o gerenciador de filas criado pelo script SampleMQM ..

Como o script SampleMQM cria e usa um gerenciador de filas chamado QM1, não execute inalterado em um sistema com um gerenciador de filas QM1 . Quaisquer mudanças feitas podem ter implicações na configuração do gerenciador de filas existente.

Sobre esta tarefa

- O aplicativo SampleMQM cria e inicia um gerenciador de filas ativado por telemetria chamado QM1. O script também configura uma fila de transmissão padrão para QM1e cria e inicia um canal padrão atendendo na porta 1883. Esse canal não executa autenticação de clientes conectados a ele. O canal possui o atributo identificador de usuário do agente do canal de mensagens (MCAUSER), configurado como 'guest' nos sistemas Windows ou 'nobody' nos sistemas Linux . Os clientes conectados ao canal são tratados como o usuário 'guest' ou o usuário 'nobody', dependendo do sistema no qual ele está em execução O script autoriza 'guest' em sistemas Windows e 'nobody ' em sistemas Linux para poder publicar e assinar qualquer tópico em QM1
- O aplicativo MQTTV3Sample é mantido no seguinte local;
  - No Windows *MQ_INSTALLATION_PATH*\mqxr\samples

em que *MQ_INSTALLATION_PATH* é o local no qual o IBM WebSphere MQ é instalado.

– No Linux *MQ_INSTALLATION_PATH*/mqxr/samples

O aplicativo MQTTV3Sample funciona como um publicador, enviando uma única mensagem para um tópico no servidor.. Ele também age como um assinante, ouvindo mensagens do servidor.

- O script de amostra CleanupMQM termina e exclui QM1 que foi criado pelo script SampleMQM . Use o script de amostra CleanupMQM se desejar executar novamente o script SampleMQM ou remover QM1.

Procedimento

1. Digite o seguinte comando em uma linha de comandos para executar o script SampleMQM
   - No Windows, o comando para executar o script SampleMQM é o seguinte:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\SampleMQM.bat
     ```

   - No AIX e Linux, o comando para executar o script SampleMQM é o seguinte:

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/SampleMQM.sh
     ```

   em que *MQ_INSTALLATION_PATH* é o local no qual o IBM WebSphere MQ é instalado.

   Um gerenciador de filas chamado MQXR_SAMPLE_QM é criado.
2. Digite o seguinte comando para executar a primeira parte do script MQTTV3Sample;
   - No Windows, em uma linha de comandos, digite o comando a seguir:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\RunMQTTV3Sample.bat -a subscribe
     ```

   - Em AIX e Linux, em uma janela shell, digite o comando a seguir;

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh -a subscribe
     ```

3. Digite o seguinte comando para executar a segunda parte do script MQTTV3Sample;
   - No Windows, em outra linha de comandos, digite o comando a seguir:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\RunMQTTV3Sample.bat -m "Hello from an MQTT v3
     application"
     ```

   - Em AIX e Linux, em outra janela shell, digite o comando a seguir;

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh -m "Hello from an MQTT v3 application"
     ```

4. Para remover o gerenciador de filas criado pelo script SampleMQM, é possível executar o script CleanupMQM usando o seguinte comando;
   - No Windows, digite o comando a seguir:

     ```
     MQ_INSTALLATION_PATH\mqxr\samples\CleanupMQM.bat
     ```

   - Em AIX e Linux em outra janela shell, digite o comando a seguir;

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/CleanupMQM.sh
     ```

Resultados

A mensagem `Hello from an MQTT v3 application` , digitada na segunda janela, será publicada por esse aplicativo e recebida pelo aplicativo na primeira janela. O aplicativo na primeira janela o mostrará na tela.

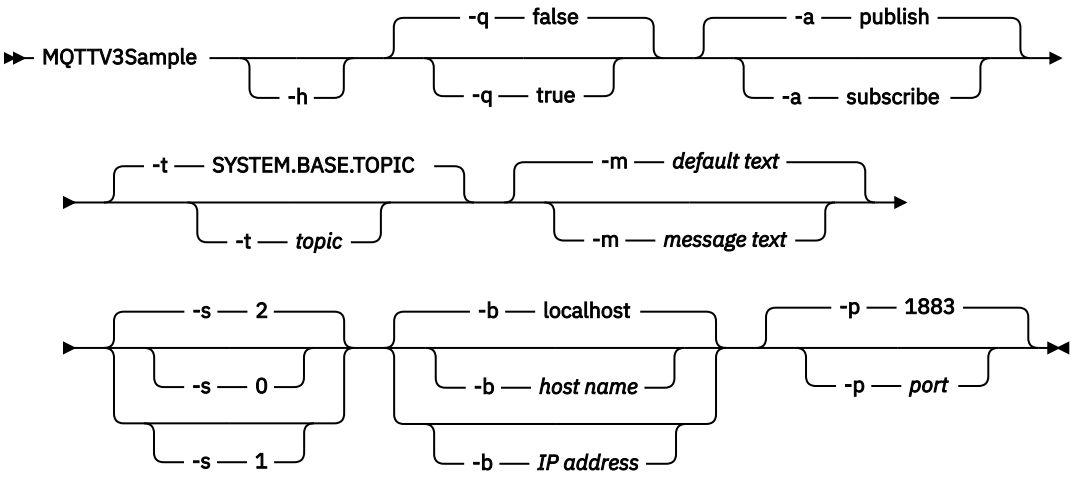
Programa MQTTV3Sample

Informações de referência sobre sintaxe de amostra e parâmetros para o programa MQTTV3Sample .

Finalidade

O programa MQTTV3Sample pode ser usado para publicar uma mensagem e assinar um tópico.

MQTTV3Sample syntax



Parâmetros

-h
Imprimir este texto de ajuda e sair

-q
Configure o modo silencioso em vez de usar o modo false padrão.

-a
Configure publicar ou assinar em vez de assumir a ação padrão de publicação.

-t
Publique ou assine o tópico em vez de publicar ou assinar o tópico padrão

-m
Publique o texto da mensagem em vez de enviar o texto de publicação padrão, "Hello de um aplicativo MQTT v3 ".

-s
Configure o QoS em vez de usar o QoS padrão, 2.

-b
Conectar a esse nome do host ou endereço IP em vez de conectar ao nome do host padrão, host local.

-p
Use essa porta em vez de usar o padrão, 1883.

Execute o programa MQTTV3Sample

Para assinar um tópico no Windows, use o comando:

```
runMQTTV3Sample -a subscribe
```

Para publicar uma mensagem no Windows, use o comando:

```
runMQTTV3Sample
```

Para obter informações adicionais sobre como executar os scripts de amostra fornecidos, consulte “IBM WebSphere MQ Telemetry programas de amostra” na página 479.

Criando seu primeiro aplicativo publicador do MQ Telemetry Transport usando Java

As etapas para criar um aplicativo cliente MQTT são descritas no modo de tutorial Cada linha de código é explicada. No final da tarefa, você terá criado um editor MQTT. É possível procurar as publicações usando o WebSphere MQ Explorer.

Antes de começar

Instale o recurso de Telemetria do WebSphere MQ em um servidor que tenha o IBM WebSphere MQ Version 7.1 ou posterior instalado

O aplicativo cliente usa o pacote `com.ibm.mq.micro.client.mqttv3` no IBM WebSphere MQ Telemetry Software Development Toolkit (SDK). O SDK faz parte da instalação do IBM WebSphere MQ Telemetry O cliente se conecta ao recurso IBM WebSphere MQ Telemetry para trocar mensagens com IBM WebSphere MQ.

Você também deve instalar as atualizações de telemetria para IBM WebSphere MQ Explorer Version 7.1 para administrar IBM WebSphere MQ Telemetry. As atualizações fazem parte da instalação do IBM WebSphere MQ Telemetry

Um cliente MQTT, em execução no Java SE, requer a versão 6.0 do Java SE ou posterior. IBM Java SE v6.0 faz parte da instalação do IBM WebSphere MQ Version 7.1 . Ele está localizado em *`WebSphere MQ installation directory`*`\java\jre`

Sobre esta tarefa

O exemplo é um aplicativo de publicação, `PubSync`. `PubSync` publica `Hello World` no tópico `MQTT Examples`e aguarda confirmação de que a publicação foi entregue ao gerenciador de filas.

Configurando uma assinatura durável para `MQTT Examples` , é possível verificar se o aplicativo funciona.

O procedimento usa Eclipse para desenvolver, construir e executar o cliente. É possível fazer download do Eclipse a partir do website do projeto Eclipse em www.eclipse.org.

Para criar o aplicativo, é possível criar os arquivos Java e compilar e executá-los usando a linha de comandos.

Em um novo diretório, crie o caminho do diretório `.\com\ibm\mq\id`. Crie dois arquivos Java, `Example.java` e `PubSync.java` Copie o código de “Código de exemplo” na página 486 para os arquivos Java.

Compile o código Java usando o comando,

```
javac -cp jar_dir\com.mq.micro.client.mqttv3.jar
       com.ibm.mq.id.PubSync.java com.ibm.mq.id.Example.java
```

Execute `PubSync` usando o comando,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubSync
```

Procedimento

1. Crie um projeto Java em Eclipse
   a) **Arquivo> Novo> Projeto Java** e digite um nome do projeto.. Clique em **Avançar**.

      Verifique se o JRE está na versão correta ou mais recente. Java SE deve estar em 6.0 ou posterior.
   b) Na página Configurações Java, clique em **Bibliotecas> Incluir Jars Externos ...**
   c) Navegue até o diretório no qual você instalou a pasta do SDK de Telemetria do WebSphere MQ . Localize a pasta SDK`\clients\java` e selecione todos os arquivos `.jar` > **> Abrir > Concluir**.
2. Instale o Javadoc do cliente do MQTT

Com o Javadoc do cliente MQTT instalado, o editor Java fornece assistência com as classes MQTT v3 .

a) Em seu projeto Java, abra **Package Explorer> Bibliotecas Referidas**. Clique com o botão direito em `com.ibm.micro.client.mqttv3.jar` **> Propriedades**.

b) No navegador Propriedades, clique em **Local do Javadoc**.

c) Na página Local do Javadoc, clique em **URL do Javadoc> Procurar ...** e localizar a *WMQ Installation directory*`\mqxr\SDK\clients\java\doc\javadoc` pasta **> OK**.

d) Clique em **Validar ...> OK**

Você recebeu um aviso para abrir um navegador para visualizar a documentação.

3. Crie a classe, PubSync, usando o assistente de Classe Java

a) Clique com o botão direito no projeto Java que você criou> **Novo** > **Classe**.

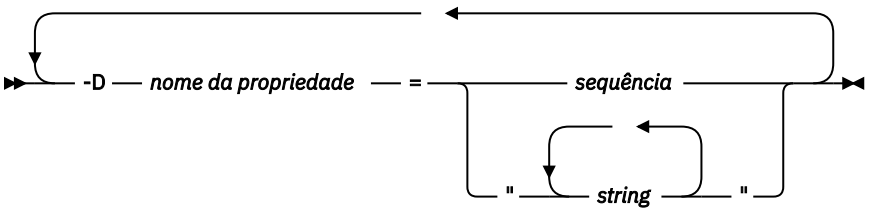
b) Digite o nome do pacote, `com.ibm.mq.id`

c) Digite o nome da classe, PubSync

d) Marque a caixa do stub de método, **public static void main(String [] args)**

4. Crie um arquivo, `Example.java` no pacote `com.ibm.mq.id`. Copie o código de Figura 89 na página 487 no arquivo.

Todos os parâmetros usados nos exemplos são configurados como propriedades. É possível substituir os valores mudando os padrões em `Example.java`ou fornecendo as propriedades como opções na linha de comandos Java usando o parâmetro `-D` :



O identificador de cliente usado nesse exemplo e os exemplos de “Criando um publicador assíncrono para o MQ Telemetry Transport usando Java” na página 487, é um nome de usuário sufixado com uma sequência aleatória.

5. Siga as etapas para criar o código ou copie o código de Figura 88 na página 486.

As etapas a seguir explicam o código em `Pubsync.java`.

6. Crie um bloco `try-catch`.

```
try { ...
 } catch (Exception e) {
  e.printStackTrace();
}
```

O cliente MQTT lança `MqttException`, `MqttPersistenceException` ou `MqttSecurityException`.. `MqttPersistenceException` e `MqttSecurityException` são subclasses de `MqttException`.

Use o método `MqttException.getReasonCode` para descobrir a razão da exceção. Se `MqttPersistenceException` ou `MqttSecurityException` for lançado, use o método `getCause` para retornar a exceção lançável subjacente.

7. Crie uma nova instância de `MqttClient`.

```
MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
```

Forneça ao cliente um endereço do servidor, que será usado posteriormente para se conectar ao WebSphere MQ. Configure o identificador de cliente para o nome do cliente.

- Opcionalmente, é possível fornecer uma implementação da interface `MqttClientPersistence` para substituir a implementação padrão. A implementação padrão do `MqttPersistence` armazena QoS 1 e 2 mensagens aguardando entrega como arquivos; consulte “Persistência de Mensagem em Clientes MQTT” na página 541.
- A porta TCP/IP padrão IBM WebSphere MQ para MQTT é 1883. Para SSL, ela é 8883. No exemplo, o endereço padrão é configurado como `tcp://localhost:1883`.
- Normalmente, é importante poder identificar um cliente físico específico usando o identificador de cliente. O identificador de cliente deve ser exclusivo em todos os clientes conectados a um servidor; consulte “Identificador de Cliente” na página 537. O uso do mesmo identificador de cliente que uma instância anterior indica que a instância presente é uma instância do mesmo cliente. Se você duplicar um identificador de cliente nos dois clientes em execução, uma exceção será lançada nos dois clientes e um deles será finalizado.
- O comprimento do identificador de cliente é limitado a 23 bytes. Uma exceção é lançada se o comprimento for excedido. O identificador de cliente deve conter apenas caracteres permitidos em um nome do gerenciador de filas; por exemplo, sem hifens ou espaços.
- Até você chamar o método `MqttClient.connect`, nenhum processamento de mensagem ocorre.

Use o objeto do cliente para publicar e assinar tópicos e recuperar informações sobre publicações que ainda não foram entregues.

8. Crie um tópico para publicar nele.

```
MqttTopic topic = client.getTopic(Example.topicString);
```

Uma sequência de tópicos é limitada a 64 Kbytes, o que excede o comprimento máximo de uma sequência de tópicos do IBM WebSphere MQ. Caso contrário, uma sequência de tópicos seguirá as mesmas regras que WebSphere MQ sequências de tópicos; consulte Sequências de tópicos. O exemplo configura uma sequência de tópicos `MQTT Examples`.

9. Crie uma mensagem de publicação.

```
MqttMessage message = new MqttMessage(Example.publication.getBytes());
```

A sequência `"Hello World"` é convertida em uma matriz de bytes e usada para criar um `MqttMessage`

- Uma carga útil da mensagem do MQTT é sempre uma matriz de bytes O método `getBytes` converte um objeto de sequência em UTF-8. `MqttMessage` tem um método `toString` de conveniência para retornar a carga útil da mensagem como uma sequência. É equivalente a `new string(message.getPayload)`
- Uma mensagem de publicação é enviada para o gerenciador de filas com um cabeçalho RFH2 e os dados da mensagem são enviados como uma mensagem `jms-bytes`.
- O objeto de mensagem tem qualidade de serviço e atributos retidos. A qualidade de serviço (QoS) determina quão confiável a mensagem é transferida entre o cliente MQTT e o gerenciador de filas; consulte “Qualidades de serviço fornecidas por um cliente MQTT” na página 545. O atributo retido controla se uma publicação é armazenada pelo gerenciador de filas para futuros assinantes. Se uma publicação não for retida, ela será enviada apenas para assinantes atuais; consulte “Publicações Retidas e Clientes MQTT” na página 547. As configurações `MqttMessage` padrão são "As mensagens são entregues pelo menos uma vez e não ficam retidas."

10. Conecte-se ao servidor.

```
client.connect();
```

O exemplo se conecta ao servidor usando as opções de conexão padrão. Após a conexão, é possível iniciar a publicação. As opções de conexão padrão são:

- Uma pequena mensagem "keep-alive" é enviada a cada 15 segundos para evitar que a conexão TCP/IP seja encerrada.
- A sessão é iniciada sem verificar a conclusão das publicações anteriores.

- O intervalo entre tentativas de enviar uma mensagem novamente é 15 segundos.
- Nenhuma mensagem de último desejo e testamento é criada para a conexão.
- O SocketFactory padrão é usado para criar a conexão.

Mude as opções de conexão criando um objeto `ConnectionOptions` e transmitindo-o como um parâmetro adicional para `client.connect`.

11. Publique.

```
MqttDeliveryToken token = topic.publish(message);
```

O exemplo envia a publicação "Hello World" no tópico "Exemplos MQTT" para o gerenciador de filas.

- Quando o método `publish` for retornado, a mensagem será transferida com segurança para o cliente MQTT, mas ainda não será transferida para o servidor Se a mensagem tiver QoS 1 ou 2, a mensagem será armazenada localmente, caso o cliente falhe antes de a entrega ser concluída.
- `publish` retorna um token de entrega, que é usado para verificar se uma confirmação já foi recebida do servidor.

12. Aguarde a confirmação do servidor.

```
token.waitForCompletion(Example.timeout);
```

O exemplo PubSync aguarda uma confirmação do servidor, que confirma se a mensagem foi entregue.

- Sem o tempo limite, o cliente aguardaria por tempo indeterminado. A tarefa “Criando um publicador assíncrono para o MQ Telemetry Transport usando Java” na página 487 mostra como receber confirmações sem aguardar usando um objeto de retorno de chamada.

13. Desconecte o cliente do servidor.

```
client.disconnect();
```

O cliente se desconecta do servidor e aguarda a conclusão de quaisquer métodos `MqttCallback` que estejam em execução. Em seguida, aguarda até 30 segundos para concluir qualquer trabalho restante. É possível especificar um tempo limite de quiesce como um parâmetro adicional.

14. Salvar mudanças em `PubSync.java` e `Example.Java`

O Eclipse compila automaticamente o Java Você está pronto para ver os resultados executando o programa.

Resultados

Para ver as publicações usando WebSphere MQ, crie um tópico, uma fila e uma assinatura durável, todos chamados `"MQTTExampleTopic"` usando o script em Figura 87 na página 485. Execute o cliente para publicar no tópico `MQTT Examples` e, em seguida, execute o programa de amostra **amqsbcg** para procurar as publicações na fila `MQTTExamples` .

1. Inicie um gerenciador de filas e inicie a execução do seu serviço de telemetria (MQXR). Certifique-se de que o endereço TCP/IP e a porta configurados para o canal de telemetria correspondam aos valores usados no aplicativo MQTT.
2. Configure uma assinatura durável criando o script de comando `mqttexamples.txt` e executando-o usando **runmqsc:**

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Figura 87. `mqttExampleTopic.txt`

Para executar o script no Windows, digite o comando:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Execute o cliente como um aplicativo Java a partir do Eclipseou executando Java em uma janela de comando:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

Nota: A janela de comando deve ser aberta no diretório que contém o caminho `com\ibm\mq\id`.

4. Navegue pelos resultados usando o WebSphere MQ Explorer ou execute o comando:

```
amqsbcg MQTTExampleQueue queue manager name
```

Código de exemplo

PubSync.java é uma listagem completa do código descrito em Procedimento. Modifique a classe `Example` em Figura 89 na página 487 para substituir os parâmetros padrão usados em PubSync.java.

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubSync {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      client.connect();
      System.out.println("Waiting for up to " + Example.sleepTimeout / 1000
          + " seconds for publication of \"" + message.toString()
          + "\" with QoS = " + message.getQos());
      System.out.println("On topic \"" + topic.getName()
          + "\" for client instance: \"" + client.getClientId()
          + "\" on address " + client.getServerURI() + "\"");
      MqttDeliveryToken token = topic.publish(message);
      token.waitForCompletion(Example.sleepTimeout);
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
      client.disconnect();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Figura 88. PubSync.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Figura 89. Example.java

Conceitos relacionados

Aplicativos de Publicação/Assinatura MQTT

Criando um publicador assíncrono para o MQ Telemetry Transport usando Java

Nesta tarefa, você segue um tutorial para modificar seu primeiro aplicativo de publicador. As modificações permitem que o aplicativo envie publicações sem esperar reconhecimentos de entrega. Os reconhecimentos de entrega são recebidos por uma classe de retorno de chamada que você cria.

Antes de começar

Instale o recurso de Telemetria do WebSphere MQ em um servidor que tenha o IBM WebSphere MQ Version 7.1 ou posterior instalado

O aplicativo cliente usa o pacote `com.ibm.mq.micro.client.mqttv3` no IBM WebSphere MQ Telemetry Software Development Toolkit (SDK). O SDK faz parte da instalação do IBM WebSphere MQ Telemetry O cliente se conecta ao recurso IBM WebSphere MQ Telemetry para trocar mensagens com IBM WebSphere MQ.

Você também deve instalar as atualizações de telemetria para IBM WebSphere MQ Explorer Version 7.1 para administrar IBM WebSphere MQ Telemetry. As atualizações fazem parte da instalação do IBM WebSphere MQ Telemetry

Um cliente MQTT, em execução no Java SE, requer a versão 6.0 do Java SE ou posterior. IBM Java SE v6.0 faz parte da instalação do IBM WebSphere MQ Version 7.1 . Ele está localizado em *WebSphere MQ installation directory*\java\jre

Sobre esta tarefa

O exemplo é um aplicativo de publicação, PubAsync. PubAsync publica `Hello World` no tópico `MQTT Examples`, sem aguardar confirmação de que a publicação foi entregue ao gerenciador de filas. Os reconhecimentos de entrega são recebidos em uma classe de retorno de chamada, `CallBack`.

Configurando uma assinatura durável para `MQTT Examples` , é possível verificar se o aplicativo funciona.

O procedimento usa Eclipse para desenvolver, construir e executar o cliente. É possível fazer download do Eclipse a partir do website do projeto Eclipse em www.eclipse.org.

As etapas em Procedimento modificam o aplicativo PubSync`.java` em "Criando seu primeiro aplicativo publicador do MQ Telemetry Transport usando Java" na página 482.

Como alternativa, é possível copiar o código, "Código de exemplo" na página 490, para um novo diretório `.\com\ibm\mq\id`. Crie três arquivos Java, `Example.java`, `CallBack.javae` PubAsync`.java`.. Compile os exemplos usando o comando,

```
javac -cp jar_dir\com.mq.micro.client.mqttv3.jar
       com.ibm.mq.id.PubAsync.java com.ibm.mq.id.CallBack.java com.ibm.mq.id.Example.java
```

Execute PubAsync usando o comando,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubAsync.class
```

Procedimento

1. No pacote `com.ibm.mq.id`, crie um arquivo, `CallBack.java`. Copie o código de Figura 92 na página 491 no arquivo.

   `Callback.java` implementa a interface `MqttCallBack`. No exemplo, um construtor adicional inicializa o retorno de chamada com alguns dados da instância.
2. No pacote `com.ibm.mq.id`, clique com o botão direito em PubSync`.java` e copie-o. Cole-o no mesmo pacote, renomeando-o para PubAsync.
3. Imediatamente antes da linha de código `client.connect();`, instancie a classe `CallBack`, passando o identificador de cliente.

   ```
   CallBack callback = new CallBack(Example.clientId);
   client.setCallback(callback);
   ```

   - A classe `CallBack` implementa `MqttCallBack`. É necessária uma instância de retorno por identificador de cliente. Nesse exemplo, o construtor passa o identificador de cliente para salvar como dados da instância. Ele é usado no retorno de chamada para identificar qual instância do retorno de chamada foi iniciada.
   - Deve-se implementar três métodos na classe de retorno de chamada:

     `public void messageArrived(MqttTopic topic, MqttMessage message)`
     Recebe uma publicação que foi assinada.

     `public void connectionLost(Throwable cause)`
     Chamado quando a conexão é perdida.

public void deliveryComplete(MqttDeliveryToken token))
Chamado quando um token de entrega é recebido para uma mensagem de QoS 1 ou 2 que foi publicada.

- O retorno de chamada é ativado por `MqttClient.connect`.

4. Desconectar o cliente
   a) Remova a instrução que contém a expressão `token.waitForCompletion`.
   O encadeamento principal continua sem esperar que a publicação seja entregue.
   b) Teste se o cliente já está desconectado.
   O cliente MQTT desconecta-se após um erro retornado ao método `lostConnection` em `MqttCallback`ou o aplicativo cliente pode desconectar. Teste para ver se há uma conexão aberta.
   c) Use a constante, `Example.quiesceTimeout`, para configurar o tempo máximo de quiesce do cliente.

```
if (client.isConnected())
    client.disconnect(Example.quiesceTimeout);
```

O cliente é concluído quando uma combinação das três condições a seguir é verdadeira:

a. O retorno de chamada foi chamado para todas as mensagens publicadas nesta sessão ou se a sessão foi reiniciada em sessões anteriores.
b. Mensagens estão em andamento e o intervalo de quiesce expirou. Por padrão, o intervalo de quiesce é de 30 segundos. É possível mudar o tempo limite de quiesce passando o número de milissegundos a esperar como um parâmetro de `client.disconnect`.
c. `client.disconnect` foi chamado após algumas mensagens serem publicadas e enfileiradas pelo cliente, mas antes que as mensagens fossem enviadas. Mensagens enfileiradas ainda não estão em andamento. Se a sessão for reinicializável, as mensagens serão reenviadas quando a sessão for reiniciada.

Resultados

Para ver as publicações usando WebSphere MQ, crie um tópico, uma fila e uma assinatura durável, todos chamados `"MQTTExampleTopic"` usando o script em Figura 90 na página 489. Execute o cliente para publicar no tópico `MQTT Examples` e, em seguida, execute o programa de amostra **amqsbcg** para procurar as publicações na fila `MQTTExamples` .

1. Inicie um gerenciador de filas e inicie a execução do seu serviço de telemetria (MQXR). Certifique-se de que o endereço TCP/IP e a porta configurados para o canal de telemetria correspondam aos valores usados no aplicativo MQTT.
2. Configure uma assinatura durável criando o script de comando `mqttexamples.txt` e executando-o usando **runmqsc:**

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Figura 90. mqttExampleTopic.txt

Para executar o script no Windows, digite o comando:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Execute o cliente como um aplicativo Java a partir do Eclipseou executando Java em uma janela de comando:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

Nota: A janela de comando deve ser aberta no diretório que contém o caminho `com\ibm\mq\id`.

4. Navegue pelos resultados usando o WebSphere MQ Explorer ou execute o comando:

```
amqsbcg MQTTExampleQueue queue manager name
```

Código de exemplo

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubAsync {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      CallBack callback = new CallBack(Example.clientId);
      client.setCallback(callback);
      client.connect();
      System.out.println("Publishing \"" + message.toString()
          + "\" on topic \"" + topic.getName() + "\" with QoS = "
          + message.getQos());
      System.out.println("For client instance \"" + client.getClientId()
          + "\" on address " + client.getServerURI() + "\"");
      MqttDeliveryToken token = topic.publish(message);
      System.out.println("With delivery token \"" + token.hashCode()
          + " delivered: " + token.isComplete());
      if (client.isConnected())
          client.disconnect(Example.quiesceTimeout);
      System.out.println("Disconnected: delivery token \"" + token.hashCode()
          + "\" received: " + token.isComplete());
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Figura 91. PubAsync.java

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class CallBack implements MqttCallback {
  private String instanceData = "";
  public CallBack(String instance) {
    instanceData = instance;
  }
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\" for instance \""
          + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public void connectionLost(Throwable cause) {
    System.out.println("Connection lost on instance \"" + instanceData
        + "\" with cause \"" + cause.getMessage() + "\" Reason code "
        + ((MqttException)cause).getReasonCode() + "\" Cause \""
        + ((MqttException)cause).getCause() +  "\"");
    cause.printStackTrace();
  }
  public void deliveryComplete(MqttDeliveryToken token) {
    try {
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" received by instance \"" + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Figura 92. CallBack.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Figura 93. Example.java

Criando um publicador assíncrono recuperável para o MQ Telemetry Transport usando Java

Nesta tarefa, você seguirá um tutorial para modificar seu aplicativo publicador assíncrono. As modificações permitem que o aplicativo conclua a entrega de publicações que não foram reconhecidas na última vez em que o cliente foi executado.

Antes de começar

Instale o recurso de Telemetria do WebSphere MQ em um servidor que tenha o IBM WebSphere MQ Version 7.1 ou posterior instalado

O aplicativo cliente usa o pacote `com.ibm.mq.micro.client.mqttv3` no IBM WebSphere MQ Telemetry Software Development Toolkit (SDK). O SDK faz parte da instalação do IBM WebSphere MQ Telemetry O cliente se conecta ao recurso IBM WebSphere MQ Telemetry para trocar mensagens com IBM WebSphere MQ.

Você também deve instalar as atualizações de telemetria para IBM WebSphere MQ Explorer Version 7.1 para administrar IBM WebSphere MQ Telemetry. As atualizações fazem parte da instalação do IBM WebSphere MQ Telemetry

Um cliente MQTT, em execução no Java SE, requer a versão 6.0 do Java SE ou posterior. IBM Java SE v6.0 faz parte da instalação do IBM WebSphere MQ Version 7.1 . Ele está localizado em *WebSphere MQ installation directory*\java\jre

Sobre esta tarefa

O exemplo é um aplicativo de publicação, PubAsyncRestartable. PubAsyncRestartable publica Hello World no tópico MQTT Examples, sem esperar pela confirmação de que a publicação foi entregue ao gerenciador de filas. Os reconhecimentos de entrega são recebidos em uma classe de retorno de chamada, CallBack. Quaisquer tokens de entrega para publicações que não foram concluídos em uma instância anterior podem ser examinados. Eles também são processados pela classe de retorno de chamada.

Configurando uma assinatura durável para MQTT Examples , é possível verificar se o aplicativo funciona.

O procedimento usa Eclipse para desenvolver, construir e executar o cliente. É possível fazer download do Eclipse a partir do website do projeto Eclipse em www.eclipse.org.

As etapas em Procedimento modificam o aplicativo PubAsync.java em "Criando um publicador assíncrono para o MQ Telemetry Transport usando Java" na página 487

Como alternativa, é possível copiar o código, "Código de exemplo" na página 496, para um novo diretório .\com\ibm\mq\id. Crie três arquivos Java, Example.java, CallBack.javae PubAsyncRestartable.java.. Compile os exemplos usando o comando,

```
javac -cp jar_dir\com.mq.micro.client.mqttv3.jar
       com.ibm.mq.id.PubAsyncRestartable.java com.ibm.mq.id.CallBack.java
com.ibm.mq.id.Example.java
```

Execute PubAsyncRestartable usando o comando,

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.PubAsyncRestartable.class
```

Procedimento

1. No pacote com.ibm.mq.id, clique com o botão direito em PubAsync.java e copie-o. Cole-o no mesmo pacote, renomeando-o como PubAsyncRestartable.
2. Crie um identificador de cliente reutilizável.

   ```
   Example.clientId = String.format(
       "%-23.23s",
       (System.getProperty("user.name") + "_" + (System.getProperty(
           "clientId", "PubAsyncRestartable."))).trim()).replace('-', '_');
   ```

   Figura 94. Identificador de cliente reutilizável

   Os aplicativos em "Criando seu primeiro aplicativo publicador do MQ Telemetry Transport usando Java" na página 482 e "Criando um publicador assíncrono para o MQ Telemetry Transport usando Java" na página 487 usavam um novo identificador de cliente para cada conexão do cliente. Para um publicador ou assinante reinicializável, deve-se usar o mesmo identificador de cliente toda vez que o cliente for conectado, mas diferentes clientes devem usar diferentes identificadores; consulte "Identificador de Cliente" na página 537. O identificador de cliente reutilizável será construído a partir do nome de usuário e do nome da classe. Ele está limitado a 23 bytes de comprimento. Ele deve ter somente caracteres que sejam válidos em nomes de objetos do gerenciador de filas. O código remove quaisquer hifens que podem ter sido inseridos.
3. A QoS da mensagem é configurada para 2 em vez de para o padrão, 1, para evitar mensagens duplicadas.

```
message.setQos(Example.QoS);
```

É necessário alterar o valor de `Example.QoS` para 2, ou passar a propriedade `QoS` como um argumento usando a opção `-DQoS=2` na linha de comandos Java

4. Crie um objeto `MqttConnectOptions` e configure seu atributo `cleanSession` como `false`.

   a) Crie um objeto `MqttConnectOptions`.

   ```
   MqttConnectOptions conOptions = new MqttConnectOptions();
   ```

   `conOptions` é um parâmetro de opção no construtor `MqttClient`.

   b) Configure o atributo clearSession.

   ```
   conOptions.setCleanSession(Example.cleanSession);
   ```

   Por padrão, o parâmetro `Example.cleanSession` está configurado como `true`, correspondendo à configuração padrão de `MqttConnectionOptions.cleanSession`.

   Quando `PubAsyncRestartable` é reiniciado, ele pode começar com uma "sessão limpa" e limpe quaisquer tokens de entrega pendente para mensagens de QoS 1 ou 2.

   Configure `Example.cleanSession` como `false` para manter todos os tokens de entrega pendente. Os tokens são processados pela classe `MqttCallBack` quando o cliente é conectado novamente.

5. Se a sessão estiver sendo reiniciada, em seguida, recupere quaisquer tokens de entrega pendente e imprima seu conteúdo.

```
if (!conOptions.isCleanSession()) {
        MqttDeliveryToken tokens[] = client.getPendingDeliveryTokens();
        System.out.println("Starting a previous session for instance \""
            + client.getClientId() + "\" with " + tokens.length
            + " delivery tokens pending");
        for (int i = 0; i < tokens.length; i++) {
          System.out.println("Message \"" + tokens[i].getMessage().toString()
              + "\" with QoS=" + tokens[i].getMessage().getQos()
              + " recovered by instance \"" + client.getClientId()
              + "\" and assigned delivery token \"" + tokens[i].hashCode()
              + "\"");
        }
      } else
        System.out.println("Starting a clean session for instance "
            + client.getClientId());
```

6. Passe o parâmetro `conOptions` para o construtor `MqttClient`.

```
client.connect(conOptions);
```

7. Ao desconectar, configure um intervalo de desconexão máximo.

```
client.disconnect(Example.timeout);
```

Para poder mostrar os tokens de entrega pendente que estão sendo processados, uma instância anterior deve terminar sem concluir a entrega. Para executar o exemplo com a possibilidade de não reconhecer as publicações antes do `PubAsyncRestartable`, configure `Example.timeout` para 0.

Resultados

Para ver as publicações usando WebSphere MQ, crie um tópico, uma fila e uma assinatura durável, todos chamados "`MQTTExampleTopic`" usando o script em Figura 95 na página 495. Execute o cliente para publicar no tópico `MQTT Examples` e, em seguida, execute o programa de amostra **amqsbcg** para procurar as publicações na fila `MQTTExamples` .

1. Inicie um gerenciador de filas e inicie a execução do seu serviço de telemetria (MQXR). Certifique-se de que o endereço TCP/IP e a porta configurados para o canal de telemetria correspondam aos valores usados no aplicativo MQTT.

2. Configure uma assinatura durável criando o script de comando `mqttexamples.txt` e executando-o usando **runmqsc:**

```
DEFINE TOPIC('MQTTExampleTopic') TOPICSTR('MQTT Example') REPLACE
DEFINE QLOCAL('MQTTExampleQueue') REPLACE
DEFINE SUB('MQTTExampleSub') DEST('MQTTExampleQueue') TOPICOBJ('MQTTExampleTopic') REPLACE
```

Figura 95. mqttExampleTopic.txt

Para executar o script no Windows, digite o comando:

```
runmqsc queue manager name < mqttExampleTopic.txt
```

3. Execute o cliente como um aplicativo Java a partir do Eclipseou executando Java em uma janela de comando:

```
java -cp jar_dir\com.mq.micro.client.mqttv3.jar
      com.ibm.mq.id.classname.class
```

Nota: A janela de comando deve ser aberta no diretório que contém o caminho `com\ibm\mq\id`.

4. Navegue pelos resultados usando o WebSphere MQ Explorer ou execute o comando:

```
amqsbcg MQTTExampleQueue queue manager name
```

Código de exemplo

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.MqttClient;
import com.ibm.micro.client.mqttv3.MqttConnectOptions;
import com.ibm.micro.client.mqttv3.MqttDeliveryToken;
import com.ibm.micro.client.mqttv3.MqttMessage;
import com.ibm.micro.client.mqttv3.MqttTopic;
public class PubAsyncRestartable {
  public static void main(String[] args) {
    Example.clientId = String.format(
        "%-23.23s",
        (System.getProperty("user.name") + "_" + (System.getProperty(
            "clientId", "PubAsyncRestartable."))).trim()).replace('-', '_');
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      CallBack callback = new CallBack(Example.clientId);
      client.setCallback(callback);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setCleanSession(Example.cleanSession);
      if (!conOptions.isCleanSession()) {
        MqttDeliveryToken tokens[] = client.getPendingDeliveryTokens();
        System.out.println("Starting a previous session for instance \""
            + client.getClientId() + "\" with " + tokens.length
            + " delivery tokens pending");
        for (int i = 0; i < tokens.length; i++) {
          System.out.println("Message \"" + tokens[i].getMessage().toString()
              + "\" with QoS=" + tokens[i].getMessage().getQos()
              + " recovered by instance \"" + client.getClientId()
              + "\" and assigned delivery token \"" + tokens[i].hashCode()
              + "\"");
        }
      } else
        System.out.println("Starting a clean session for instance \""
            + client.getClientId() + "\"");
      client.connect(conOptions);
      System.out.println("Publishing \"" + message.toString()
          + "\" on topic \"" + topic.getName() + "\" with QoS = "
          + message.getQos());
      System.out.println("For client instance \"" + client.getClientId()
          + "\" on address " + client.getServerURI() + "\"");
      MqttDeliveryToken token = topic.publish(message);
      System.out.println("With delivery token \"" + token.hashCode()
          + " delivered: " + token.isComplete());
      if (client.isConnected())
          client.disconnect(Example.quiesceTimeout);
      System.out.println("Disconnected: delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Figura 96. PubAsyncRestartable.java

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class CallBack implements MqttCallback {
  private String instanceData = "";
  public CallBack(String instance) {
    instanceData = instance;
  }
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\" for instance \""
          + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public void connectionLost(Throwable cause) {
    System.out.println("Connection lost on instance \"" + instanceData
        + "\" with cause \"" + cause.getMessage() + "\" Reason code "
        + ((MqttException)cause).getReasonCode() + "\" Cause \""
        + ((MqttException)cause).getCause() +  "\"");
    cause.printStackTrace();
  }
  public void deliveryComplete(MqttDeliveryToken token) {
    try {
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" received by instance \"" + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Figura 97. CallBack.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Figura 98. Example.java

Criando um assinante para o MQ Telemetry Transport usando Java

Nesta tarefa, você seguirá um tutorial para criar um aplicativo de assinante. O assinante cria uma assinatura para um tópico e recebe publicações para a assinatura.

Antes de começar

Instale o recurso de Telemetria do WebSphere MQ em um servidor que tenha o IBM WebSphere MQ Version 7.1 ou posterior instalado

O aplicativo cliente usa o pacote `com.ibm.mq.micro.client.mqttv3` no IBM WebSphere MQ Telemetry Software Development Toolkit (SDK). O SDK faz parte da instalação do IBM WebSphere MQ Telemetry O cliente se conecta ao recurso IBM WebSphere MQ Telemetry para trocar mensagens com IBM WebSphere MQ.

Você também deve instalar as atualizações de telemetria para IBM WebSphere MQ Explorer Version 7.1 para administrar IBM WebSphere MQ Telemetry. As atualizações fazem parte da instalação do IBM WebSphere MQ Telemetry

Um cliente MQTT, em execução no Java SE, requer a versão 6.0 do Java SE ou posterior. IBM Java SE v6.0 faz parte da instalação do IBM WebSphere MQ Version 7.1 . Ele está localizado em *WebSphere MQ installation directory*\java\jre

Sobre esta tarefa

O exemplo é um aplicativo de assinante, `Subscribe` `Subscribe` cria um tópico de assinatura, `MQTT Examples`, e aguarda publicações na assinatura por 30 segundos.

Um assinante pode criar uma assinatura e aguardar publicações. Ele também pode receber publicações enviadas para uma assinatura criada anteriormente para o mesmo identificador de cliente. O atributo booleano `MqttConnectionOptions.cleanSession` controla se as publicações enviadas anteriormente são ou não recebidas; consulte "Assinaturas" na página 548.

É possível usar os programas de exemplo de publicação para criar publicações ou usar o explorador do WebSphere MQ para criar uma publicação de teste no tópico `MQTT Examples` .

O procedimento usa Eclipse para desenvolver, construir e executar o cliente. É possível fazer download do Eclipse a partir do website do projeto Eclipse em www.eclipse.org.

As instruções em Procedimento presumem que você já criou o pacote `com.ibm.mq.id` em uma das tarefas anteriores e copiou as classes `Example.java` e `Callback.java`.

Procedimento

1. Crie a classe, `Subscribe` no pacote `com.ibm.mq.id`
2. Crie um identificador de cliente reutilizável.

```
Example.clientId = String.format(
    "%-23.23s",
    (System.getProperty("user.name") + "_" + (System.getProperty(
        "clientId", "Subscribe."))).trim()).replace('-', '_');
```

Figura 99. Identificador de cliente reutilizável

Os aplicativos em "Criando seu primeiro aplicativo publicador do MQ Telemetry Transport usando Java" na página 482 e "Criando um publicador assíncrono para o MQ Telemetry Transport usando Java" na página 487 usavam um novo identificador de cliente para cada conexão do cliente. Para um publicador ou assinante reinicializável, deve-se usar o mesmo identificador de cliente toda vez que o cliente for conectado, mas diferentes clientes devem usar diferentes identificadores; consulte "Identificador de Cliente" na página 537. O identificador de cliente reutilizável será construído a partir do nome de usuário e do nome da classe. Ele está limitado a 23 bytes de comprimento. Ele deve ter somente caracteres que sejam válidos em nomes de objetos do gerenciador de filas. O código remove quaisquer hifens que podem ter sido inseridos.

3. Crie um bloco `try-catch`.

```
try { ...
 } catch (Exception e) {
  e.printStackTrace();
}
```

O cliente MQTT lança `MqttException`, `MqttPersistenceException` ou `MqttSecurityException`.. `MqttPersistenceException` e `MqttSecurityException` são subclasses de `MqttException`.

Use o método `MqttException.getReasonCode` para descobrir a razão da exceção. Se `MqttPersistenceException` ou `MqttSecurityException` for lançado, use o método `getCause` para retornar a exceção lançável subjacente.

4. Crie uma nova instância de `MqttClient`.

```
MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
```

Forneça ao cliente um endereço do servidor, que será usado posteriormente para se conectar ao WebSphere MQ. Configure o identificador de cliente para o nome do cliente.

- Opcionalmente, é possível fornecer uma implementação da interface `MqttClientPersistence` para substituir a implementação padrão. A implementação padrão do `MqttPersistence` armazena QoS 1 e 2 mensagens aguardando entrega como arquivos; consulte "Persistência de Mensagem em Clientes MQTT" na página 541.
- A porta TCP/IP padrão IBM WebSphere MQ para MQTT é 1883. Para SSL, ela é 8883. No exemplo, o endereço padrão é configurado como `tcp://localhost:1883`.
- Normalmente, é importante poder identificar um cliente físico específico usando o identificador de cliente. O identificador de cliente deve ser exclusivo em todos os clientes conectados a um servidor; consulte "Identificador de Cliente" na página 537. O uso do mesmo identificador de cliente que uma instância anterior indica que a instância presente é uma instância do mesmo cliente. Se você duplicar um identificador de cliente nos dois clientes em execução, uma exceção será lançada nos dois clientes e um deles será finalizado.
- O comprimento do identificador de cliente é limitado a 23 bytes. Uma exceção é lançada se o comprimento for excedido. O identificador de cliente deve conter apenas caracteres permitidos em um nome do gerenciador de filas; por exemplo, sem hifens ou espaços.
- Até você chamar o método `MqttClient.connect`, nenhum processamento de mensagem ocorre.

Use o objeto do cliente para publicar e assinar tópicos e recuperar informações sobre publicações que ainda não foram entregues.

5. Imediatamente antes da linha de código `client.connect();`, instancie a classe `CallBack`, passando o identificador de cliente.

```
CallBack callback = new CallBack(Example.clientId);
client.setCallback(callback);
```

   - A classe `CallBack` implementa `MqttCallBack`. É necessária uma instância de retorno por identificador de cliente. Nesse exemplo, o construtor passa o identificador de cliente para salvar como dados da instância. Ele é usado no retorno de chamada para identificar qual instância do retorno de chamada foi iniciada.
   - Deve-se implementar três métodos na classe de retorno de chamada:

     `public void messageArrived(MqttTopic topic, MqttMessage message)`
     Recebe uma publicação que foi assinada.

     `public void connectionLost(Throwable cause)`
     Chamado quando a conexão é perdida.

     `public void deliveryComplete(MqttDeliveryToken token))`
     Chamado quando um token de entrega é recebido para uma mensagem de QoS 1 ou 2 que foi publicada.
   - O retorno de chamada é ativado por `MqttClient.connect`.

6. Crie um objeto `MqttConnectOptions` e configure seu atributo `cleanSession`.
   a) Crie um objeto `MqttConnectOptions`.

```
MqttConnectOptions conOptions = new MqttConnectOptions();
```

      `conOptions` é um parâmetro de opção no construtor `MqttClient`.
   b) Configure o atributo clearSession.

```
conOptions.setCleanSession(Example.cleanSession);
```

      Por padrão, o parâmetro `Example.cleanSession` está configurado como `true`, correspondendo à configuração padrão de `MqttConnectionOptions.cleanSession`.

Se você usar `MqttConnectOptions` padrão ou configurar `MqttConnectOptions.cleanSession` como `true` antes de se conectar ao cliente, todas as assinaturas antigas para o cliente serão

removidas quando o cliente se conectar. Quaisquer novas assinaturas feitas pelo cliente durante a sessão serão removidas quando ele se desconectar.

Se você configurar `MqttConnectOptions.cleanSession` como `false` antes de se conectar, todas as assinaturas criadas pelo cliente serão incluídas em todas as assinaturas que existiam para o cliente antes de ele se conectar. Todas as assinaturas permanecem ativas quando o cliente se desconecta.

Outra maneira de entender a maneira como o atributo `cleanSession` afeta assinaturas é pensar nele como um atributo modal. Em seu modo padrão, `cleanSession=true`, o cliente cria assinaturas e recebe publicações apenas dentro do escopo da sessão. No modo alternativo, `cleanSession=false`, assinaturas são duráveis. O cliente pode se conectar e se desconectar e suas assinaturas permanecem ativas. Ao se reconectar, o cliente recebe todas as publicações não entregues. Enquanto estiver conectado, ele pode modificar o conjunto de assinaturas que estão ativas em seu nome.

Você deve configurar o modo `cleanSession` antes de se conectar; o modo dura a sessão inteira. Para alterar essa configuração, você deve desconectar e reconectar o cliente. Se você alterar os modos do uso de `cleanSession=false` para `cleanSession=true`, todas as assinaturas anteriores para o cliente e todas as publicações que ainda não foram recebidas serão descartadas.

7. Passe o parâmetro `conOptions` para o construtor `MqttClient`.

```
client.connect(conOptions);
```

8. Crie uma assinatura.

```
      client.subscribe(Example.topicString, Example.QoS);
```

O exemplo usa um método `MqttClient.subscribe` que passa um filtro de tópico com uma opção QoS. O método `MqttClient.subscribe` tem quatro assinaturas e é possível passar matrizes de filtros de assinatura, bem como um único filtro.

O exemplo usa a sequência de tópicos usada pelos exemplos de publicação como um filtro de tópico, portanto, ele recebe todas as publicações criadas.

Cada vez que você executa o exemplo, `subscribe.java`, ele cria uma assinatura. Se você não mudar Example.topicString, ele recriará a mesma assinatura novamente. Se uma assinatura for recriada, o resultado não será duas assinaturas idênticas. Um cliente não recebe cópias duplicadas de publicações que correspondem a uma assinatura idêntica.

Assinaturas estão descritas em “Assinaturas” na página 548 e os filtros em “Sequências e filtros de tópicos em clientes MQTT” na página 550.

9. Aguarde algumas publicações chegarem e desconecte o cliente.

```
      Thread.sleep(Example.sleepTimeout);
      client.disconnect();
```

Publicações são recebidas pela implementação do método `MqttCallback.messageArrived`.

O aplicativo de assinatura não publicou nenhuma mensagem e, portanto, não aguarda nenhum token de entrega. `client.disconnect` acontece sem qualquer atraso.

Código de exemplo

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.MqttClient;
import com.ibm.micro.client.mqttv3.MqttConnectOptions;
public class Subscribe {
  public static void main(String[] args) {
    Example.clientId = String.format(
        "%-23.23s",
        (System.getProperty("user.name") + "_" + System.getProperty("clientId",
            "Subscribe."))).trim();
    try {
      MqttClient client = new MqttClient(Example.TCPAddress, Example.clientId);
      CallBack callback = new CallBack(Example.clientId);
      client.setCallback(callback);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setCleanSession(Example.cleanSession);
      client.connect(conOptions);
      System.out.println("Subscribing to topic \"" + Example.topicString
          + "\" for client instance \"" + client.getClientId()
          + "\" using QoS " + Example.QoS + ". Clean session is "
          + Example.cleanSession);
      client.subscribe(Example.topicString, Example.QoS);
      System.out.println("Going to sleep for " + Example.sleepTimeout / 1000
          + " seconds");
      Thread.sleep(Example.sleepTimeout);
      client.disconnect();
      System.out.println("Finished");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Figura 100. Subscribe.java

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class CallBack implements MqttCallback {
  private String instanceData = "";
  public CallBack(String instance) {
    instanceData = instance;
  }
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\" for instance \""
          + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public void connectionLost(Throwable cause) {
    System.out.println("Connection lost on instance \"" + instanceData
        + "\" with cause \"" + cause.getMessage() + "\" Reason code "
        + ((MqttException)cause).getReasonCode() + "\" Cause \""
        + ((MqttException)cause).getCause() +  "\"");
    cause.printStackTrace();
  }
  public void deliveryComplete(MqttDeliveryToken token) {
    try {
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" received by instance \"" + instanceData + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Figura 101. CallBack.java

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Figura 102. Example.java

Conceitos relacionados

Aplicativos de Publicação/Assinatura MQTT

Autenticando um cliente MQTT Java usando JAAS

Saiba como autenticar um cliente usando JAAS. Modifique o programa de amostra `JAASLoginModule.java` e o programa Java de exemplo `PubSync.java` Configure um canal de telemetria para requerer a autenticação JAAS e execute o publicador modificado, verificando seu `nome do usuário` e `senha` usando JAAS.

Antes de começar

Supõe-se que você tenha instalado os arquivos jar do cliente MQTT v3 , Javadoc, Eclipse, configurado canais de telemetria e codificado e executado PubSync.java antes de executar esta tarefa Você tem uma área de trabalho do Eclipse que inclui uma versão em execução de PubSync.java.

A tarefa é escrita para Windows. Mude os caminhos de diretório para o Linux.

Sobre esta tarefa

A tarefa é baseada na modificação da classe JAASLoginModule de amostra em *WMQ Installation directory*\mqxr\samples\JAASLoginModule.java para criar MyLogin.java.. Na tarefa, você também modifica o código de exemplo, PubSync.java em "Criando seu primeiro aplicativo publicador do MQ Telemetry Transport usando Java" na página 482, para configurar um nome do usuário e uma senha. Como um teste, MyLogin.java aceita ou rejeita aleatoriamente o nome do usuário e a senha.

As etapas da tarefa são escritas como um exercício de programação. Deve-se adaptar o procedimento para executar autenticação real em um ambiente de produção.

Em uma explicação típica de como programar a autenticação JAAS, presume-se que o módulo de login esteja autenticando o contexto que carregou JAAS. Quando o serviço de telemetria (MQXR) chama JAAS, o contexto que carregou JAAS é o serviço de telemetria (MQXR). Não há nenhuma razão para autenticar o contexto do serviço de telemetria (MQXR); sempre é mqm. Em vez disso, o serviço de telemetria (MQXR) configura o nome do usuário e a senha do cliente para estarem disponíveis para a classe do módulo de login. O nome do usuário e a Senha são passados para o módulo de login usando dois retornos de chamada.

```
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);
callbackHandler.handle(callbacks);
String username =
      ((javax.security.auth.callback.NameCallback) callbacks[0]).getName();
char[] password =
      ((javax.security.auth.callback.PasswordCallback) callbacks[1]).getPassword();
```

O nome do usuário e a senha do cliente são as únicas informações sobre o cliente que estão disponíveis para o módulo de login.

Procedimento

1. Crie dois pacotes samples e security.jaas no mesmo projeto Java que PubSync.java.

   O pacote samples é utilizado somente para referência. Faça as mudanças de código no pacote security.jaas.
2. Importe JAASLoginModule.java e JAASPrincipal.java para ambos os pacotes.

   Se necessário, refatore as instruções do pacote na origem Java para eliminar os erros de compilação
3. Refatore o nome da classe, JAASLoginModule, no pacote security.jaas para MyLogin
4. Em MyLogin.java, substitua parte do código no método login para mostrar o módulo funcionando.
   a) Substitua o código:

   ```
   // Accept everything.
   if (true)
     loggedIn = true;
   else
     throw new javax.security.auth.login.FailedLoginException("Login failed");
   ```

   b) Pelo código:

   ```
   // login half the users randomly
   PrintWriter pw = new PrintWriter(new FileWriter(System.getProperty("user.dir")
           + "\\MyLogin.log", true));
   pw.println("Called JAASLogin.login at "
           + System.getProperty("publication", "Hello World "
           + String.format("%tc", System.currentTimeMillis())));
   if (Math.random() < 0.5)
     loggedIn = true;
   pw.println("Username: \"" + username + "\", Password: \""
       + String.valueOf(password) + "\" loggedIn: " + loggedIn);
   ```

```
pw.close();
if (!loggedIn)
  throw new javax.security.auth.login.FailedLoginException("Login failed");
principal= new JAASPrincipal(username);
```

A origem completa para `MyLogin.java` está em Figura 105 na página 507. A origem para `JAASPrincipal.java`, com o nome do pacote refatorado para `security.jaas` está em Figura 106 na página 508.

5. Configure o caminho de classe em `service.env` para apontar para o diretório que contém o caminho para `security/jaas/MyLogin.class` e `security/jaas/JAASPrincipal.class`.

```
CLASSPATH=C:\WMQTelemetryApps\MQTTSecureExamples\bin
```

Consulte Configuração do canal de telemetria JAAS para obter informações sobre como usar `service.env` para transmitir um caminho de classe para um serviço do WebSphere MQ .

6. Inclua uma sub-rotina do módulo de login em `jaas.config`.

```
MyLoginExample {
  security.jaas.MyLogin required debug=true;
  };
```

Consulte Configuração de JAAS do canal de telemetria para obter informações sobre o uso de `jaas.config` para definir um módulo de login JAAS.

7. Inclua um canal de telemetria usando o assistente **Novo canal de telemetria** no WebSphere MQ Explorer, configurando o canal para requerer autenticação JAAS . Refira-o à sub-rotina `MyLoginExample`.

Por exemplo, adapte as informações digitadas no assistente a partir dessa sub-rotina no arquivo `mqxr_win.properties`. Se estiver trabalhando no Linux, o arquivo será chamado `mqxr_unix.properties`. Não edite o arquivo de propriedades de telemetria diretamente; use o assistente.

```
com.ibm.mq.MQXR.channel/JAASMCAUser: \
com.ibm.mq.MQXR.Port=1884;\
com.ibm.mq.MQXR.JAASConfig=MyLoginExample;\
com.ibm.mq.MQXR.UserName=Admin;\
com.ibm.mq.MQXR.StartWithMQXRService=true
```

Nota: Se você modifica qualquer um dos parâmetros do canal de telemetria ou modifica a classe `security.jaas.Mylogin`, deve-se parar e reiniciar o serviço de telemetria (MQXR). Somente quando o serviço for reiniciado é que as mudanças entrarão em vigor.

8. Faça uma cópia de `PubSync.java` no pacote `com.ibm.mq.id` e denomine a cópia `PubSyncJAAS.java`.

Consulte “Criando seu primeiro aplicativo publicador do MQ Telemetry Transport usando Java” na página 482 para obter as etapas para criar PubSync.java no pacote `com.ibm.mq.id`.

9. Configure `MqttConnectOptions.username` e `MqttConnectOptions.password` no programa `PubSyncJAAS.java` e passe `MqttConnectOptions` como um parâmetro de `MqttClient.connect`.

```
MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setUserName(Example.username);
conOptions.setPassword(Example.password);
client.connect(conOptions);
```

Consulte o código em itálico no JAAS do PubSyncJAAS.Java usando as constantes configuradas em Example.java

10. Configure `Example.TCPAddress` para o endereço de soquete do canal de telemetria configurado para usar a configuração JAAS, `MyLoginExample`. Por exemplo, use 1884 como o número da porta.
11. Execute `PubSyncJAAS` diversas vezes para ver o cliente efetuar login e ser aceito ou rejeitado.

Uma exceção é lançada toda vez que a tentativa de login for rejeitada.

Resultados

Figura 103 na página 506 mostra os resultados do PubSyncJAAS.Java duas vezes.. Os registros de log são mostrados em Figura 104 na página 506.

```
Waiting for up to 10 seconds for publication of "Hello World Fri Jun 04 08:31:05 BST 2010" with
QoS = 1
On topic "MQTT Example" for client instance: "Admin_61c57a18_4bf7_40d" on address tcp://
localhost:1884"
With username "Admin" and password "Password"
Client exception caught
Client is not connected (32104)
    at
com.ibm.micro.client.mqttv3.internal.ExceptionHelper.createMqttException(ExceptionHelper.java:33
)
    at com.ibm.micro.client.mqttv3.internal.ClientComms.internalSend(ClientComms.java:88)
    at com.ibm.micro.client.mqttv3.internal.ClientComms.sendNoWait(ClientComms.java:105)
    at com.ibm.micro.client.mqttv3.MqttTopic.publish(MqttTopic.java:68)
    at com.ibm.mq.id.PubSync.main(PubSync.java:24)
```

```
Waiting for up to 10 seconds for publication of "Hello World Fri Jun 04 08:31:40 BST 2010" with
QoS = 1
On topic "MQTT Example" for client instance: "Admin_1d1599a0_50f5_4ea" on address tcp://
localhost:1884"
With username "Admin" and password "Password"
Delivery token "1731749688" has been received: true
```

Figura 103. Saída do console de PubSyncJAAS.java

O arquivo de log MyLogin.log é armazenado em *WMQ Data directory*; por exemplo C:\IBM\MQ\Data\MyLogin.log:

```
Called JAASLogin.login at Hello World Fri Jun 04 08:31:05 BST 2010
Username: "Admin", Password: "Password" loggedIn: false
Called JAASLogin.login at Hello World Fri Jun 04 08:31:40 BST 2010
Username: "Admin", Password: "Password" loggedIn: true
```

Figura 104. MyLogin.log

Examples

O código em itálico em Figura 105 na página 507 é a modificação da amostra JAASLoginModule.java.

```
package security.jaas;
import java.io.FileWriter;
import java.io.PrintWriter;

public class JAASLogin implements javax.security.auth.spi.LoginModule {
  private javax.security.auth.Subject                  subject;
  private javax.security.auth.callback.CallbackHandler callbackHandler;
  JAASPrincipal                                        principal;
  boolean                                              loggedIn = false;
  public void initialize(javax.security.auth.Subject subject,
      javax.security.auth.callback.CallbackHandler callbackHandler,
      java.util.Map<String, ?> sharedState, java.util.Map<String, ?> options) {
    this.subject = subject;
    this.callbackHandler = callbackHandler;
  }
  public boolean login() throws javax.security.auth.login.LoginException {
    try {
      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);

      callbackHandler.handle(callbacks);
      String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
          .getName();
      char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
          .getPassword();
      // login half the users randomly
      PrintWriter pw = new PrintWriter(new FileWriter(System
          .getProperty("user.dir")
          + "\\mylogin.log", true));
      pw.println("Called JAASLogin.login at "
          + System.getProperty("publication", "Hello World "
              + String.format("%tc", System.currentTimeMillis())));
      if (Math.random() < 0.5)
        loggedIn = true;
      pw.println("Username: \"" + username + "\", Password: \""
          + String.valueOf(password) + "\" loggedIn: " + loggedIn);
      pw.close();
      if (!loggedIn)
        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;
  }
  public boolean abort() throws javax.security.auth.login.LoginException {
    logout();
    return true;
  }
  public boolean commit() throws javax.security.auth.login.LoginException {
    if (loggedIn) {
      if (!subject.getPrincipals().contains(principal))
        subject.getPrincipals().add(principal);
    }
    return true;
  }
  public boolean logout() throws javax.security.auth.login.LoginException {
    subject.getPrincipals().remove(principal);
    principal = null;
    loggedIn = false;
    return true;
  }
}
```

Figura 105. MyLogin.java

Figura 106 na página 508 é o código de amostra JAASLoginPrincipal.java copiado para o pacote security.jaas. O propósito de JAASLoginPrincipal é implementar a interface

`java.security.Principal` para manter um registro dos usuários que efetuaram login com sucesso pelo `MyLogin`.

```
package security.jaas;
public class JAASPrincipal implements java.security.Principal,
    java.io.Serializable {
  private static final long serialVersionUID = 1L;
  String                    name;
  public JAASPrincipal(String name) {
    this.name = name;
  }
  public String getName() {
    return name;
  }
  public String toString() {
    return (name);
  }
  public boolean equals(Object object) {
    if (object != null && object instanceof JAASPrincipal
        && name.equals(((JAASPrincipal) object).getName()))
      return true;
    else
      return false;
  }
  public int hashCode() {
    return name.hashCode();
  }
}
```

Figura 106. JAASLoginPrincipal.java

O código em PubSync.java modificado para incluir um `nome do usuário` e uma `senha` está em itálico em Figura 107 na página 508.

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubSyncSSL {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.SSLAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setUserName(Example.username);
      conOptions.setPassword(Example.password);
      client.connect(conOptions);
      System.out.println("Waiting for up to " + Example.sleepTimeout / 1000
              + " seconds for publication of \"" + message.toString()
              + "\" with QoS = " + message.getQos());
      System.out.println("On topic \"" + topic.getName() + "\" for client instance: \""
              + client.getClientId() + "\" on address " + client.getServerURI() + "\"");
      System.out.println("With username \"" + conOptions.getUserName()
              + "\" and password \"" + String.valueOf(conOptions.getPassword()) + "\"");
      MqttDeliveryToken token = topic.publish(message);
      token.waitForCompletion(Example.sleepTimeout);
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
      client.disconnect();
    } catch (Exception e) {
      System.out.println("Client exception caught");
      e.printStackTrace();
    }
  }
}
```

Figura 107. PubSyncJAAS.Java

Modifique as constantes de Example.java para corresponderem à sua configuração. Ignore as configurações de SSL para este exemplo.

```
package com.ibm.mq.id;
import java.util.Properties;
import java.util.UUID;
public final class Example {
  public static final String          TCPAddress =
    System.getProperty("TCPAddress", "tcp://localhost:1883");
  public static final String          SSLAddress =
    System.getProperty("SSLAddress", "ssl://localhost:8883");
  public static final String            username =
    System.getProperty("username", System.getProperty("user.name"));
  public static final char []           password =
    System.getProperty("password", "Password").toCharArray();
  public static String                  clientId =
  String.format("%-23.23s", username  + "_" +
    System.getProperty("clientId",
    (UUID.randomUUID().toString())).trim()).replace('-', '_');
  public static final String         topicString =
    System.getProperty("topicString", "MQTT Example");
  public static final String         publication =
    System.getProperty("publication", "Hello World " +
  String.format("%tc", System.currentTimeMillis()));
  public static final int         quiesceTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final int           sleepTimeout =
    Integer.parseInt(System.getProperty("timeout", "10000"));
  public static final boolean       cleanSession =
    Boolean.parseBoolean(System.getProperty("cleanSession", "false"));
  public static final int                   QoS =
    Integer.parseInt(System.getProperty("QoS", "1"));
  public static final boolean           retained =
    Boolean.parseBoolean(System.getProperty("retained", "false"));
  public static final Properties getSSLSettings() {
    final Properties properties = new Properties();
    properties.setProperty("com.ibm.ssl.keyStore",
        "C:\\IBM\\MQ\\Data\\ClientKeyStore.jks");
    properties.setProperty("com.ibm.ssl.keyStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.keyStorePassword",
        "password");
    properties.setProperty("com.ibm.ssl.trustStore",
        "C:\\IBM\\MQ\\Data\\ClientTrustStore.jks");
    properties.setProperty("com.ibm.ssl.trustStoreType",
        "JKS");
    properties.setProperty("com.ibm.ssl.trustStorePassword",
        "password");
    return properties;
  }
}
```

Figura 108. Example.java

Autenticando uma conexão de telemetria SSL usando certificados autoassinados

Use certificados autoassinados gerados usando **Keytool** para autenticar uma conexão SSL. Você tem a opção de autenticar o canal de telemetria ou o canal de telemetria e os clientes que se conectam a ele. As mensagens que fluem na conexão são criptografadas.

Antes de começar

Execute a tarefa, “Criando seu primeiro aplicativo publicador do MQ Telemetry Transport usando Java” na página 482 antes de iniciar, para que PubSync.java trabalhe com uma conexão TCP/IP não segura. Nesta tarefa, você modifica PubSync.java para trabalhar com uma conexão SSL.

Sobre esta tarefa

As etapas da tarefa são escritas como um exercício de programação. Deve-se adaptar o procedimento para executar autenticação real em um ambiente de produção.

A tarefa é escrita para Windows. Mude os caminhos de diretório para o Linux.

Procedimento

1. Execute a tarefa, "Modificando PubSync.java para usar a SSL" na página 510, para modificar o PubSync.java para usar SSL.
2. Configure o canal de telemetria e crie os keystores para usar SSL.

   Autentique apenas o canal de telemetria ou o canal e os clientes que se conectam a ele:

   - Execute a tarefa, "Autenticando o canal de telemetria" na página 511, para se conectar com a SSL, autenticando o canal de telemetria.
   - Execute a tarefa, "Autenticando o canal de telemetria e os clientes" na página 512, para se conectar com a SSL, autenticando o canal de telemetria e os clientes que se conectam a ele.
3. Pare e reinicie o serviço de telemetria (MQXR) para escolher as mudanças nas configurações do canal de telemetria.
4. Execute o programa cliente para ver se a configuração funciona.

Modificando PubSync.java para usar a SSL

Modifique o exemplo do programa do primeiro publicador para se conectar a um canal de telemetria usando SSL. Configure as propriedades SSL usadas pelo programa modificado.

Antes de começar

Supõe-se que você tenha instalado os arquivos jar do cliente MQTT v3 , Javadoc, Eclipse, configurado canais de telemetria e codificado e executado PubSync.java antes de executar esta tarefa Você tem uma área de trabalho do Eclipse que inclui uma versão em execução de PubSync.java.

Sobre esta tarefa

A tarefa usa o cliente do publicador, PubSync.java, que você criou no "Criando seu primeiro aplicativo publicador do MQ Telemetry Transport usando Java" na página 482 como uma base. Apenas pequenas modificações são necessárias para usar a SSL; consulte Figura 109 na página 511 e Figura 110 na página 511.

Procedimento

1. Faça uma cópia de `PubSync.java` no pacote `com.ibm.mq.id` e o nome da cópia do `PubSyncSSL.java`.

   Consulte "Criando seu primeiro aplicativo publicador do MQ Telemetry Transport usando Java" na página 482 para obter as etapas para criar PubSync.java no pacote `com.ibm.mq.id`.
2. Configure `Example.SSLAddress` para o endereço de soquete do canal de telemetria configurado para usar para a configuração de SSL.
3. Mude o parâmetro do endereço de soquete do construtor cliente para usar `Example.SSLAddress`.

   ```
   MqttClient client = new MqttClient(Example.SSLAddress, Example.clientId);
   ```
4. Configure `MqttConnectOptions.SSLProperties` em `PubSyncSSL.java` e transmita o `MqttConnectOptions` como um parâmetro de `MqttClient.connect`.

   ```
   MqttConnectOptions conOptions = new MqttConnectOptions();
   conOptions.setSSLProperties(Example.getSSLSettings());
   client.connect(conOptions);
   ```

   Consulte o código em itálico PubSyncSSL.Java usando as constantes configuradas em Example.java

Examples

As modificações em PubSync.java para incluir a SSL são mostradas em Figura 109 na página 511 em itálico.

```
package com.ibm.mq.id;
import com.ibm.micro.client.mqttv3.*;
public class PubSyncSSL {
  public static void main(String[] args) {
    try {
      MqttClient client = new MqttClient(Example.SSLAddress, Example.clientId);
      MqttTopic topic = client.getTopic(Example.topicString);
      MqttMessage message = new MqttMessage(Example.publication.getBytes());
      message.setQos(Example.QoS);
      MqttConnectOptions conOptions = new MqttConnectOptions();
      conOptions.setSSLProperties(Example.getSSLSettings());
      client.connect(conOptions);
      System.out.println("Waiting for up to " + Example.sleepTimeout / 1000
              + " seconds for publication of \"" + message.toString()
              + "\" with QoS = " + message.getQos());
      System.out.println("On topic \"" + topic.getName() + "\" for client instance: \""
              + client.getClientId() + "\" on address " + client.getServerURI() + "\"");
      System.out.println("SSL Properties" + conOptions.getSSLProperties());
      MqttDeliveryToken token = topic.publish(message);
      token.waitForCompletion(Example.sleepTimeout);
      System.out.println("Delivery token \"" + token.hashCode()
          + "\" has been received: " + token.isComplete());
      client.disconnect();
    } catch (Exception e) {
      System.out.println("Client exception caught");
      e.printStackTrace();
    }
  }
}
```

Figura 109. PubSyncSSL.Java

As modificações em Example.java são mostradas em Figura 110 na página 511.

```
public static final String          SSLAddress =
   System.getProperty("SSLAddress", "ssl://localhost:8883");

public static final Properties getSSLSettings() {
   final Properties properties = new Properties();
   properties.setProperty("com.ibm.ssl.keyStore", "C:\\Certificates\\SSClientKey.jks");
   properties.setProperty("com.ibm.ssl.keyStoreType", "JKS");
   properties.setProperty("com.ibm.ssl.keyStorePassword", "password");
   properties.setProperty("com.ibm.ssl.trustStore", "C:\\Certificates\\SSClientTrust.jks");
   properties.setProperty("com.ibm.ssl.trustStoreType", "JKS");
   properties.setProperty("com.ibm.ssl.trustStorePassword", "password");
   return properties;
```

Figura 110. Modificações em Example.java

Autenticando o canal de telemetria

Os clientes autenticam o canal de telemetria para criptografar o conteúdo das mensagens que fluem no canal e para assegurar que um cliente se conecta ao canal de telemetria correto. O servidor não autentica o cliente.

Sobre esta tarefa

É possível usar um número de editores de keystore diferentes para criar e gerenciar certificados autoassinados. A tarefa usa o comando **keytool** da linha de comandos que faz parte do JRE. É possível usar a ferramenta da GUI **iKeyman**, que é enviada com o WebSphere MQ para procurar armazenamentos de chaves e gerar chaves Ative **iKeyman** usando o comando **strmqikm**.

Procedimento

1. Crie um canal de telemetria, SSLSSOptClients que requer uma conexão SSL usando o assistente **Novo canal de telemetria**. O canal aceita clientes anônimos.

   Adapte a configuração do seu canal da sub-rotina de configuração a seguir. Não edite o arquivo de propriedades de telemetria diretamente; use o assistente.

   ```
   com.ibm.mq.MQXR.channel/SSLSSOptClients: \
   com.ibm.mq.MQXR.Port=8883;\
   com.ibm.mq.MQXR.Backlog=4096;\
   com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\SSServerOptKey.jks;\
   com.ibm.mq.MQXR.PassPhrase=password;\
   com.ibm.mq.MQXR.ClientAuth=OPTIONAL;\
   com.ibm.mq.MQXR.UserName=Admin;\
   com.ibm.mq.MQXR.StartWithMQXRService=true
   ```

2. Gere as chaves para que o cliente autentique o canal de telemetria.

   a) Gere um par de chaves autoassinado para o canal de telemetria em um novo keystore, SSServerOptKey.jks:

   ```
   Keytool -genkey -noprompt -alias SSServerPrivate
           -dname "CN=mqttserver.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore SSServerOptKey.jks -storepass password -keypass password
   ```

   b) Exporte seu certificado público como um arquivo ASCII, usando a opção -rfc:

   ```
   Keytool -export -noprompt -alias SSServerPrivate -file SSServerPublic.cer
           -keystore SSServerOptKey.jks -storepass password -rfc
   ```

   Se você estiver executando a tarefa em Windows, dê um clique duplo em SSServerPublic.cer para inspecionar seu conteúdo.

   c) Importe o certificado público para um novo armazenamento confiável do cliente, SSClientTrust.jks:

   ```
   Keytool -import -noprompt -alias SSServerPublic -file SSServerPublic.cer
           -keystore SSClientTrust.jks -storepass password
   ```

   d) Crie um keystore do cliente vazio, SSClientKey.jks.

   Keytool não possui um comando para criar um keystore vazio. Há duas opções:

   i) Execute **strmqikm** e crie um keystore, SSClientKey.jks, mas não inclua nenhuma chave.

   ii) Execute a etapa 3a em "Autenticando o canal de telemetria e os clientes" na página 512, mas não use as teclas ainda.

Autenticando o canal de telemetria e os clientes

Os clientes autenticam o canal de telemetria e o canal de telemetria autentica os clientes que se conectam a ele. As mensagens que fluem no canal são criptografadas.

Sobre esta tarefa

É possível usar um número de editores de keystore diferentes para criar e gerenciar certificados autoassinados. A tarefa usa o comando **keytool** da linha de comandos que faz parte do JRE. É possível usar a ferramenta da GUI **iKeyman**, que é enviada com o WebSphere MQ para procurar armazenamentos de chaves e gerar chaves Ative **iKeyman** usando o comando **strmqikm**.

O canal de telemetria é configurado com um keystore diferente para a tarefa, "Autenticando o canal de telemetria" na página 511. É possível usar o mesmo keystore e omitir a etapa "2" na página 513 para incluir chaves no keystore.

Procedimento

1. Crie um canal de telemetria, SSLSSReqClients que requer uma conexão SSL usando o assistente **Novo canal de telemetria**. O canal aceita apenas os clientes autenticados.

   Adapte a configuração do seu canal da sub-rotina de configuração a seguir:

   ```
   com.ibm.mq.MQXR.channel/SSLSSReqClients: \
   com.ibm.mq.MQXR.Port=8884;\
   com.ibm.mq.MQXR.Backlog=4096;\
   com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\SSServerReqKey.jks;\
   com.ibm.mq.MQXR.PassPhrase=password;\
   com.ibm.mq.MQXR.ClientAuth=REQUIRED;\
   com.ibm.mq.MQXR.UserName=Admin;\
   com.ibm.mq.MQXR.StartWithMQXRService=true
   ```

2. Gere as chaves para que o cliente autentique o canal de telemetria.

   a) Gere um par de chaves autoassinado para o canal de telemetria em um novo keystore, SSServerReqKey.jks:

   ```
   Keytool -genkey -noprompt -alias SSServerPrivate
           -dname "CN=mqttserver.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore SSServerReqKey.jks -storepass password -keypass password
   ```

   b) Exporte seu certificado público como um arquivo ASCII, usando a opção -rfc:

   ```
   Keytool -export -noprompt -alias SSServerPrivate -file SSServerPublic.cer
           -keystore SSServerReqKey.jks -storepass password -rfc
   ```

   Se você estiver executando a tarefa em Windows, dê um clique duplo em SSServerPublic.cer para inspecionar seu conteúdo.

   c) Importe o certificado público para um novo armazenamento confiável do cliente, SSClientTrust.jks:

   ```
   Keytool -import -noprompt -alias SSServerPublic -file SSServerPublic.cer
           -keystore SSClientTrust.jks -storepass password
   ```

3. Gere as chaves para que o canal de telemetria autentique um cliente.

   a) Gere um par de chaves autoassinado para o cliente em um novo keystore, SSClientKey.jks:

   ```
   Keytool -genkey -noprompt -alias SSClientPrivate
           -dname "CN=mqttclient.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore SSClientKey.jks -storepass password -keypass password
   ```

   b) Exporte seu certificado público como um arquivo ASCII, usando a opção -rfc:

   ```
   Keytool -export -noprompt -alias SSClientPrivate -file SSClientPublic.cer
           -keystore SSClientKey.jks -storepass password -rfc
   ```

   Se você estiver executando a tarefa em Windows, dê um clique duplo em SSClientPublic.cer para inspecionar seu conteúdo.

   c) Importe o certificado público no keystore do servidor, SSServerReqKey.jks:

   ```
   Keytool -import -noprompt -alias SSClientPublic -file SSClientPublic.cer
           -keystore SSServerReqKey.jks -storepass password
   ```

   Os canais de telemetria usam o mesmo armazenamento para ambas as chaves privadas e certificados confiáveis.

Autenticando uma conexão de telemetria SSL usando uma sequência de certificados

Use certificados assinados obtidos a partir de uma autoridade de certificação ou da implementação de seu próprio procedimento de certificação, para autenticar uma conexão SSL. Você tem a opção de

autenticar o canal de telemetria ou o canal de telemetria e os clientes que se conectam a ele. As mensagens que fluem na conexão são criptografadas.

Antes de começar

Execute a tarefa, “Autenticando uma conexão de telemetria SSL usando certificados autoassinados” na página 509 antes de iniciar, para que o `PubSyncSSL.Java` funcione com uma conexão TCP/IP segura usando certificados autoassinados

Sobre esta tarefa

Nesta tarefa, modifique as tarefas “Autenticando o canal de telemetria” na página 511 e “Autenticando o canal de telemetria e os clientes” na página 512 em “Autenticando uma conexão de telemetria SSL usando certificados autoassinados” na página 509, para trabalhar com chaves certificadas por uma sequência de certificados.

É possível obter os certificados para esta tarefa a partir de uma autoridade de certificação ou usar um website como http://www.openca.org/ para obter certificados. As autoridades de certificação comercial geralmente fornecem certificados de avaliação para curto prazo sem encargos. Essa tarefa foi testada usando certificados obtidos de forma comercial.

Outra opção é construir seu próprio processo de certificação e executá-lo em seus próprios computadores, usando as ferramentas da web como https://www.openssl.org/.

Os armazenamentos confiáveis `cacerts` JRE não são usados nessa tarefa. É possível usar o armazenamento confiável `cacerts` JRE no cliente na tarefa, “Autenticando o canal de telemetria” na página 514, em vez de usar o armazenamento confiável especificado. A sequência de certificados pode ser assinada por uma autoridade de certificação bem conhecida que já tem seu certificado raiz no armazenamento `cacerts` no cliente. Nesse caso, não especifique um armazenamento confiável no cliente. Certifique-se de que, se houver vários JREs instalados no cliente, você gerencie o armazenamento `cacerts` correto.

Procedimento

1. Se você ainda não tiver feito isso, execute a tarefa, “Modificando PubSync.java para usar a SSL” na página 510, para modificar PubSync.java para usar SSL.
2. Configure o canal de telemetria e crie os keystores para usar SSL.

   Autentique apenas o canal de telemetria ou o canal e os clientes que se conectam a ele:

   - Execute a tarefa, “Autenticando o canal de telemetria” na página 514, para se conectar com a SSL, autenticando o canal de telemetria.
   - Execute a tarefa, “Autenticando o canal de telemetria e os clientes” na página 516, para se conectar com a SSL, autenticando o canal de telemetria e os clientes que se conectam a ele.
3. Pare e reinicie o serviço de telemetria (MQXR) para escolher as mudanças nas configurações do canal de telemetria.
4. Execute o programa cliente para ver se a configuração funciona.

Autenticando o canal de telemetria

Os clientes autenticam o canal de telemetria para criptografar o conteúdo das mensagens que fluem no canal e para assegurar que um cliente se conecte ao canal de telemetria correto. O servidor não autentica o cliente.

Sobre esta tarefa

É possível usar um número de editores de keystore diferentes para criar e gerenciar certificados. A tarefa usa o comando **keytool** da linha de comandos que faz parte do JRE. É possível usar a ferramenta da GUI **iKeyman**, que é enviada com o WebSphere MQ para procurar armazenamentos de chaves e gerar chaves Ative **iKeyman** usando o comando **strmqikm**.

Procedimento

1. Crie um canal de telemetria, `SSLCAOptClients` que requeira uma conexão SSL usando o assistente **Novo canal de telemetria**. O canal aceita clientes anônimos.

   Adapte a configuração do seu canal da sub-rotina de configuração a seguir. Não edite o arquivo de propriedades de telemetria diretamente; use o assistente.

   ```
   com.ibm.mq.MQXR.channel/SSLCAOptClients: \
   com.ibm.mq.MQXR.Port=8885;\
   com.ibm.mq.MQXR.Backlog=4096;\
   com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\CAServerOptKey.jks;\
   com.ibm.mq.MQXR.PassPhrase=password;\
   com.ibm.mq.MQXR.ClientAuth=OPTIONAL;\
   com.ibm.mq.MQXR.UserName=Admin;\
   com.ibm.mq.MQXR.StartWithMQXRService=true
   ```

2. Gere uma chave assinada pela CA para que o cliente autentique o canal de telemetria.

   a) Gere um par de chaves autoassinado para o canal de telemetria em um novo keystore, `SSServerOptKey.jks`:

   ```
   Keytool -genkey -noprompt -alias CAServerPrivate -keyalg RSA
           -dname "CN=mqttserverOpt.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore CAServerOptKey.jks -storepass password -keypass password
   ```

   O algoritmo de chave é configurado para RSA porque algumas autoridades de certificação exigem. O nome comum do certificado deve ser exclusivo, algumas autoridades de certificação não emitem chaves com nomes comuns idênticos.

   b) Crie uma certificate signing request (CSR) como um arquivo ASCII

   ```
   Keytool -certreq -noprompt -alias CAServer -file CAServerOptKey.csr
           -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore CAServerOptKey.jks -storepass password -keypass password
   ```

   c) Execute o software de autoridade de certificação ou efetue logon no seu website. Cole no conteúdo de `CAServerOptKey.csr` quando solicitado para o arquivo CSR.

   d) A autoridade de certificação retorna um ou dois certificados e um arquivo de resposta assinado como arquivos ASCII. Cole o conteúdo em dois ou três arquivos:

   Certificado raiz
   Cole-o em `CARoot.cer`

   Certificado intermediário
   Cole-o em `CAInter.cer`

   Arquivo de resposta do servidor assinado
   Cole-o em `CAServerOpt.rsp`

   O armazenamento de certificados JRE não é usado nesta tarefa. Se você recebeu um certificado raiz e uma resposta assinada da CA, use o certificado raiz e a resposta assinada nas etapas a seguir. Se você recebeu um certificado raiz e um intermediário, use o certificado intermediário e a resposta assinada.

   e) Receba a resposta do servidor assinada no keystore do servidor a partir do qual você emitiu a solicitação de certificado.

   Receber a resposta modifica o certificado autoassinado para que ele seja assinado pela CA. Se você olhar o certificado no keystore antes e depois de receber a resposta, o assinante muda. Se não, um erro será relatado pela ferramenta de gerenciamento de chaves. Antes de usar o certificado, inspecione-o e verifique se o assinante agora é a CA.

   ```
   Keytool -import -noprompt -alias CAServer -file CAServerOpt.rsp
           -keystore CAServerOptKey.jks -storepass password
   ```

   Em alguns software de gerenciamento de chaves, como **iKeyman**, você recebe, em vez de importar, os arquivos de resposta.

f) Importe o certificado CA no armazenamento confiável do cliente.

Importe o certificado intermediário se você recebeu dois certificados da CA ou o certificado raiz, se você recebeu somente um certificado.

Execute um dos dois procedimentos:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAClientTrust.jks -storepass password
```

Ou:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAClientTrust.jks -storepass password
```

Autenticando o canal de telemetria e os clientes

Os clientes autenticam o canal de telemetria e o canal de telemetria autentica clientes que estão se conectando a ele. As mensagens que fluem no canal são criptografadas.

Sobre esta tarefa

É possível usar um número de editores de keystore diferentes para criar e gerenciar certificados. A tarefa usa o comando **keytool** da linha de comandos que faz parte do JRE. É possível usar a ferramenta da GUI **iKeyman**, que é enviada com o WebSphere MQ para procurar armazenamentos de chaves e gerar chaves Ative **iKeyman** usando o comando **strmqikm**.

O canal de telemetria é configurado com um keystore diferente para aquele na tarefa, “Autenticando o canal de telemetria” na página 514. É possível usar o mesmo keystore e omitir a etapa “2” na página 516 para incluir chaves no keystore.

Procedimento

1. Crie um canal de telemetria, `SSLCAReqClients` que requeira uma conexão SSL usando o assistente **Novo canal de telemetria**. O canal aceita apenas os clientes autenticados.

   Adapte a configuração do seu canal da sub-rotina de configuração a seguir. Não edite o arquivo de propriedades de telemetria diretamente; use o assistente.

   ```
   com.ibm.mq.MQXR.channel/SSLCAReqClients: \
   com.ibm.mq.MQXR.Port=8886;\
   com.ibm.mq.MQXR.Backlog=4096;\
   com.ibm.mq.MQXR.KeyFileName=C:\\Certificates\\CAServerReqKey.jks;\
   com.ibm.mq.MQXR.PassPhrase=password;\
   com.ibm.mq.MQXR.ClientAuth=REQUIRED;\
   com.ibm.mq.MQXR.UserName=Admin;\
   com.ibm.mq.MQXR.StartWithMQXRService=true
   ```

2. Gere uma chave assinada pela CA para que o cliente autentique o canal de telemetria.

   a) Gere um par de chaves autoassinado para o canal de telemetria em um novo keystore, `CAServerReqKey.jks`:

   ```
   Keytool -genkey -noprompt -alias CAServerPrivate -keyalg RSA
           -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore CAServerReqKey.jks -storepass password -keypass password
   ```

   O algoritmo de chave é configurado para RSA porque algumas autoridades de certificação exigem. O nome comum do certificado deve ser exclusivo, algumas autoridades de certificação não emitem chaves com nomes comuns idênticos.

   b) Crie uma certificate signing request (CSR) como um arquivo ASCII

   ```
   Keytool -certreq -noprompt -alias CAServer -file CAServerReqKey.csr
           -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
           -keystore CAServerReqKey.jks -storepass password -keypass password
   ```

c) Execute o software de autoridade de certificação ou efetue logon no seu website. Cole no conteúdo de `CAServerReqKey.csr` quando solicitado para o arquivo CSR.

d) A autoridade de certificação retorna um ou dois certificados e um arquivo de resposta assinado como arquivos ASCII. Cole o conteúdo em dois ou três arquivos:

Certificado raiz
Cole-o em `CARoot.cer`

Certificado intermediário
Cole-o em `CAInter.cer`

Arquivo de resposta do servidor assinado
Cole-o em `CAServerReq.rsp`

O armazenamento de certificados JRE não é usado nesta tarefa. Se você recebeu um certificado raiz e uma resposta assinada da CA, use o certificado raiz e a resposta assinada nas etapas a seguir. Se você recebeu um certificado raiz e um intermediário, use o certificado intermediário e a resposta assinada.

e) Receba a resposta do servidor assinada no keystore do servidor a partir do qual você emitiu a solicitação de certificado.

Receber a resposta modifica o certificado autoassinado para que ele seja assinado pela CA. Se você olhar o certificado no keystore antes e depois de receber a resposta, o assinante muda. Se não, um erro é relatado pela ferramenta de gerenciamento de chaves. Antes de usar o certificado, inspecione-o e verifique se o assinante agora é a CA.

```
Keytool -import -noprompt -alias CAServer -file CAServerReq.rsp
        -keystore CAServerReqKey.jks -storepass password
```

Em alguns software de gerenciamento de chaves, como **iKeyman**, você recebe, em vez de importar, os arquivos de resposta.

f) Importe o certificado CA no armazenamento confiável do cliente.

Importe o certificado intermediário se você recebeu dois certificados da CA ou o certificado raiz, se você recebeu somente um certificado.

Execute um dos dois procedimentos:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAClientTrust.jks -storepass password
```

Ou:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAClientTrust.jks -storepass password
```

3. Gere uma chave assinada pela CA para que o canal de telemetria autentique clientes.

a) Gere um par de chaves autoassinado para os clientes em um novo keystore, `CAClientKey.jks`:

```
Keytool -genkey -noprompt -alias CAClientPrivate -keyalg RSA
        -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
        -keystore CAClientKey.jks -storepass password -keypass password
```

O algoritmo de chave é configurado para RSA porque algumas autoridades de certificação exigem. O nome comum do certificado deve ser exclusivo, algumas autoridades de certificação não emitem chaves com nomes comuns idênticos.

b) Crie uma certificate signing request (CSR) como um arquivo ASCII

```
Keytool -certreq -noprompt -alias CAClient -file CAClientKey.csr
        -dname "CN=mqttserverReq.ibm.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
        -keystore CAClientKey.jks -storepass password -keypass password
```

c) Execute o software de autoridade de certificação ou efetue logon no seu website. Cole no conteúdo de `CAClientKey.csr` quando solicitado para o arquivo CSR.

d) A autoridade de certificação retorna um ou dois certificados e um arquivo de resposta assinado como arquivos ASCII. Cole o conteúdo em dois ou três arquivos:

Certificado raiz
Cole-o em `CARoot.cer`

Certificado intermediário
Cole-o em `CAInter.cer`

Arquivo de resposta assinado pelo cliente
Cole-o em `CAClient.rsp`

O armazenamento de certificados JRE não é usado nesta tarefa. Se você recebeu um certificado raiz e uma resposta assinada da CA, use o certificado raiz e a resposta assinada nas etapas a seguir. Se você recebeu um certificado raiz e um intermediário, use o certificado intermediário e a resposta assinada.

e) Receba a resposta assinada do cliente no keystore do cliente a partir do qual você emitiu a solicitação de certificado.

Receber a resposta modifica o certificado autoassinado para que ele seja assinado pela CA. Se você olhar o certificado no keystore antes e depois de receber a resposta, o assinante muda. Se não, um erro é relatado pela ferramenta de gerenciamento de chaves. Antes de usar o certificado, inspecione-o e verifique se o assinante agora é a CA.

```
Keytool -import -noprompt -alias CAClient -file CAClient.rsp
        -keystore CAClientKey.jks -storepass password
```

Em alguns software de gerenciamento de chaves, como **iKeyman**, você recebe, em vez de importar, os arquivos de resposta.

f) Importe o certificado de CA no keystore do servidor.

Importe o certificado intermediário se você recebeu dois certificados da CA ou o certificado raiz, se você recebeu somente um certificado.

Execute um dos dois procedimentos:

```
keytool -import -alias CAInter -file CAInter.cer
        -keystore CAServerReqKey.jks -storepass password
```

Ou:

```
keytool -import -alias CARoot -file CARoot.cer
        -keystore CAServerReqKey.jks -storepass password
```

Criando seu primeiro aplicativo publicador do MQ Telemetry Transport usando C

As etapas para criar um aplicativo publicador cliente MQTT são descritas no modo tutorial. Cada linha de código C é explicada. No final da tarefa, você terá criado um editor MQTT.

Antes de começar

O aplicativo cliente desenvolvido usa as bibliotecas do cliente MQTT v3 C. O aplicativo se conecta ao daemon do WebSphere MQ Telemetry para dispositivos para publicar mensagens. Consulte Criando seu primeiro publicador para obter um exemplo de um cliente que se comunica com o WebSphere MQ Telemetry

Sobre esta tarefa

O exemplo é um aplicativo de publicação, `pubsync.c`. O programa `pubsync.c` publica uma mensagem com a carga útil `Hello World!` para o tópico `MQTT Examplee` espera a confirmação de que a publicação foi entregue para o daemon

Para simplificar, os códigos de retorno de algumas funções usadas não são testados para conclusão correta. No código de produção, os códigos de retorno podem ser verificados para assegurar que o programa se comporte conforme esperado. As ações deverão ser tomadas caso ocorra um erro inesperado.

Ao configurar um assinante para `MQTT Example` é possível verificar se o aplicativo funciona.

Use o ambiente de desenvolvimento C selecionado para desenvolver, construir e executar o cliente. Se preferir, será possível copiar o código diretamente dos exemplos.

Procedimento

1. Crie um novo arquivo de origem vazio, `pubsync.c`
2. Crie um arquivo `settings.h`. Copie o código na Figura 2 no arquivo.

   Todos os parâmetros usados no programa são definidos em `settings.h`. É possível substituir os valores mudando os valores no arquivo.
3. As etapas a seguir explicam o código. Siga as etapas ou copie o código de Figura 1 no `pubsync.c`.
4. Inclua as instruções include do arquivo de cabeçalho nas bibliotecas padrão necessárias e nos arquivos `MQTTClient.h` e `settings.h`.

   ```
   #include "stdio.h"
   #include "stdlib.h"
   #include "string.h"
   #include "MQTTClient.h"
   #include "settings.h"
   ```
5. Inicie a definição da função `main()`.

   ```
   int main(int argc, char* argv[])
   {
   ```
6. Defina as variáveis locais usadas no programa.

   ```
   MQTTClient client;
   MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
   MQTTClient_message pubmsg;
   MQTTClient_deliveryToken token;
   int rc;
   ```

   Nota: As opções de conexão são requeridas pela função `MQTTClient_connect`. `MQTTClient_connectOptions_initializer` contém as opções padrão.
7. Crie um cliente.

   ```
   MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
   ```

   - `& client` é um ponteiro para uma manipulação do cliente recém-criado. Quando esta função retorna com um código de retorno 0, ela contém um identificador para o novo cliente. O exemplo presume o êxito. Teste a conclusão correta do código de erro no código de produção.
   - ADDRESS é o URI da porta MQTT que o daemon monitora para solicitações de conexão do cliente recebidas.
   - `CLIENTID` é o nome usado para identificar o cliente para o daemon. Cada cliente ativo deve ter um nome exclusivo. Se você duplicar um identificador de cliente nos dois clientes em execução, uma exceção será lançada nos dois clientes e um deles será finalizado. O nome é usado pelo daemon para reconhecer que um cliente tjhat está reconectando após uma desconexão, consulte O identificador do cliente.
   - `MQTTCLIENT_PERSISTENCE_NONE` especifica que o estado do cliente é mantido na memória e é perdido se ocorrer uma falha no sistema. `MQTTCLIENT_PERSISTENCE_DEFAULT` especifica a persistência baseada em sistema de arquivos, fornecendo alguma proteção contra falhas. Para obter aplicativos mais especializados, é possível usar `MQTTCLIENT_PERSISTENCE_USER`, que fornece uma interface para você implementar seu próprio mecanismo de persistência. Para obter mais detalhes, consulte a documentação da API para `MQTTClientPersistence.h`. Se a

persistência for necessária, é uma questão de design do aplicativo. Para obter mais detalhes, consulte Persistência de mensagem

- A porta TCP/IP do daemon padrão para o MQTT é 1883 No exemplo, o endereço padrão é configurado como `tcp://localhost:1883`.
- Até você chamar a função `MQTTClient_connect`, nenhum processamento de mensagem ocorrerá.

8. Conecte o cliente ao daemon.

```
if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
     printf("Failed to connect, return code %d\n", rc);
     exit(-1);
}
```

- A função `MQTTClient_connect` é chamada, passando o identificador do cliente e um ponteiro para as opções de conexão como argumentos.
- O código de retorno da chamada `MQTTClient_connect` é testado para certificar-se de que a solicitação de conexão foi bem-sucedida.
- Se o `MQTTClient_connect` falhar, o programa é encerrado com um código de erro -1.
- Depois que o aplicativo se conecta, é possível iniciar a publicação e a assinatura.
- Uma pequena mensagem "keep-alive" é enviada a cada 20 segundos para evitar que a conexão TCP/IP seja encerrada. Essa opção é configurada por `conn_opts.keepAliveInterval`.
- A sessão é iniciada sem verificar a conclusão das mensagens em andamento restantes de uma conexão anterior porque `conn_opts.cleansession` está configurado como true. Para obter mais detalhes, consulte Limpar Sessões
- Nenhuma mensagem de último desejo e testamento é criada para a conexão. Para obter mais detalhes, consulte Última vontade e testamento.

9. Preencha a estrutura *MQTTClient_message* com os dados para definir a carga útil da mensagem e seus atributos.

```
pubmsg.payload = PAYLOAD;
pubmsg.payloadlen = strlen(PAYLOAD);
pubmsg.qos = QOS;
pubmsg.retained = 0;
```

- PAYLOAD é nosso conteúdo da mensagem.
- O exemplo usa uma carga útil de sequência, mas as cargas úteis do MQTT são matrizes de bytes O comprimento da sequência é necessário para especificar o tamanho da carga útil.
- O exemplo publica uma mensagem QoS=1, portanto, configure o valor adequadamente
- O atributo retido é configurado para false (0), pois a mensagem não deve ser retida pelo daemon. Para obter mais detalhes, consulte Publicações retidas

10. Publique a mensagem.

```
MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
```

- A função publicar especifica o cliente, o tópico e a carga útil a ser enviada ao daemon.
- `TOPIC` é definido em `settings.h` como `MQTT Example`..
- A função também é transmitida a um ponteiro para uma *MQTTClient_deliveryToken*. Esse ponteiro é preenchido com um token que representa a mensagem quando a função retorna.
- A mensagem agora é transferida com segurança para o cliente MQTT, mas ainda não foi transferida para o daemon. Se a mensagem tiver QoS=1 ou 2, a mensagem é armazenada localmente, caso o cliente falhe antes que a entrega seja concluída.
- Essa função retorna um código de erro cuja conclusão correta é possível testar no código de produção.

11. Aguarde a confirmação do servidor.

```
rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
```

- O exemplo `pubsync.c` espera uma confirmação do servidor, que confirma que a mensagem foi entregue.
- Os argumentos do cliente e do token identificam a mensagem específica que o programa está aguardando ser concluída.
- `TIMEOUT` limita quanto tempo o programa aguarda a mensagem para concluir a entrega. A tarefa Criando um publicador assíncrono para o MQ Telemetry Transport usando C mostra como receber confirmações sem esperar usando funções de retorno de chamada.
- Essa função retorna um código de erro que pode ter a conclusão correta testada no código de produção.

12. Desconecte o cliente do daemon.

```
MQTTClient_disconnect(client, 10000);
```

- O cliente se desconecta do servidor e aguarda quaisquer funções de retorno de chamada (não usado neste exemplo) para que as mensagens em andamento sejam concluídas.
- O segundo argumento especifica um tempo limite em modo quiesce em milissegundos. O exemplo aguarda até 10 segundos para concluir qualquer trabalho diferente que deve ser realizado antes da desconexão.
- Essa função retorna um código de erro cuja conclusão correta deve ser testada no código de produção.

13. Libere a memória usada pelo cliente e encerre o programa.

```
MQTTClient_destroy(&client);
}
```

Resultados

Para ver as publicações enviadas por este cliente, crie um assinante para o tópico `MQTT Example`. Para obter mais detalhes, consulte Criando um assinante para o MQ Telemetry Transport usando C

Exemplo

Figura 1 é uma listagem completa do código descrito em Procedimento. O arquivo `settings.h` em Figura 2 permite que você mude os parâmetros padrão usados em `pubsync.c`.

```
#include "stdio.h"
#include "stdlib.h"
#include "MQTTClient.h"
#include "settings.h"

int pubsync_main(int argc, char* argv[]) {
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    MQTTClient_message pubmsg;
    MQTTClient_deliveryToken token;
    int rc;

    MQTTClient_create(&client, ADDRESS, CLIENTID,  MQTTCLIENT_PERSISTENCE_NONE, NULL);
    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
          printf("Failed to connect, return code %d\n", rc);
         exit(-1);
    }
    pubmsg.payload = PAYLOAD;
    pubmsg.payloadlen = strlen(PAYLOAD);
    pubmsg.qos = QOS;
    pubmsg.retained = 0;
    MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
    printf("Waiting for up to %d seconds for publication of %s\n"
          "on topic %s for client with ClientID: %s\n",
          TIMEOUT/1000, PAYLOAD, TOPIC, CLIENTID);
    rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
    printf("Message with delivery token %d delivered\n", token);
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
}
```

Figura 111. pubsync.c

```
#define ADDRESS   "tcp://localhost:1883"
#define CLIENTID  "ExampleClientPub"
#define TOPIC     "MQTT Example"
#define PAYLOAD   "Hello World!"
#define QOS      1
#define TIMEOUT   10000L
```

Figura 112. settings.h

Criando um publicador assíncrono para o MQ Telemetry Transport usando C

As etapas para criar um aplicativo publicador assíncrono do cliente MQTT são descritas no modo tutorial. Cada linha de código C é explicada. No final da tarefa, você terá criado um editor assíncrono MQTT.

Nesta tarefa, você segue um tutorial para modificar seu primeiro aplicativo de publicador. As modificações permitem que o aplicativo envie publicações sem esperar reconhecimentos de entrega. As confirmações de entrega são recebidas por uma função de retorno de chamada que você criar.

Antes de começar

O aplicativo cliente desenvolvido usa as bibliotecas do cliente MQTT v3 C. O aplicativo se conecta ao daemon do WebSphere MQ Telemetry para dispositivos para publicar mensagens. Consulte Criando seu primeiro publicador para obter um exemplo de um cliente que se comunica com o WebSphere MQ Telemetry

Sobre esta tarefa

O exemplo é um aplicativo de publicação, pubasync.c. O programa pubasync.c publica uma mensagem com a carga útil `Hello World!` para o tópico `MQTT Example`, sem esperar a confirmação de que a publicação foi entregue para o daemon As confirmações de entrega são recebidas em uma função de retorno de chamada, `MQTTClient_deliveryComplete`.

Para simplificar, os códigos de retorno de algumas funções usadas não são testados para conclusão correta. No código de produção, os códigos de retorno podem ser verificados para assegurar que o programa se comporte conforme esperado. As ações deverão ser tomadas caso ocorra um erro inesperado.

Ao configurar um assinante para MQTT Example, é possível verificar se o aplicativo funciona.

Use o ambiente de desenvolvimento C selecionado para desenvolver, construir e executar o cliente.

As etapas em Procedimento modificam o aplicativo pubsync.c a partir do “Criando seu primeiro aplicativo publicador do MQ Telemetry Transport usando C” na página 518. Se preferir, será possível copiar o código diretamente dos exemplos.

Procedimento

1. Crie um novo arquivo de origem e vazio, callback.h.
2. Copie o código na Figura 2 no arquivo.
   - callback.h declara os três métodos de retorno de chamada necessários para a operação do cliente assíncrono.
   - Uma variável, *deliveredtoken*, também é declarada. Isso é acessada pelo programa principal e o retorno de chamada em diversos encadeamentos de execução. Portanto, é declarado como volátil. Ao usar os retornos de chamadas, tome cuidado para assegurar-se de que as variáveis relevantes sejam acessadas em uma maneira thread-safe.
3. Crie um novo arquivo de origem e vazio, callback.c.
4. Copie o código na Figura 3 no arquivo.
   - callback.c implementa os três métodos de retorno de chamada usados pelo cliente para operação assíncrona, delivered, msgarrvd e connlost.
5. Inclua uma instrução de inclusão para callback.h após os outros incluírem em pubasync.c.

   ```
   #include "callback.h"
   ```

6. Copie o conteúdo de pubsync.c em um novo arquivo, pubasync.c.
7. Logo antes da chamada de função MQTTClient_connect em pubasync.c, configure os métodos de retorno de chamada para o cliente.

   ```
   MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);
   ```

   - Deve-se especificar três funções de retorno de chamada. Essas funções são implementadas em callback.c.
   - MQTTClient_messageArrived é chamado quando uma mensagem é enviada para o cliente devido a uma assinatura correspondente. Este deverá retornar true quando a mensagem recebida foi recebida com sucesso pelo aplicativo cliente. Ao retornar false informa ao cliente que seu aplicativo teve um problema ao receber a mensagem.
   - MQTTClient_connectionLost é chamado quando o cliente perde sua conexão com o servidor.
   - MQTTClient_deliveryComplete é chamado quando uma mensagem QoS1 ou QoS2 chega e é confirmada pelo servidor. Ele não é chamado para as mensagens QoS0. No exemplo, esta função salva o token da mensagem entregue em *deliveredtoken* para indicar que uma mensagem chegou.
   - MQTTClient_setCallbacks deve ser chamado enquanto o cliente está desconectado do servidor.
   - O segundo argumento permite que você transmita informações contextuais para as funções de retorno de chamada. Isso não é usado no exemplo, portanto, é definido como NULL.
8. Imediatamente antes da chamada para MQTTClient_publishMessage, limpe *deliveredtoken*. MQTTClient_deliveryComplete configura *deliveredtoken* quando um token é recebido.

   ```
   deliveredtoken = 0;
   ```

9. Remova a chamada MQTTClient_waitForCompletion e a instrução printf que a segue e substitua por um loop que aguarda uma correspondência do token original e do token recebido no retorno de chamada.

```
while(deliveredtoken != token);
```

Este é um exemplo e não lida com um número de situações que devem ser acomodadas no código de produção de design. Essas situações incluem:

- Caso a entrega não seja concluído, um tempo limite poderá ser implementado
- Várias mensagens podem estar em andamento. O programa de amostra apenas permite que um token de entrega seja verificado por vez.

10. Desconecte o cliente do daemon.

```
MQTTClient_disconnect(client, 10000);
```

- O cliente se desconecta do servidor e aguarda quaisquer funções de retorno de chamada para mensagens em andamento serem concluídas.
- O segundo argumento especifica um tempo limite em modo quiesce em milissegundos. O exemplo aguarda até 10 segundos para concluir qualquer trabalho diferente que deve ser realizado antes da desconexão.
- Essa função retorna um código de erro cuja conclusão correta deve ser testada no código de produção.

11. Libere a memória usada pelo cliente e encerre o programa.

```
MQTTClient_destroy(&client);
}
```

Resultados

Para ver a publicação enviada por este cliente, crie um assinante para o tópico `MQTT Example`. Para obter mais detalhes, consulte Criando um assinante para o MQ Telemetry Transport

Exemplo

`pubasync.c`, `callbacks.c` e `callbacks.h` são listagens completas do código descrito em Procedimento.

```
#include "stdio.h"
#include "stdlib.h"
#include "MQTTClient.h"
#include "settings.h"
#include "callback.h"

int main(int argc, char* argv[]) {
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    MQTTClient_message pubmsg;
    MQTTClient_deliveryToken token;
    int rc;

    MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
    MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);
    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
        printf("Failed to connect, return code %d\n", rc);
        exit(-1);
    }
    pubmsg.payload = PAYLOAD;
    pubmsg.payloadlen = strlen(PAYLOAD);
    pubmsg.qos = QOS;
    pubmsg.retained = 0;
    deliveredtoken = 0;
    MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
    printf("Waiting for publication of %s\n"
           "on topic %s for client with ClientID: %s\n", PAYLOAD, TOPIC, CLIENTID);
    while(deliveredtoken != token);
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
}
```

Figura 113. pubasync.c

```
MQTTClient_deliveryComplete delivered;
MQTTClient_messageArrived msgarrvd;
MQTTClient_connectionLost connlost;

extern volatile MQTTClient_deliveryToken deliveredtoken;
```

Figura 114. callback.h

```
#include "MQTTClient.h"

volatile MQTTClient_deliveryToken deliveredtoken;

void delivered(void *context, MQTTClient_deliveryToken dt)
{
    printf("Message with token value %d delivery confirmed\n", dt);
    deliveredtoken = dt;
}

int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message)
{
    int i;
    char* payloadptr;

    printf("Message arrived\n");
    printf("     topic: %s\n", topicName);
    printf("   message: ");

    payloadptr = message->payload;
    for(i=0; i<message->payloadlen; i++) {
        putchar(*payloadptr++);
    }
    putchar('\n');
    MQTTClient_freeMessage(&message);
    free(topicName);
    return 1;
}

void connlost(void *context, char *cause)
{
    printf("\nConnection lost\n");
    printf("     cause: %s\n", cause);
}
```

Figura 115. `callback.c`

```
#define ADDRESS    "tcp://localhost:1883"
#define CLIENTID   "ExampleClientPub"
#define TOPIC      "MQTT Example"
#define PAYLOAD    "Hello World!"
#define QOS        1
#define TIMEOUT    10000L
```

Figura 116. `settings.h`

Criando um assinante para o MQ Telemetry Transport usando C

As etapas para criar um aplicativo assinante cliente MQTT são descritas no modo tutorial. Cada linha de código C é explicada. No final da tarefa, você criará um assinante MQTT.

Antes de começar

O aplicativo cliente desenvolvido usa as bibliotecas do cliente MQTT v3 C. O aplicativo se conecta ao daemon do WebSphere MQ Telemetry para dispositivos para publicar mensagens. Consulte Criando seu primeiro publicador para obter um exemplo de um cliente que se comunica com o WebSphere MQ Telemetry

Sobre esta tarefa

O exemplo é um aplicativo de assinante, `subscribe.c`. O programa `subscribe.c` assina o tópico `MQTT Example` e aguarda publicações que correspondam à assinatura até que o usuário termine o programa.

Um assinante cria uma assinatura para um tópico e aguarda mensagens que correspondem ao tópico de assinatura. As mensagens publicadas enquanto o cliente está desconectado e que correspondem a uma assinatura criada anteriormente pelo cliente, podem ser recebidas quando o cliente se reconecta. O serviço ou daemon de telemetria do WebSphere MQ para dispositivos reconhece um cliente que foi conectado anteriormente pelo identificador de cliente. Para obter mais informações, consulte O identificador de cliente. O atributo booleano `MQTTClient_connectOptions.cleansession` controla

se as publicações enviadas anteriormente são recebidas ou não. Para obter mais detalhes, consulte “Sessões limpas” na página 535.

Para simplificar, os códigos de retorno de algumas funções usadas não são testados para conclusão correta. No código de produção, os códigos de retorno podem ser verificados para assegurar que o programa se comporte conforme esperado. A ação apropriada pode ser tomadas caso ocorra um erro inesperado.

É possível usar os programas de exemplo de publicação descritos anteriormente para enviar publicações correspondentes para o daemon do WebSphere MQ Telemetry para dispositivos Como alternativa, use o WebSphere MQ Explorer para criar publicações de teste no tópico `MQTT Example` se você desejar conectar o cliente a um canal do WebSphere MQ Telemetry .

As instruções em Procedimento supõem que você já criou os arquivos `callback.c, callback.h` e `settings.h` em uma das tarefas anteriores.

Use o ambiente de desenvolvimento C selecionado para desenvolver, construir e executar o cliente. Se preferir, será possível copiar o código diretamente dos exemplos.

Procedimento

1. Crie uma cópia de `settings.h` para este exemplo e mude a instrução de definição *CLIENTID* para o seguinte:

   ```
   #define CLIENTID "ExampleClientSub"
   ```

   - Se dois clientes com o mesmo ID tentarem se conectar a um único servidor, um deles será obrigatoriamente desconectado. Normalmente, a nova tentativa de conexão é bem-sucedida e a conexão mais antiga é desconectada.
   - Alterar o `ClientID` permite usar os exemplos de publicação desenvolvidos anteriormente para enviar mensagens para este assinante.
2. Crie um novo arquivo de origem vazio, `subscribe.c`.
3. As etapas a seguir explicam o código. Siga as etapas ou copie o código de Figura 117 na página 530 no arquivo `subscribe.c`.
4. Inclua as instruções include do arquivo de cabeçalho nas bibliotecas padrão necessárias e nos arquivos `MQTTClient.h` e `settings.h`.

   ```
   #include "stdio.h"
   #include "stdlib.h"
   #include "MQTTClient.h"
   #include "settings.h"
   ```

5. Inicie a definição da função `main()`.

   ```
   int main(int argc, char* argv[]) {
   ```

6. Defina as variáveis locais usadas no programa.

   ```
   MQTTClient client;
   MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
   MQTTClient_deliveryToken token;
   int rc;
   ```

   As opções de conexão são requeridas pela função `MQTTClient_connect`. `MQTTClient_connectOptions_initializer` contém as opções padrão.
7. Crie um cliente.

   ```
   MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
   ```

   - `& client` é um ponteiro para uma manipulação do cliente recém-criado. Quando essa função retorna com um código de retorno 0, o ponteiro contém um identificador para o novo cliente. O exemplo presume o êxito. O código de erro pode ser testado para conclusão correta no código de produção.

- ADDRESS é o URI da porta MQTT que o daemon monitora para solicitações de conexão do cliente recebidas.
- CLIENTID é o nome usado para identificar o cliente para o daemon. Cada cliente ativo deve ter um nome exclusivo. Se você duplicar um identificador de cliente nos dois clientes em execução, uma exceção será lançada nos dois clientes e um deles será finalizado. O nome é usado pelo daemon para reconhecer que um cliente está reconectando após uma desconexão, consulte O identificador do cliente.
- MQTTCLIENT_PERSISTENCE_NONE especifica que o estado do cliente é mantido na memória e é perdido se ocorrer uma falha no sistema. MQTTCLIENT_PERSISTENCE#_DEFAULT especifica a persistência baseada no sistema de arquivos, fornecendo alguma proteção contra falhas. Para obter aplicativos mais especializados, é possível usar MQTTCLIENT_PERSISTENCE_USER, que fornece uma interface para você implementar seu próprio mecanismo de persistência. Se a persistência for necessária, é uma questão de design do aplicativo. Para obter mais detalhes, consulte Persistência de mensagem
- A porta TCP/IP do daemon padrão para o MQTT é 1883 No exemplo, o endereço padrão é configurado como `tcp://localhost:1883`.
- Até você chamar a função `MQTTClient_connect`, nenhum processamento de mensagem ocorrerá.

8. Conecte o cliente ao daemon

```
if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
    printf("Failed to connect, return code %d\n", rc);
    exit(-1);
}
```

   - A função `MQTTClient_connect` é chamada, passando o identificador do cliente e um ponteiro para as opções de conexão como argumentos.
   - O código de retorno da chamada `MQTTClient_connect` é testado para certificar-se de que a solicitação de conexão foi bem-sucedida.
   - Se a chamada de conexão falhar, o programa será finalizado com um código de erro -1.
   - Depois que o aplicativo se conecta, ele pode iniciar a publicação e a assinatura.
   - Uma pequena mensagem "keep-alive" é enviada a cada 20 segundos para evitar que a conexão TCP/IP seja encerrada. Essa opção é configurada por `conn_opts.keepAliveInterval`.
   - A sessão é iniciada sem verificar a conclusão das mensagens em andamento restantes de uma conexão anterior porque *conn_opts.cleansession* está configurado como true. Para obter mais detalhes, consulte Limpar Sessões
   - Nenhuma mensagem de último desejo e testamento é criada para a conexão. Para obter mais detalhes, consulte Última vontade e testamento

9. Assine o tópico.

```
MQTTClient_subscribe(client, TOPIC, QOS);
```

   - Use a função `MQTTClient_subscribe` para assinar o aplicativo cliente para o tópico selecionado. O nome do tópico pode incluir caracteres curinga. Para obter mais detalhes, consulte “Sequências e filtros de tópicos em clientes MQTT” na página 550.
   - A configuração de QoS determina a qualidade de serviço máxima que é aplicada a mensagens enviadas para este assinante. O servidor envia mensagens no valor inferior dessa configuração e a configuração QoS para a mensagem original.
   - Essa função retorna um código de erro que pode ter a conclusão correta testada no código de produção.

10. Aguarde em um loop até que o usuário insira um caractere 'Q' a partir do teclado.

```
do {
    ch = getchar();
    } while(ch!='Q' && ch != 'q');
```

O programa agora aguarda a chegada de mensagens. Neste exemplo, toda a manipulação de mensagem ocorre na função de retorno de chamada `MQTTClient_messageArrived`. Para obter mais detalhes, consulte “Como receber mensagens” na página 529.

11. Desconecte o cliente do daemon.

```
MQTTClient_disconnect(client, 10000);
```

- O cliente se desconecta do servidor e aguarda quaisquer funções de retorno de chamada (não usado neste exemplo) para que as mensagens em andamento sejam concluídas.
- O segundo argumento especifica um tempo limite em modo quiesce em milissegundos. O exemplo aguarda até 10 segundos para concluir qualquer outro trabalho que ele deve executar antes de desconectar.
- Essa função retorna um código de erro que pode ter a conclusão correta testada no código de produção.

12. Libere a memória usada pelo cliente e encerre o programa.

```
MQTTClient_destroy(&client);
}
```

Como receber mensagens

Sobre esta tarefa

Quando as mensagens chegam do servidor, a função `MQTTClient_messageArrived` é iniciada. As etapas a seguir explicam o código.

Procedimento

1. Inicie a definição da função de retorno de chamada. Essa definição deve corresponder ao modelo de função `MQTTClient_messageArrived`.

```
int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message) {
```

- `context` fornece acesso ao contexto transmitido para a biblioteca do cliente quando a função `MQTTClient_setCallbacks` foi chamada. Essa função não é usada no exemplo.
- `topicName` é um ponteiro para o tópico para o qual a mensagem recebida é publicada. Se você tiver assinado usando caracteres curinga, esse parâmetro identificará o tópico específico usado para a mensagem.
- `topicLen` é o comprimento da sequência de tópicos. Esta opção é fornecida para usuários que devem integrar caracteres NULL em sequências de tópicos.
- `message` é um ponteiro para a estrutura `MQTTClient_message` que contém a carga útil e os atributos da mensagem.

2. Defina as variáveis locais usadas.

```
int i;
char* payloadptr;
```

Essas variáveis são usadas no exemplo para imprimir a carga útil por iteração sobre ela.

3. Imprima uma mensagem, exibindo o tópico e a carga útil da mensagem

```
printf("Message arrived\n");
printf("     topic: %s\n",topicName);
printf("   message: ");
payloadptr = message->payload;
for(i=0; i<message->payloadlen; i++){
    putchar(*payloadptr++);
    }
putchar('\n');
```

- O exemplo assume que a carga útil recebida é uma sequência de caracteres imprimíveis.
- Uma carga útil MQTT é uma matriz de bytes. O aplicativo é responsável por interpretar seu significado.

4. Libere a memória usada para armazenar a mensagem.

```
MQTTClient_freeMessage(&message);
MQTTClient_free(topicName);
```

- No exemplo, toda a manipulação de mensagem ocorre na função de retorno de chamada.
- Assegure-se de que as funções de retorno de chamada sejam curtas e retorne o controle para seu encadeamento de chamada assim que possível.
- O ponteiro da mensagem é transmitido para a manipulação da parte principal do programa.
- O programa principal deve liberar a memória usada pela mensagem quando o processamento é concluído. `MQTTClient_freeMessage()` é uma função conveniente que retorna os dois blocos de memória usados para manter a estrutura *MQTTClient_message* e a carga útil da mensagem de volta no sistema. A memória alocada para o `topicName` deve ser liberada separadamente conforme mostrado.

5. Retorne um valor true quando o retorno de chamada manipulou com êxito a mensagem

```
    return 1;
}
```

- Retornar um valor true indica que a biblioteca do cliente pode tratar a mensagem como entregue com êxito.
- Se a função de retorno de chamada não puder processar corretamente a mensagem, um valor falso será retornado. Por exemplo, se o retorno de chamada estiver colocando mensagens em uma fila para que o programa principal processe e a fila estiver cheia, retornar false seria apropriado.
- Para mensagens QoS1 e QoS2, retornar um valor false indica que a mensagem não foi entregue e outras tentativas de entregar serão feitas.

Código de exemplo

```
#include "stdio.h"
#include "stdlib.h"
#include "MQTTClient.h"
#include "settings.h"
#include "callback.h"

int main(int argc, char* argv[]) {
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    int rc;
    int ch;

    MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);

    MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);

    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
        printf("Failed to connect, return code %d\n", rc);
        exit(-1);
        }
    printf("Subscribing to topic %s\nfor client %s using QoS%d\n\n"
           "Press Q<Enter> to quit\n\n", TOPIC, CLIENTID, QOS);

    MQTTClient_subscribe(client, TOPIC, QOS);
    do {
        ch = getchar();
        } while(ch!='Q' && ch != 'q');
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
}
```

Figura 117. `subscriber.c`

```
#include "MQTTClient.h"

volatile MQTTClient_deliveryToken deliveredtoken;

void delivered(void *context, MQTTClient_deliveryToken dt) {
    printf("Message with token value %d delivery confirmed\n", dt);
    deliveredtoken = dt;
}

int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message) {
    int i;
    char* payloadptr;

    printf("Message arrived\n");
    printf("     topic: %s\n", topicName);
    printf("   message: ");

    payloadptr = message->payload;
    for(i=0; i<message->payloadlen; i++) {
        putchar(*payloadptr++);
        }
    putchar('\n');
    MQTTClient_freeMessage(&message);
    MQTTClient_free(topicName);
    return 1;
}
void connlost(void *context, char *cause) {
    printf("\nConnection lost\n");
    printf("     cause: %s\n", cause);
}
```

Figura 118. callback.h

```
#define ADDRESS    "tcp://localhost:1883"
#define CLIENTID   "ExampleClientSub"
#define TOPIC      "MQTT Example"
#define PAYLOAD    "Hello World!"
#define QOS        1
#define TIMEOUT    10000L
```

Figura 119. settings.h

Conceitos de programação do cliente MQTT

Os conceitos descritos nesta seção o ajudam a entender as bibliotecas do cliente C do Java, JavaScript para a versão 3.1 do MQTT protocol. Os conceitos complementam a documentação da API que acompanha as bibliotecas do cliente.

O `com.ibm.micro.client.mqttv3` contém as classes que fornecem os métodos públicos para as bibliotecas do cliente para o protocolo MQTT version 3.1 Uma versão do pacote `com.ibm.micro.client.mqttv3` e o pacote de acompanhamento que implementam o protocolo para Java SE e ME, é fornecido com a instalação do IBM WebSphere MQ Telemetry. Para obter a versão mais recente das bibliotecas do cliente MQTT (Java, JavaScript) e visualizar ou fazer download da documentação da API, consulte "Referência de programação do cliente MQTT".

Para desenvolver e executar um cliente MQTT, é necessário copiar ou instalar esses pacotes no dispositivo do cliente. Não é necessário instalar um tempo de execução de cliente separado.

As condições de licenciamento para clientes são associadas ao servidor ao qual você está conectando os clientes.

As bibliotecas do cliente MQTT são implementações de referência da versão 3.1 do MQTT protocol.. É possível implementar seus próprios clientes em diferentes idiomas apropriados para diferentes plataformas de dispositivo. Consulte Formato e Protocolo do MQ Telemetry Transport.

A documentação da API não faz suposições sobre qual servidor MQTT o cliente está conectado. O comportamento do cliente poderá ser um pouco diferente quando conectado a diferentes servidores. As descrições que seguem descrevem o comportamento do cliente quando conectado ao serviço de telemetria do IBM WebSphere MQ.

Retornos de chamadas e sincronização em aplicativos do cliente MQTT

O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

Retornos de chamada

A interface `MqttCallback` possui três métodos de retorno de chamada; consulte uma implementação de exemplo em Callback.java.

`connectionLost(java.lang.Throwable cause)`

`connectionLost` será chamado quando um erro de comunicação levar ao descarte da conexão. Ele também será chamado, se o servidor descartar a conexão como resultado de um erro no servidor após a conexão ter sido estabelecida. Os erros do servidor são registrados no log de erros do gerenciador de filas. O servidor descarta a conexão com o cliente e o cliente chama o `MqttCallback.connectionLost`.

Os únicos erros remotos lançados como exceções no mesmo encadeamento que o aplicativo cliente são exceções a partir de `MqttClient.connect`. Os erros detectados pelo servidor após a conexão ser estabelecida serão relatados de volta ao método de retorno de chamada `MqttCallback.connectionLost` como `throwables`.

Os erros típicos do servidor que resultam em `connectionLost` são erros de autorização. Por exemplo, o servidor de telemetria tenta publicar em um tópico em nome de um cliente que não está autorizado a publicar no tópico. Qualquer coisa que resulte no retorno de um código de condição `MQCC_FAIL` ao servidor de telemetria pode resultar no descarte da conexão.

`deliveryComplete(MqttDeliveryToken token)`

`deliveryComplete` é chamado pelo cliente MQTT para transmitir um token de entrega de volta ao aplicativo cliente; consulte “Tokens de entrega” na página 538. Usando o token de entrega, o retorno de chamada pode acessar a mensagem publicada com o método `token.getMessage`.

Quando o retorno de chamada do aplicativo retornar o controle para o cliente MQTT após ser chamado pelo método `deliveryComplete`, a entrega será concluída. Até que a entrega seja concluída, mensagens com QoS 1 ou 2 serão retidas pela classe de persistência.

A chamada para `deliveryComplete` é um ponto de sincronização entre o aplicativo e a classe de persistência. O método `deliveryComplete` nunca é chamado duas vezes para a mesma mensagem.

Quando o retorno de chamada do aplicativo retorna de `deliveryComplete` para o cliente MQTT, o cliente chama `MqttClientPersistence.remove` para mensagens com QoS 1 ou 2. `MqttClientPersistence.remove` exclui a cópia armazenada localmente da mensagem publicada.

De uma perspectiva de processamento de transações, a chamada para `deliveryComplete` é uma transação de fase única que confirma a entrega. Se o processamento falhar durante o retorno de chamada, na reinicialização do cliente, `MqttClientPersistence.remove` será chamado novamente para excluir a cópia local da mensagem publicada. O retorno de chamada não é chamado novamente. Se você estiver usando o retorno de chamada para armazenar um log de mensagens entregues, não será possível sincronizar o log com o cliente MQTT. Se você deseja armazenar um log confiavelmente, então atualize o log na classe `MqttClientPersistence`.

O token de entrega e a mensagem são referenciados pelo encadeamento de aplicativos principal e o cliente MQTT. O cliente MQTT desreferenciará do objeto `MqttMessage` quando a entrega for concluída e o objeto do token de entrega quando o cliente for desconectado. O objeto `MqttMessage` poderá ser lixo coletado após a entrega ser concluída se o aplicativo cliente desreferenciá-lo. O token de entrega poderá ser lixo coletado após a sessão ser desconectada.

É possível obter os atributos `MqttDeliveryToken` e `MqttMessage` após uma mensagem ter sido publicada. Se você tentar configurar quaisquer atributos `MqttMessage` após a mensagem ter sido publicada, o resultado será indefinido.

O cliente MQTT continuará a processar confirmações de entrega, se o cliente se reconectar à sessão anterior com o mesmo `ClientIdentifier`; consulte "Sessões limpas" na página 535. O aplicativo cliente MQTT deve configurar `MqttClient.CleanSession` como `false` para a sessão anterior e configurá-lo como `false` na nova sessão. O cliente MQTT cria novos tokens de entrega e objetos de mensagem na nova sessão para as entregas pendentes. Ele recupera os objetos usando a classe `MqttClientPersistence`. Se o aplicativo cliente ainda tiver referências ao antigo tokens de entrega e mensagens, os desreferenciem. O retorno de chamada do aplicativo é chamado na nova sessão para quaisquer entregas iniciada na sessão anterior e concluídas nessa sessão.

O retorno de chamada do aplicativo será chamado depois que o aplicativo cliente se conectar, quando uma entrega pendente for concluída. Antes de o aplicativo cliente se conectar, ele poderá recuperar entregas pendentes usando o método `MqttClient.getPendingDeliveryTokens`.

Observe que o aplicativo cliente criou originalmente o objeto de mensagem publicado e sua matriz de bytes de carga útil. O cliente MQTT referencia esses objetos. O objeto de mensagem retornado pelo token de entrega no método `token.getMessage` não é necessariamente o mesmo objeto de mensagem criado pelo cliente. Se uma nova instância do cliente MQTT recriar o token de entrega, a classe `MqttClientPersistence` recriará o objeto `MqttMessage`. Para consistência, o `token.getMessage` retornará `null` se o `token.isCompleted` for `true`, independentemente se o objeto de mensagem foi criado pelo aplicativo cliente ou pela classe `MqttClientPersistence`.

`messageArrived(MqttTopic topic, MqttMessage message)`

`messageArrived` será chamado quando uma publicação chegar ao cliente que correspondeu a um tópico de assinatura. `topic` é o tópico da publicação, não o filtro de assinatura. Os dois poderão ser diferentes se o filtro contiver caracteres curingas.

Se o tópico corresponder a diversas assinaturas criadas pelo cliente, o cliente receberá diversas cópias da publicação. Se um cliente publicar em um tópico que também assina, ele receberá uma cópia da sua própria publicação.

Se uma mensagem for enviada com um QoS de 1 ou 2, a mensagem será armazenada pelo `MqttClientPersistence` de classe antes de o cliente MQTT chamar o `messageArrived`. `messageArrived` se comporta como `deliveryComplete`: é chamado apenas uma vez para uma publicação e a cópia local da publicação será removida por `MqttClientPersistence.remove` quando `messageArrived` retornar ao cliente MQTT. O cliente MQTT descartará suas referências para o tópico e a mensagem quando `messageArrived` retornar ao cliente MQTT. Os objetos de tópicos e mensagens serão o lixo coletado, se o aplicativo cliente não tiver retido em uma referência aos objetos.

Retornos de chamadas, encadeamento e sincronização do aplicativo cliente

O cliente MQTT chama um método de retorno de chamada em um encadeamento separado para o encadeamento de aplicativo principal. O aplicativo cliente não cria um encadeamento para o retorno de chamada, é criado pelo cliente MQTT.

O cliente MQTT sincroniza os métodos de retorno de chamada. Apenas uma instância do método de retorno de chamada é executada por vez. A sincronização torna mais fácil a atualização de um objeto que registra a contagem total que as publicações foram entregues. Uma instância do `MqttCallback.deliveryComplete` é executada por vez, e, portanto, é seguro para atualizar a contagem total sem sincronização adicional. Nesse caso, somente uma publicação chega por vez. Seu código no método `messageArrived` pode atualizar um objeto sem sincronizá-lo. Se você estiver se referindo à contagem total ou ao objeto que está sendo atualizado, em outro encadeamento, sincronize a contagem total ou o objeto.

O token de entrega fornece um mecanismo de sincronização entre o encadeamento principal do aplicativo e a entrega de uma publicação. O método `token.waitForCompletion` aguarda até que a entrega de uma publicação específica seja concluída ou até que um tempo limite opcional expire. Você pode usar `token.waitForCompletion` de algumas maneiras simples para processar uma publicação de cada vez:

1. Para pausar o aplicativo cliente até a entrega da publicação ser concluída; consulte Figura 88 na página 486.
2. Para sincronizar com o método `MqttCallback.deliveryComplete`. Somente quando o `MqttCallback.deliveryComplete` retornar para o cliente MQTT, `token.waitForCompletion` continuará. Usando esse mecanismo será possível sincronizar a execução de código em `MqttCallback.deliveryComplete` antes de o código ser executado no encadeamento de aplicativos principal.

E se você desejava publicar sem aguardar que cada publicação seja entregue, mas deseja confirmação quando todas as publicações foram entregues? Se você publicar em um único encadeamento, a última publicação a ser enviada será também a última a ser entregue.

Sincronização de solicitações enviadas ao servidor

Tabela 70 na página 534 descreve os métodos no cliente Java MQTT que enviam uma solicitação para o servidor. A menos que o aplicativo cliente configure um tempo limite indefinido, o cliente nunca aguardará indefinidamente pelo servidor. Se o cliente for interrompido, será um problema de programação de aplicativos ou um defeito no cliente MQTT.

Tabela 70. Comportamento de sincronização de métodos que resultam em solicitações para o servidor

Método	**Sincronização**	**Intervalo de tempo limite**
`MqttClient.Connect`	Aguarda para que uma conexão seja estabelecida com o servidor.	Padronizado como 30 segundos ou como configurado por um parâmetro, em seguida, lança uma exceção.
`MqttClient.Disconnect`	Aguarda o cliente MQTT concluir qualquer trabalho que deve ser realizado e a sessão TCP/IP para se desconectar.	
`MqttClient.Subscribe`	Espera pela conclusão do método Subscribe ou UnSubscribe.	
`MqttClient.UnSubscribe`		
`MqttClient.Publish`	Retorna imediatamente para o encadeamento de aplicativos após transmitir a solicitação ao cliente MQTT.	Nenhum.
`MqttDeliveryToken.waitForCompletion`	Aguarda o token de entrega a ser retornado.	Indefinido ou configurado como um parâmetro.

Conceitos relacionados

Sessões limpas

O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

Identificador de Cliente

Tokens de entrega

Publicação last will and testament

Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação

e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

Persistência de Mensagem em Clientes MQTT

Publicações
Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT pode criar publicações para enviar para o IBM WebSphere MQe assinar tópicos no IBM WebSphere MQ MQ para receber publicações

Qualidades de serviço fornecidas por um cliente MQTT
Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

Publicações Retidas e Clientes MQTT
Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

Assinaturas
Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

Sequências e filtros de tópicos em clientes MQTT
As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

Sessões limpas

O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

Ao conectar um aplicativo cliente MQTT usando o método `MqttClient.connect` , o cliente identifica a conexão usando o identificador de cliente e o endereço do servidor. O servidor verifica se informações da sessão foram salvas de uma conexão anterior no servidor. Se uma sessão anterior ainda existir e `cleanSession=true`, então, as informações da sessão anterior no cliente e no servidor serão limpas. Se `cleanSession=false`, a sessão anterior será continuada. Se não existir nenhuma sessão anterior, uma nova sessão será iniciada.

Nota: O Administrador do WebSphere MQ pode forçar o fechamento de uma sessão aberta e excluir todas as informações da sessão Se o cliente reabrir a sessão com `cleanSession=false`, uma nova sessão será iniciada.

Publicações

Se você usar o padrão `MqttConnectOptions` ou configurar `MqttConnectOptions.cleanSession` para `true` antes de se conectar ao cliente, todas as entregas de publicação pendentes para o cliente serão removidas quando o cliente se conectar.

A configuração de sessão limpa não tem efeito sobre as publicações enviadas com `QoS=0`. Para `QoS=1` e `QoS=2`, o uso de `cleanSession=true` pode resultar na perda de uma publicação.

Assinaturas

Se você usar `MqttConnectOptions` padrão ou configurar `MqttConnectOptions.cleanSession` como `true` antes de se conectar ao cliente, todas as assinaturas antigas para o cliente serão removidas quando o cliente se conectar. Quaisquer novas assinaturas feitas pelo cliente durante a sessão serão removidas quando ele se desconectar.

Se você configurar `MqttConnectOptions.cleanSession` como `false` antes de se conectar, todas as assinaturas criadas pelo cliente serão incluídas em todas as assinaturas que existiam para o cliente antes de ele se conectar. Todas as assinaturas permanecem ativas quando o cliente se desconecta.

Outra maneira de entender a maneira como o atributo `cleanSession` afeta assinaturas é pensar nele como um atributo modal. Em seu modo padrão, `cleanSession=true`, o cliente cria assinaturas e recebe publicações apenas dentro do escopo da sessão. No modo alternativo, `cleanSession=false`, assinaturas são duráveis. O cliente pode se conectar e se desconectar e suas assinaturas permanecem ativas. Ao se reconectar, o cliente recebe todas as publicações não entregues. Enquanto estiver conectado, ele pode modificar o conjunto de assinaturas que estão ativas em seu nome.

Você deve configurar o modo `cleanSession` antes de se conectar; o modo dura a sessão inteira. Para alterar essa configuração, você deve desconectar e reconectar o cliente. Se você alterar os modos do uso de `cleanSession=false` para `cleanSession=true`, todas as assinaturas anteriores para o cliente e todas as publicações que ainda não foram recebidas serão descartadas.

Conceitos relacionados

Retornos de chamadas e sincronização em aplicativos do cliente MQTT
O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

Identificador de Cliente

Tokens de entrega

Publicação last will and testament
Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

Persistência de Mensagem em Clientes MQTT

Publicações
Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT pode criar publicações para enviar para o IBM WebSphere MQe assinar tópicos no IBM WebSphere MQ MQ para receber publicações

Qualidades de serviço fornecidas por um cliente MQTT
Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

Publicações Retidas e Clientes MQTT
Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

Assinaturas
Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

Sequências e filtros de tópicos em clientes MQTT

As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

Identificador de Cliente

O identificador de cliente é uma sequência de 23 bytes que identifica um cliente MQTT . Cada identificador deve ser exclusivo para apenas um cliente conectado de cada vez. O identificador deve conter apenas caracteres válidos em um nome de gerenciador de filas. Nessas restrições, será possível usar qualquer sequência de identificação. É importante ter um procedimento para alocar identificadores de cliente e um meio de configurar um cliente com seu identificador escolhido.

O identificador do cliente é usado na administração de um sistema MQTT. Com centenas de milhares de clientes em potencial para administrar, é necessário estar apto para identificar rapidamente um cliente específico. Suponha por exemplo, que há um dispositivo e você é notificado, talvez por um cliente ligando para um help desk. Como o cliente identifica o dispositivo e como você correlaciona essa identificação com o servidor que normalmente conectado ao cliente? Você tem que consultar um banco de dados que mapeia cada dispositivo para um identificador do cliente e para um servidor? O nome do dispositivo identifica a qual servidor ele está conectado? Ao navegar pelas conexões do cliente MQTT, cada conexão será rotulada com o identificador do cliente. É necessário consultar uma tabela para mapear um identificador do cliente para um dispositivo físico?

O identificador do cliente identifica um determinado dispositivo, usuário ou aplicativo em execução no cliente? Se um cliente substituir um dispositivo com falha por um novo, o novo dispositivo terá o mesmo identificador que o dispositivo antigo? Você aloca um novo identificador? Se você mudar um dispositivo físico, mas mantiver o mesmo identificador, as publicações pendentes e assinaturas ativas serão automaticamente transferidas para o novo dispositivo.

Como você assegura que identificadores do cliente sejam exclusivos? Assim como um sistema para gerar identificadores exclusivos, deve-se ter um processo confiável para configurar o identificador no cliente. Talvez o dispositivo do cliente seja uma "caixa preta" sem interface com o usuário. Você fabrica o dispositivo com um identificador do cliente - como o uso de um endereço de Controle de Acesso à Mídia? Ou você tem um processo de instalação e configuração de software que configura o dispositivo antes de ele ser ativado?

Você pode criar um identificador do cliente a partir do endereço de Controle de Acesso à Mídia do dispositivo com 48 bits para manter o identificador curto e exclusivo. Se o tamanho da transmissão não for um problema crítico, você poderá usar os 17 bytes restantes para deixar o endereço mais fácil de administrar.

Conceitos relacionados

Retornos de chamadas e sincronização em aplicativos do cliente MQTT
O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

Sessões limpas
O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

Tokens de entrega

Publicação last will and testament
Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

Persistência de Mensagem em Clientes MQTT

Publicações
Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT pode criar publicações para enviar para o IBM WebSphere MQe assinar tópicos no IBM WebSphere MQ MQ para receber publicações

Qualidades de serviço fornecidas por um cliente MQTT
Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

Publicações Retidas e Clientes MQTT
Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

Assinaturas
Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

Sequências e filtros de tópicos em clientes MQTT
As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

Tokens de entrega

Quando um cliente publica em um tópico, um novo token de entrega é criado. Use o token de entrega para monitorar a entrega de uma publicação ou para bloquear o aplicativo cliente até que a entrega seja concluída.

O token é um objeto `MqttDeliveryToken`. Ele é criado chamando o método MqttTopic.publish() e é retido pelo cliente MQTT até que a sessão do cliente seja desconectada e a entrega seja concluída.

O uso normal do token é para verificar se a entrega foi concluída. Bloqueie o aplicativo cliente até que a entrega seja concluída usando o token retornado para chamar `token.waitForCompletion`. Como alternativa, forneça um identificador `MqttCallBack`. Quando o cliente MQTT tiver recebido todas as confirmações esperadas como parte da entrega da publicação, ele chamará `MqttCallBack.deliveryComplete` transmitindo o token de entrega como um parâmetro.

Até a entrega ser concluída, será possível inspecionar a publicação usando o token de entrega retornado chamando `token.getMessage`.

Entregas concluídas
A conclusão de entregas é assíncrona e depende da qualidade de serviço associada à publicação.

No máximo uma vez

`QoS=0`
A entrega é concluída imediatamente no retorno de `MqttTopic.publish`. `MqttCallback.deliveryComplete` é chamado imediatamente.

Pelo menos uma vez

`QoS=1`
A entrega será concluída quando uma confirmação da publicação tiver sido recebida do gerenciador de filas. `MqttCallback.deliveryComplete` será chamado quando a confirmação for recebida. A mensagem poderá ser entregue mais de uma vez antes de `MqttCallback.deliveryComplete` ser chamado, se as comunicações forem lentas ou não confiáveis.

Exatamente uma vez

QoS=2
A entrega será concluída quando o cliente receber uma mensagem de conclusão de que a publicação foi publicada para os assinantes. `MqttCallback.deliveryComplete` será chamado assim que a mensagem de publicação for recebida. Ele não espera a mensagem de conclusão.

Em raras circunstâncias, o aplicativo cliente pode não retornar normalmente para o cliente MQTT a partir do `MqttCallback.deliveryComplete`. Você sabe que a entrega foi concluída porque o `MqttCallback.deliveryComplete` foi chamado. Se o cliente reiniciar a mesma sessão, `MqttCallback.deliveryComplete` não será chamado novamente.

Entregas incompletas

Se a entrega não for concluída após a sessão do cliente ser desconectada, será possível conectar o cliente novamente e concluir a entrega. Só será possível concluir a entrega de uma mensagem, se a mensagem foi publicada em uma sessão com o atributo `MqttConnectionOptions` configurado como `false`.

Crie o cliente usando o mesmo identificador de cliente e endereço do servidor e, em seguida, se conecte configurando o atributo `cleanSession` `MqttConnectionOptions` como `false` novamente. Se você configurar `cleanSession` como `true`, os tokens de entrega pendentes serão descartados.

Será possível verificar se há alguma entrega pendente chamando `MqttClient.getPendingDeliveryTokens`. Será possível chamar `MqttClient.getPendingDeliveryTokens` antes de se conectar ao cliente.

Conceitos relacionados

Retornos de chamadas e sincronização em aplicativos do cliente MQTT
O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

Sessões limpas
O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

Identificador de Cliente

Publicação last will and testament
Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

Persistência de Mensagem em Clientes MQTT

Publicações
Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT pode criar publicações para enviar para o IBM WebSphere MQe assinar tópicos no IBM WebSphere MQ MQ para receber publicações

Qualidades de serviço fornecidas por um cliente MQTT
Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

Publicações Retidas e Clientes MQTT

Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

Assinaturas
Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

Sequências e filtros de tópicos em clientes MQTT
As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

Publicação last will and testament

Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

Crie um tópico para a publicação last will and testament. Você pode criar um tópico, como `MQTTManagement/Connections/`*`server URI`*`/`*`client identifer`*`/Lost`, por exemplo,

Configure um "último testamento" usando o método `MqttConnectionOptions.setWill(MqttTopic lastWillTopic, byte [] lastWillPayload, int lastWillQos, boolean lastWillRetained)` .

Considere criar um registro de data e hora na mensagem `lastWillPayload`. Inclua outras informações do cliente que auxiliem na identificação do cliente e nas circunstâncias da conexão. Transmita o objeto `MqttConnectionOptions` para o construtor `MqttClient`.

Configure `lastWillQos` como 1 ou 2para tornar a mensagem persistente no WebSphere MQe garantir a entrega. Para reter as informações da última conexão perdida, configure `lastWillRetained` como `true`.

A publicação "last will and testament" será enviada para os assinantes, se a conexão terminar inesperadamente. Ela será enviada se a conexão terminar sem que o cliente chame o método `MqttClient.disconnect`.

Para monitorar conexões, complemente a publicação "last will and testament" com outras publicações para registrar conexões e desconexões programadas.

Conceitos relacionados

Retornos de chamadas e sincronização em aplicativos do cliente MQTT
O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

Sessões limpas
O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

Identificador de Cliente

Tokens de entrega

Persistência de Mensagem em Clientes MQTT

Publicações

Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT pode criar publicações para enviar para o IBM WebSphere MQe assinar tópicos no IBM WebSphere MQ MQ para receber publicações

Qualidades de serviço fornecidas por um cliente MQTT
Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

Publicações Retidas e Clientes MQTT
Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

Assinaturas
Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

Sequências e filtros de tópicos em clientes MQTT
As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

Persistência de Mensagem em Clientes MQTT

As mensagens de publicação serão transformadas em persistentes se forem enviadas com uma qualidade de serviço de "pelo menos uma vez" ou "exatamente uma vez". É possível implementar seu próprio mecanismo de persistência no cliente ou usar o mecanismo de persistência padrão fornecido com o cliente. A persistência funciona em ambas as direções, para publicações enviadas para o cliente ou por meio dele.

Na persistência de mensagem do MQTT há dois aspectos; como a mensagem será transferida e se ela está enfileirada no IBM MessageSight e IBM WebSphere MQ como uma mensagem persistente.

1. O cliente MQTT acopla a persistência da mensagem com a qualidade de serviço. Dependendo da qualidade de serviço escolhida para uma mensagem, a mensagem se torna persistente. A persistência de mensagem é necessária para implementar a qualidade de serviço necessária.

   Se você especificar "no máximo uma vez", `QoS=0`, o cliente descartará a mensagem assim que for publicada. Se houver alguma falha no processamento de envio de dados da mensagem, ela não será enviada novamente. Mesmo se o cliente permanecer ativo, a mensagem não será enviada novamente. O comportamento das mensagens `QoS=0` é o mesmo que as mensagens não persistentes rápidas do IBM WebSphere MQ.

   Se uma mensagem for publicada por um cliente com `QoS` de 1 ou 2, ela se tornará persistente. A mensagem é armazenada localmente e descartada apenas do cliente quando não é mais necessária para garantir "pelo menos uma vez", `QoS=1`ou "exatamente uma vez", `QoS=2`, entrega.
2. Se uma mensagem estiver marcada como `QoS` 1 ou 2, ela será enfileirada no IBM MessageSight e IBM WebSphere MQ como uma mensagem persistente. Se estiver marcada como `QoS=0`, ela será enfileirada no IBM MessageSight e IBM WebSphere MQ como uma mensagem não persistente. No IBM WebSphere MQ , mensagens não persistentes são transferidas entre os gerenciadores de filas "exatamente uma vez", a menos que o canal de mensagens tenha o atributo `NPMSPEED` configurado como FAST.

Uma publicação persistente é armazenada no cliente até ser recebida por um aplicativo cliente. Para `QoS=2`, a publicação será descartada a partir do cliente quando o retorno de chamada do aplicativo retornar ao controle. Para `QoS=1`, o aplicativo poderá receber a publicação novamente, se ocorrer uma falha. Para `QoS=0`, o retorno de chamada não recebe a publicação mais de uma vez. Ele poderá não receber a publicação se houver uma falha ou se o cliente estiver desconectado no momento da publicação.

Quando você se subscrever para um tópico, será possível reduzir a QoS com a qual o assinante recebe as mensagens para corresponder a suas capacidades de persistência. As publicações criadas em uma QoS `maior` superior são enviadas com a QoS mais alta que o assinante solicitou.

Armazenando de mensagens

A implementação do armazenamento de dados em pequenos dispositivos varia bastante. O modelo de mensagens persistentes de salvamento temporariamente em armazenamento gerenciado pelo cliente MQTT, pode ser muito lento ou demandar muito armazenamento. Em dispositivos móveis, o sistema operacional móvel poderá fornecer um serviço de armazenamento ideal para mensagens do MQTT.

Para fornecer flexibilidade para atender às restrições de pequenos dispositivos, o cliente MQTT tem duas interfaces de persistência. As interfaces definem as operações envolvidas no armazenamento de mensagens persistentes. As interfaces são descritas na documentação da API para o Cliente de MQTT para Java. Para obter links para a documentação da API do cliente para as bibliotecas do cliente MQTT, consulte Referência de programação do cliente MQTT.. É possível implementar as interfaces para se adequar a um dispositivo. O cliente MQTT executado no Java SE tem uma implementação padrão das interfaces que armazenam mensagens persistentes no sistema de arquivos. Ele usa o pacote `java.io`. O cliente também possui uma implementação padrão para o Java ME, `MqttDefaultMIDPPersistence`

Classes de persistência

`MqttClientPersistence`
: Transmite uma instância de sua implementação de `MqttClientPersistence` para o cliente MQTT como um parâmetro do construtor `MqttClient`. Se você omitir o parâmetro `MqttClientPersistence` do construtor `MqttClient`, o cliente MQTT armazenará as mensagens persistentes usando a classe `MqttDefaultFilePersistence` ou `MqttDefaultMIDPPersistence`.

`MqttPersistable`
: `MqttClientPersistence` obtém e coloca objetos `MqttPersistable` usando uma chave de armazenamento. Deve-se fornecer uma implementação de `MqttPersistable`, bem como a implementação de `MqttClientPersistence`, se não estiver usando o `MqttDefaultFilePersistence` ou `MqttDefaultMIDPPersistence`.

`MqttDefaultFilePersistence`

: O cliente MQTT fornece a classe `MqttDefaultFilePersistence`. Se você instanciar `MqttDefaultFilePersistence` em seu aplicativo cliente, será possível fornecer o diretório para armazenar mensagens persistentes como um parâmetro do construtor `MqttDefaultFilePersistence`.

: Como alternativa, o cliente MQTT pode instanciar `MqttDefaultFilePersistence` e colocar arquivos em um diretório padrão. O nome do diretório é *`client identifier`*`-tcp` *`hostname portnumber`* `"\", "\\", "/", ":"` e `" "` são removidos da sequência do nome do diretório

: O caminho para o diretório é o valor da propriedade de sistema `rcp.data`. Se `rcp.data` não estiver configurado, o caminho será o valor da propriedade de sistema `usr.data`.

: `rcp.data` é uma propriedade associada à instalação de uma Eclipse Rich Client Platform (RCP) ou um OSGi.

: `usr.data` é o diretório no qual o comando Java que iniciou o aplicativo foi ativado..

`MqttDefaultMIDPPersistence`
: `MqttDefaultMIDPPersistence` tem um construtor padrão e nenhum parâmetro. Ele usa o pacote `javax.microedition.rms.RecordStore` para armazenar mensagens.

Conceitos relacionados

Retornos de chamadas e sincronização em aplicativos do cliente MQTT
O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão

de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

Sessões limpas
O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

Identificador de Cliente

Tokens de entrega

Publicação last will and testament
Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

Publicações
Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT pode criar publicações para enviar para o IBM WebSphere MQe assinar tópicos no IBM WebSphere MQ MQ para receber publicações

Qualidades de serviço fornecidas por um cliente MQTT
Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

Publicações Retidas e Clientes MQTT
Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

Assinaturas
Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

Sequências e filtros de tópicos em clientes MQTT
As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

Publicações

Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT pode criar publicações para enviar para o IBM WebSphere MQe assinar tópicos no IBM WebSphere MQ MQ para receber publicações

Um `MqttMessage` tem uma matriz de bytes como sua carga útil. Tente manter as mensagens o menor possível. O comprimento máximo de mensagem permitido pelo protocolo do MQTT é 250 MB.

Geralmente, um programa cliente do MQTT usa `java.lang.String` ou `java.lang.StringBuffer` para manipular o conteúdo da mensagem. Por conveniência, a classe `MqttMessage` tem um método `toString` para converter sua carga útil para uma sequência. Para criar a carga útil da matriz de bytes a partir de um `java.lang.String` ou `java.lang.StringBuffer`, use o método `getBytes`.

O método `getBytes` converte uma sequência para o conjunto de caracteres padrão para a plataforma. O conjunto de caracteres padrão geralmente é `UTF-8`. As publicações do MQTT que contêm apenas texto

são normalmente codificadas em UTF-8. Use o método `getBytes("UTF8")` para substituir o conjunto de caracteres padrão.

No IBM WebSphere MQ, uma publicação do MQTT é recebida como uma mensagem `jms-bytes`. A mensagem inclui uma pasta MQRFH2 que contém um <mqtt> e uma pasta <mqps>. A pasta <mqtt> contém o `clientId` e o qos, mas esse conteúdo poderá mudar no futuro.

Um `MqttMessage` tem três atributos adicionais: qualidade de serviço, se está retida e se é uma duplicata. O sinalizador duplicado será configurado somente se a qualidade de serviço for "pelo menos uma vez" ou "exatamente uma vez". Se a mensagem foi enviada anteriormente e não for confirmada rápido suficiente pelo cliente MQTT, a mensagem será enviada novamente, com o atributo duplicado configurado como `true`.

Publicando

Para criar uma publicação em um aplicativo cliente MQTT, crie um `MqttMessage`. Configure sua carga útil, qualidade de serviço e se ela está retida, e chame o método `MqttTopic.publish(MqttMessage message)`; `MqttDeliveryToken` é retornado e a conclusão da publicação é assíncrona.

Como alternativa, o cliente MQTT pode criar um objeto de mensagem temporário para você a partir dos parâmetros no método `MqttTopic.publish(byte [] payload, int qos, boolean retained)` ao criar uma publicação.

Se a publicação tiver uma qualidade de serviço "pelo menos uma vez" ou "exatamente uma vez", QoS=1 ou QoS=2, o cliente MQTT chamará a interface `MqttClientPersistence`. Ele chamará `MqttClientPersistence` para armazenar a mensagem antes de retornar um token de entrega para o aplicativo.

O aplicativo pode escolher bloquear até a mensagem ser entregue ao servidor usando o método `MqttDeliveryToken.waitForCompletion`. Como alternativa, o aplicativo pode continuar sem bloqueio. Se você deseja verificar se as publicações foram entregues, sem bloqueio, registre uma instância de uma classe de retorno de chamada que implementa `MqttCallback` com o cliente MQTT. O cliente MQTT chama o método `MqttCallback.deliveryComplete` assim que a publicação foi entregue. Dependendo da qualidade de serviço, a entrega pode ser quase imediatamente para QoS=0 ou ela pode levar algum tempo para QoS=2.

Use o método `MqttDeliveryToken.isComplete` para pesquisar se a entrega está concluída. Enquanto o valor de `MqttDeliveryToken.isComplete` for `false`, será possível chamar `MqttDeliveryToken.getMessage` para obter o conteúdo da mensagem. Se o resultado da chamada `MqttDeliveryToken.isComplete` for `true`, a mensagem foi descartada e a chamada de `MqttDeliveryToken.getMessage` lançaria uma exceção nula de ponteiro. Não há nenhuma sincronização integrada entre o `MqttDeliveryToken.getMessage` e `MqttDeliveryToken.isComplete`.

Se o cliente se desconectar antes de receber todos os tokens de entrega pendentes, uma nova instância do cliente poderá consultar os tokens de entrega pendentes antes de se conectar. Até que o cliente se conecte, nenhuma nova entrega será concluída e segura para chamar `MqttDeliveryToken.getMessage`. Use o método `MqttDeliveryToken.getMessage` para descobrir quais publicações não foram entregues. Os tokens de entrega pendentes serão descartados se você se conectar com `MqttConnectOptions.cleanSession` configurado em seu valor padrão, `true`.

Assinando

Um gerenciador de filas ou IBM MessageSight é responsável por criar publicações para enviar para um assinante do MQTT. O gerenciador de filas verificará se o filtro de tópicos em uma assinatura criada por um cliente MQTT corresponderá à sequência de tópicos em uma publicação. A correspondência pode ser uma correspondência exata ou a correspondência pode incluir caracteres curingas. Antes de a publicação ser encaminhada para o assinante pelo gerenciador de filas, o gerenciador de filas verificará os atributos do tópico associada à publicação. Ele segue o procedimento de procura descrito em Assinatura usando uma sequência de tópicos que contém caracteres curingas para identificar se um objeto de tópico administrativo concederá a autoridade do usuário para assinatura.

Quando o cliente MQTT receber uma publicação com a qualidade de serviço "pelo menos uma vez", ele chamará o método `MqttCallback.messageArrived` para processar a publicação. Se a qualidade de serviço da publicação for "exatamente uma vez", `QoS=2`, o cliente MQTT chamará a interface `MqttClientPersistence` para armazenar a mensagem quando ela for recebida. Ela então chama `MqttCallback.messageArrived`.

Conceitos relacionados

Retornos de chamadas e sincronização em aplicativos do cliente MQTT
O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

Sessões limpas
O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

Identificador de Cliente

Tokens de entrega

Publicação last will and testament
Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

Persistência de Mensagem em Clientes MQTT

Qualidades de serviço fornecidas por um cliente MQTT
Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

Publicações Retidas e Clientes MQTT
Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

Assinaturas
Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

Sequências e filtros de tópicos em clientes MQTT
As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

Qualidades de serviço fornecidas por um cliente MQTT

Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

A qualidade de serviço de uma publicação é um atributo de `MqttMessage`. Ela é configurada pelo método `MqttMessage.setQos`.

O método `MqttClient.subscribe` pode reduzir a qualidade de serviço aplicada às publicações enviadas para um cliente em um tópico. A qualidade de serviço de uma publicação encaminhada para um assinante pode ser diferente da qualidade de serviço da publicação. O menor dos dois valores é usado para encaminhar uma publicação.

No máximo uma vez
`QoS=0`

A mensagem é entregue no máximo uma vez ou não é entregue de modo algum. Sua entrega na rede não é reconhecida.

A mensagem não é armazenada. A mensagem poderá ser perdida se o cliente for desconectado ou se o servidor falhar.

`QoS=0` é o modo de transferência mais rápido. Ele é, às vezes, chamado de "fire and forget".

O protocolo MQTT não requer que servidores encaminhem publicações com `QoS=0` para um cliente. Se o cliente estiver desconectado no momento em que o servidor receber a publicação, a publicação poderá ser descartada, dependendo do servidor. O serviço de telemetria (MQXR) não descarta mensagens enviadas com `QoS=0`. Elas são armazenadas como mensagens não persistentes e só serão descartadas, se o gerenciador de filas for interrompido.

Pelo menos uma vez
`QoS=1`

`QoS=1` é o modo de transferência padrão.

A mensagem é sempre entregue pelo menos uma vez. Se o emissor não receber uma confirmação, a mensagem será enviada novamente com o sinalizador DUP configurado até que uma confirmação seja recebida. Como resultado, o receptor pode receber a mesma mensagem diversas vezes e processá-la diversas vezes.

A mensagem deve ser armazenada localmente no emissor e no receptor até ser processada.

A mensagem será excluída do receptor após ter processado a mensagem. Se o receptor for um broker, a mensagem será publicada para seus assinantes. Se o receptor for um cliente, a mensagem será entregue para o aplicativo de assinante. Após a mensagem ser excluída, o receptor enviará uma confirmação para o emissor.

A mensagem será excluída do emissor após ter recebido uma confirmação do receptor.

Exatamente uma vez
`QoS=2`

A mensagem é sempre entregue exatamente uma vez.

A mensagem deve ser armazenada localmente no emissor e no receptor até ser processada.

`QoS=2` é o mais seguro e o modo de transferência mais lento. Ele levará pelo menos dois pares de transmissões entre o emissor e o receptor antes de a mensagem ser excluída do emissor. A mensagem poderá ser processada no receptor após a primeira transmissão.

No primeiro par de transmissões, o emissor transmite a mensagem e recebe a confirmação do receptor de que ele armazenou a mensagem. Se o emissor não receber uma confirmação, a mensagem será enviada novamente com o sinalizador DUP configurado até que uma confirmação seja recebida.

No segundo par de transmissões, o emissor informa ao receptor que ele pode concluir o processamento da mensagem, "PUBREL" Se o emissor não receber uma confirmação da mensagem "PUBREL" , a mensagem "PUBREL" será enviada novamente até que uma confirmação seja recebida. O emissor exclui a mensagem salva quando recebe o reconhecimento para a mensagem "PUBREL"

O receptor poderá processar a mensagem na primeira ou na segunda fase, contanto que não reprocesse a mensagem. Se o receptor for um broker, ele publicará a mensagem para os assinantes. Se o receptor for um cliente, ele entregará a mensagem para o aplicativo de assinante. O receptor envia uma mensagem de conclusão de volta para o emissor que concluiu o processamento da mensagem.

Conceitos relacionados

Retornos de chamadas e sincronização em aplicativos do cliente MQTT

O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

Sessões limpas
O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

Identificador de Cliente

Tokens de entrega

Publicação last will and testament
Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

Persistência de Mensagem em Clientes MQTT

Publicações
Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT pode criar publicações para enviar para o IBM WebSphere MQe assinar tópicos no IBM WebSphere MQ MQ para receber publicações

Publicações Retidas e Clientes MQTT
Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

Assinaturas
Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

Sequências e filtros de tópicos em clientes MQTT
As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

Publicações Retidas e Clientes MQTT

Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

Use o método `MqttMessage.setRetained` para especificar se uma publicação em um tópico está ou não retida.

Para excluir uma publicação retida no IBM WebSphere MQ, execute o comando do MQSC **CLEAR TOPICSTR**

Se você criar uma publicação com uma carga útil nula, a publicação vazia será encaminhada aos assinantes. Outros brokers do MQTT podem não encaminhar uma publicação vazia aos assinantes.

Se você publicar uma publicação não retida para um tópico que tenha uma publicação retida, a publicação retida não será afetada. Os assinantes atuais recebem a nova publicação. Novos assinantes recebem a publicação retida por primeiro, em seguida, recebem quaisquer novas publicações.

Ao criar ou atualizar uma publicação retida, envie a publicação com um QoS ou 1 ou 2. Se você enviar com um QoS de 0, o IBM WebSphere MQ cria uma publicação retida não persistente. A publicação não será retida se o gerenciador de filas for interrompido.

Use publicações retidas para registrar o valor mais recente de uma medida. Novos assinantes para o tópico retido recebem imediatamente o valor mais recente da medida. Se nenhuma nova medida for adquirida desde que o assinante se inscreveu por último no tópico de publicação e se o assinante se inscrever novamente, o assinante receberá a publicação retida mais recente no tópico novamente.

Conceitos relacionados

Retornos de chamadas e sincronização em aplicativos do cliente MQTT
O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

Sessões limpas
O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

Identificador de Cliente

Tokens de entrega

Publicação last will and testament
Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

Persistência de Mensagem em Clientes MQTT

Publicações
Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT pode criar publicações para enviar para o IBM WebSphere MQe assinar tópicos no IBM WebSphere MQ MQ para receber publicações

Qualidades de serviço fornecidas por um cliente MQTT
Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

Assinaturas
Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

Sequências e filtros de tópicos em clientes MQTT
As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

Assinaturas

Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são

enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

Crie assinaturas usando os métodos `MqttClient.subscribe`, passando um ou mais filtros de tópicos e parâmetros de qualidade de serviço. O parâmetro de qualidade de serviço configura a qualidade de serviço máxima que o assinante está preparado para usar para receber uma mensagem. Mensagens enviadas para esse cliente não podem ser entregues com qualidade de serviço superior. A qualidade de serviço é configurada para o valor original mais baixo quando a mensagem foi publicada e para o nível especificado para a assinatura. A qualidade de serviço padrão para o recebimento de mensagens é `QoS=1`, pelo menos uma vez.

A solicitação de assinatura em si é enviada com `QoS=1`.

Publicações são recebidas por um assinante quando o cliente MQTT chama o método `MqttCallback.messageArrived`. O método `messageArrived` também passa a sequência de tópicos com a qual a mensagem foi publicada para o assinante.

É possível remover uma assinatura ou um conjunto ou assinaturas usando os métodos `MqttClient.unsubscribe`.

Um comando do WebSphere MQ pode remover uma assinatura. Listar assinaturas usando o WebSphere MQ Explorer ou usando comandos do **runmqsc** ou PCF Todas as assinaturas do cliente MQTT são nomeadas. Eles recebem um nome do formato: *`ClientIdentifier:Topic name`*

Se você usar `MqttConnectOptions` padrão ou configurar `MqttConnectOptions.cleanSession` como `true` antes de se conectar ao cliente, todas as assinaturas antigas para o cliente serão removidas quando o cliente se conectar. Quaisquer novas assinaturas feitas pelo cliente durante a sessão serão removidas quando ele se desconectar.

Se você configurar `MqttConnectOptions.cleanSession` como `false` antes de se conectar, todas as assinaturas criadas pelo cliente serão incluídas em todas as assinaturas que existiam para o cliente antes de ele se conectar. Todas as assinaturas permanecem ativas quando o cliente se desconecta.

Outra maneira de entender a maneira como o atributo `cleanSession` afeta assinaturas é pensar nele como um atributo modal. Em seu modo padrão, `cleanSession=true`, o cliente cria assinaturas e recebe publicações apenas dentro do escopo da sessão. No modo alternativo, `cleanSession=false`, assinaturas são duráveis. O cliente pode se conectar e se desconectar e suas assinaturas permanecem ativas. Ao se reconectar, o cliente recebe todas as publicações não entregues. Enquanto estiver conectado, ele pode modificar o conjunto de assinaturas que estão ativas em seu nome.

Você deve configurar o modo `cleanSession` antes de se conectar; o modo dura a sessão inteira. Para alterar essa configuração, você deve desconectar e reconectar o cliente. Se você alterar os modos do uso de `cleanSession=false` para `cleanSession=true`, todas as assinaturas anteriores para o cliente e todas as publicações que ainda não foram recebidas serão descartadas.

Publicações que correspondem a assinaturas ativas são enviadas para o cliente assim que são publicadas. Se o cliente estiver desconectado, elas serão enviadas para o cliente se reconectar ao mesmo servidor com o mesmo identificador de cliente e se `MqttConnectOptions.cleanSession` estiver configurado para `false`.

Assinaturas para um determinado cliente são identificadas pelo identificador de cliente. É possível reconectar o cliente de um dispositivo de cliente diferente ao mesmo servidor, continuar com as mesmas assinaturas e receber publicações não entregues.

Conceitos relacionados

Retornos de chamadas e sincronização em aplicativos do cliente MQTT
O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

Sessões limpas
O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente

uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

Identificador de Cliente

Tokens de entrega

Publicação last will and testament
Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

Persistência de Mensagem em Clientes MQTT

Publicações
Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT pode criar publicações para enviar para o IBM WebSphere MQe assinar tópicos no IBM WebSphere MQ MQ para receber publicações

Qualidades de serviço fornecidas por um cliente MQTT
Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

Publicações Retidas e Clientes MQTT
Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

Sequências e filtros de tópicos em clientes MQTT
As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

Sequências e filtros de tópicos em clientes MQTT

As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

As sequências de tópicos são usadas para enviar publicações para os assinantes. Crie uma sequência de tópicos usando o método, `MqttClient.getTopic(java.lang.String topicString)`

Os filtros de tópicos são usados para assinar tópicos e receber publicações. Os filtros de tópicos podem conter caracteres curingas. Com caracteres curinga, é possível assinar vários tópicos. Crie um filtro de tópico usando um método de subscrição; por exemplo, `MqttClient.subscribe(java.lang.String topicFilter)`

Sequências de tópicos

A sintaxe de uma sequência de tópicos IBM WebSphere MQ é descrita em Sequências de tópicos. A sintaxe das sequências de tópicos do MQTT é descrita na classe `MqttClient` na documentação da API para o Cliente de MQTT para Java.Para obter links para a documentação da API do cliente para as bibliotecas do cliente MQTT, consulte Referência de programação do cliente MQTT..

A sintaxe de cada tipo de sequência de tópicos é quase idêntica. Há quatro pequenas diferenças:

1. As sequências de tópicos enviadas para IBM WebSphere MQ por clientes MQTT devem seguir a convenção para nomes do gerenciador de filas. Em particular, as sequências de tópicos não contêm hifens.

2. Os comprimentos máximos são diferentes. As sequências de tópicos do IBM WebSphere MQ são limitadas a 10.240 caracteres. Um cliente MQTT pode criar sequências de tópicos de até 65.535 bytes.
3. Uma sequência de tópicos criada por um cliente MQTT não pode conter um caractere nulo.
4. No WebSphere Message Broker, um nível de tópico nulo, `'...//...'` era inválido. Os níveis de tópicos nulos são suportados pelo IBM WebSphere MQ.

Diferente da publicação/assinatura do IBM WebSphere MQ, o protocolo `mqttv3` não terá um conceito de um objeto do tópico administrativo. Não é possível construir uma sequência de tópicos a partir de um objeto do tópico e uma sequência de tópicos. No entanto, uma sequência de tópicos é mapeada para um tópico administrativo no IBM WebSphere MQ. O controle de acesso associado ao tópico administrativo determina se uma publicação é publicada para o tópico ou descartada. Os atributos aplicados a uma publicação quando for redirecionada para os assinantes serão influenciados pelos atributos do tópico administrativo.

Filtros de tópico

A sintaxe de um filtro de tópico IBM WebSphere MQ é descrita em esquema curinga baseado em tópico. A sintaxe dos filtros de tópicos que você pode construir com um cliente MQTT são descritas na classe `MqttClient` na documentação da API para o Cliente de MQTT para Java. Para obter links para a documentação da API do cliente para as bibliotecas do cliente MQTT, consulte Referência de programação do cliente MQTT..

A sintaxe de cada tipo de filtro de tópico é quase idêntica. A única diferença está na maneira como diferentes brokers do MQTT interpretam um filtro de tópico No WebSphere Message Broker V6, um curinga de vários níveis só poderia ser usado no final de um filtro de tópico Em IBM WebSphere MQ, um curinga de vários níveis pode ser usado em qualquer nível na árvore de tópicos; por exemplo, USA/#/ `Dutchess County`

Conceitos relacionados

Retornos de chamadas e sincronização em aplicativos do cliente MQTT
O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

Sessões limpas
O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

Identificador de Cliente

Tokens de entrega

Publicação last will and testament
Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

Persistência de Mensagem em Clientes MQTT

Publicações
Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT pode criar publicações para enviar para o IBM WebSphere MQe assinar tópicos no IBM WebSphere MQ MQ para receber publicações

Qualidades de serviço fornecidas por um cliente MQTT

Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

Publicações Retidas e Clientes MQTT
Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

Assinaturas
Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

Conceitos de programação do cliente C

As diferenças entre o cliente C e Java para a versão 3.1 do MQ Telemetry Transport são descritas neste tópico. O tópico complementa os conceitos do cliente e as informações de referência C.

O tópico está organizado da mesma maneira que “Conceitos de programação do cliente MQTT” na página 531. Cada título corresponde a um tópico em *WebSphere(r) MQ conceitos de programação do cliente de Transporte de Telemetria*. As seções descrevem as diferenças entre o cliente C e o cliente Java Não são descritas pequenas diferenças nas assinaturas entre os métodos Java e as funções C.

O cliente C é mais frequentemente usado para implementar um adaptador leve entre um dispositivo de telemetria e o daemon WebSphere MQ Telemetry para dispositivos. O daemon é normalmente usado como um concentrador de rede entre dispositivos de telemetria muito leves e o serviço de telemetria (MQXR).

O daemon do WebSphere MQ Telemetry para dispositivos também é um cliente C e diferenças em seu comportamento do serviço de telemetria (MQXR) são descritas. O daemon não fornece uma implementação do JAAS ou SSL para clientes que se conectam a ele.

`mqttclient.dll` e `mqttclient.lib` são as bibliotecas do Windows de 32 bits que contêm funções do cliente para a implementação C do protocolo MQ Telemetry Transport versão 3.1 .. As bibliotecas do Linux de 32 bits são `libmqttclient.so` e `libmqttclient.a`. Dois arquivos de cabeçalho contêm a função e outras declarações necessárias pelos aplicativos clientes: `MQTTClient.h` e `MQTTClientPersistence.h`. Esses arquivos são fornecidos pela instalação do WebSphere MQ Telemetry.

Para desenvolver e executar um cliente de Transporte de Telemetria MQ , é necessário copiar esses arquivos no dispositivo do cliente. Diferentemente dos clientes do WebSphere MQ , não é necessário instalar um tempo de execução do cliente separado

Consulte as condições de licenciamento associadas ao recurso WebSphere MQ Telemetry que controlam a conexão de clientes do MQ Telemetry Transport para WebSphere MQ e o daemon do WebSphere MQ Telemetry para dispositivos.

O cliente C é uma implementação de referência da versão 3.1 do MQ Telemetry Transport. É possível implementar seus próprios clientes em diferentes idiomas apropriados para diferentes plataformas de dispositivo. Consulte Formato e protocolo do MQ Telemetry Transport para obter os detalhes.

O identificador do cliente MQTT

“Identificador de Cliente” na página 537	O identificador de cliente é uma sequência de 23 bytes que identifica um cliente MQTT . Cada identificador deve ser exclusivo para apenas um cliente conectado de cada vez. O identificador deve conter apenas caracteres válidos em um nome de gerenciador de filas. Nessas restrições, será possível usar qualquer sequência de identificação. É importante ter um procedimento para alocar identificadores de cliente e um meio de configurar um cliente com seu identificador escolhido.

- Não há diferenças.

Publicações

“Publicações” na página 543	Publicações são instâncias de `MqttMessage` associadas a uma sequência de tópicos. O cliente do MQTT

- A função de retorno de chamada não é chamada para publicações com qualidade de serviço "disparar e esquecer", `QoS=0`.

Tokens de entrega

“Tokens de entrega” na página 538	Quando um cliente publica em um tópico, um novo token de entrega é criado. Use o token de entrega para monitorar a entrega de uma publicação ou para bloquear o aplicativo cliente até que a entrega seja concluída.

- Um token de entrega é um `int`. Ele tem um typedef de `MQTTClient_deliveryToken`
- A função de retorno de chamada não é chamada para publicações com qualidade de serviço "disparar e esquecer", `QoS=0`.

Publicações Retidas

“Publicações Retidas e Clientes MQTT” na página 547	Se você criar uma assinatura para um tópico que tenha uma publicação retida, a publicação retida mais recente no tópico será encaminhada imediatamente para você.

- As mensagens retidas são salvas apenas no daemon se a persistência estiver configurada. Consulte Salvando mensagens e assinaturas retidas

  Para WebSphere MQ, a qualidade de serviço afeta se uma mensagem retida é salva permanentemente. Se um cliente estiver conectado ao serviço de telemetria, mensagens retidas com a qualidade de serviço “disparar e esquecer", `QoS=0` são descartadas, se o gerenciador de filas for encerrado.

Assinaturas

“Assinaturas” na página 548	Crie assinaturas para registrar um interesse em tópicos de publicação usando um filtro de tópico. Um cliente pode criar diversas assinaturas ou uma assinatura contendo um filtro de tópicos que use curingas para registrar um interesse em diversos tópicos. Publicações em tópicos correspondentes aos filtros são enviadas para o cliente. Assinaturas podem permanecer ativas enquanto um cliente está desconectado. As publicações são enviadas para o cliente quando ele se reconecta.

- As assinaturas duráveis serão salvas no daemon somente se a persistência estiver configurada. Consulte Salvando mensagens e assinaturas retidas
- As publicações podem ser recebidas de forma síncrona. Chame a função `MQTTClient_receive`.

Retornos de chamada e sincronização

“Retornos de chamadas e sincronização em aplicativos do cliente MQTT” na página 532	O modelo de programação cliente MQTT usa encadeamentos extensivamente. Os encadeamentos desacoplam um aplicativo cliente MQTT, o máximo que puderem, a partir dos atrasos na transmissão de mensagens de e para servidor. Publicações, tokens de entrega e eventos perdidos de conexão são entregues para os métodos em uma classe de retorno de chamada que implementa `MqttCallback`.

- A operação de sincronização no cliente C é modal. A chamada de `MQTTClient_setCallback` coloca o cliente em modo assíncrono.
- No modo síncrono, o aplicativo cliente deve voluntariamente produzir controle para que o cliente MQTT possa processar reconhecimentos e emitir pings MQTT para manter a rede ativa. Ceda o controle chamando `MQTTClient_receive` ou `MQTTClient_yield`.

Sequências de tópicos e filtros

“Sequências e filtros de tópicos em clientes MQTT” na página 550	As sequências de tópicos e os filtros de tópicos são usados para publicar e para assinar. A sintaxe de sequências e filtros de tópicos em clientes MQTT é basicamente a mesma que as sequências de tópicos no IBM WebSphere MQ.

- O daemon do WebSphere MQ Telemetry para dispositivos manipula o curinga de vários níveis # de forma diferente do WebSphere MQ v7 `/#` deve ser os dois últimos caracteres na sequência de filtro para # para se comportar como um curinga. No WebSphere MQ v7, `../#/..` é um uso válido do curinga multinível. O daemon do WebSphere MQ Telemetry para dispositivos trata o curinga de vários níveis da mesma forma que o WebSphere MQ Broker v6

Qualidade de serviço

“Qualidades de serviço fornecidas por um cliente MQTT” na página 545	Um cliente MQTT fornece três qualidades de serviço para entregar publicações para o WebSphere MQ e para o cliente MQTT : "no máximo uma vez", "pelo menos uma vez" e "exatamente uma vez". Quando um cliente MQTT envia uma solicitação ao WebSphere MQ para criar uma assinatura, a solicitação é enviada com a qualidade de serviço "no mínimo uma vez".

- Não há diferenças.

Persistência de mensagem

“Persistência de Mensagem em Clientes MQTT” na página 541	As mensagens de publicação serão transformadas em persistentes se forem enviadas com uma qualidade de serviço de "pelo menos uma vez" ou "exatamente uma vez". É possível implementar seu próprio mecanismo de persistência no cliente ou usar o mecanismo de persistência padrão fornecido com o cliente. A persistência funciona em ambas as direções, para publicações enviadas para o cliente ou por meio dele.

- Devido a diferenças de ligação de linguagem, configure o mecanismo de persistência de mensagem no cliente C como a seguir. Chame o cliente MQTT C com uma das três opções configuradas como o quarto parâmetro para `MQTTClient_create`:

  `MQTTCLIENT_PERSISTENCE_DEFAULT`
  A persistência baseada em arquivo, os detalhes que são específicos para a plataforma do cliente.

  `MQTTCLIENT_PERSISTENCE_NONE`
  Os dados são apenas mantidos na memória e perdidos quando o cliente para. O daemon do WebSphere MQ Telemetry para dispositivos suporta apenas essa opção

  `MQTTCLIENT_PERSISTENCE_USER`
  É possível desenvolver funções para implementar seu próprio mecanismo de persistência. Passe uma estrutura, `MQTTClient_persistence` que contém ponteiros para as funções na chamada `MQTTClient_create`. Leia as informações de referência do cliente MQTT C para obter detalhes.

Sessões limpas

"Sessões limpas" na página 535	O cliente MQTT e o serviço de telemetria (MQXR) mantém as informações de estado da sessão. As informações de estado são usadas para assegurar a entrega "pelo menos uma vez" e "exatamente uma vez" e o recebimento "exatamente uma vez" das publicações. O estado da sessão também inclui assinaturas criadas por um cliente MQTT. É possível optar por executar um cliente MQTT com ou sem manter informações de estado entre as sessões. Altere o modo de sessão limpa configurando `MqttConnectOptions.cleanSession` antes de conectar.

- Não há diferenças.

Último Desejo e Testamento

"Publicação last will and testament" na página 540	Se uma conexão do cliente MQTT for encerrada inesperadamente, será possível configurar o WebSphere MQ Telemetry para enviar uma publicação de "último testamento". Predefina o conteúdo da publicação e o tópico para enviá-lo. A "last will and testament" é uma propriedade da conexão. Crie-o antes de conectar o cliente.

- Não há diferenças.

Manipulando erros do programa

Estas informações explicarão os erros associados às chamadas MQI de aplicativos, ao realizar uma chamada ou quando sua mensagem for entregue para seu destino final.

Sempre que possível, o gerenciador de filas retornará quaisquer erros assim que uma chamada MQI for realizada. Elas são *erros determinados localmente*.

Ao enviar mensagens para uma fila remota, os erros poderão não estar aparentes quando a chamada MQI for realizada. Nesse caso, o gerenciador de filas que identifica os erros os relatam enviando outra mensagem para o programa de origem. Elas são *erros determinados remotamente*.

Erros determinados localmente

Informações sobre erros localmente determinados que incluem: falha em uma chamada MQI, interrupções do sistema e mensagens contendo dados incorretos.

As três causas mais comuns de erros que o gerenciador de filas pode relatar imediatamente são:

- falha de uma chamada MQI; por exemplo, porque a fila está cheia

- Uma interrupção à execução de alguma parte do sistema do qual seu aplicativo depende; por exemplo, o gerenciador de filas
- Mensagens que contêm dados que não podem ser processados com sucesso

Se você estiver usando o recurso de entrada assíncrona, os erros não são relatados imediatamente. Use a chamada MQSTAT para recuperar informações de status sobre operações de entrada assíncronas anteriores.

Falha de uma chamada MQI

O gerenciador de filas pode relatar imediatamente quaisquer erros na codificação de uma chamada MQI. Ele faz isto usando um conjunto de códigos de retorno predefinidos. Esses são divididos em códigos de conclusão e códigos de razão.

Para mostrar se uma chamada foi bem-sucedida, o gerenciador de filas retorna um *código de conclusão* quando a chamada é concluída. Há três códigos de conclusão, indicando êxito, conclusão parcial e falha da chamada. O gerenciador de filas também retorna um *código de razão* que indica a razão para a conclusão parcial ou a falha da chamada.

Os códigos de conclusão e de razão para cada chamada são listados com a descrição dessa chamada em Códigos de retorno. Para obter informações mais detalhadas, incluindo ideias para ação corretiva, consulte:

- Códigos de razão para todas as outras plataformas WebSphere MQ

Projete seus programas para manipular todos os códigos de retorno que podem surgir a partir de cada chamada.

Interrupções do sistema

Seu aplicativo pode não estar ciente de qualquer interrupção se o gerenciador de filas ao qual ele está conectado tiver de se recuperar de uma falha do sistema. No entanto, deve-se projetar seu aplicativo para assegurar que os dados não sejam perdidos se ocorrer uma interrupção.

Os métodos que é possível usar para assegurar que seus dados permaneçam consistentes depende da plataforma na qual o gerenciador de filas está em execução:

Sistemas UNIX, Linux e Windows
Nestes ambientes, é possível fazer suas chamadas MQPUT e MQGET no modo usual, mas deve-se declarar pontos de sincronização usando as chamadas MQCMIT e MQBACK (consulte “Confirmando e fazendo backup de unidades de trabalho” na página 327). No ambiente CICS , os comandos MQCMIT e MQBACK estão desativados, porque é possível fazer suas chamadas MQPUT e MQGET dentro de unidades de trabalho que são gerenciadas pelo CICS

Use mensagens persistentes para transportar todos os dados que não podem ser perdidos. Mensagens persistentes são recolocadas em filas se o gerenciador de filas tiver de se recuperar de uma falha. Com os sistemas WebSphere MQ em UNIX, Linuxe Windows , uma chamada MQGET ou MQPUT dentro de seu aplicativo falhará no ponto de preencher todos os arquivos de log, com a mensagem MQRC_RESOURCE_PROBLEM. Para obter mais informações sobre arquivos de log nos sistemas AIX, HP-UX, Linux, Solaris e Windows , consulte Administrando.

Se o gerenciador de filas for interrompido por um operador enquanto um aplicativo está em execução, a opção quiesce geralmente é usada. O gerenciador de filas entra em um estado quiesce no qual os aplicativos podem continuar a trabalhar, mas eles devem finalizar assim que conveniente. Aplicativos pequenos e rápidos provavelmente podem ignorar o estado de quiesce e continuar até que eles finalizem normalmente. Aplicativos de execução mais longa ou aqueles que aguardam a chegada de mensagens devem usar a opção *fail if quiescing* quando usarem as chamadas MQOPEN, MQPUT, MQPUT1 e MQGET. Essas opções significam que as chamadas falham quando o gerenciador de filas executa quiesce, mas o aplicativo ainda pode ter tempo para finalizar limpamente, emitindo chamadas que ignoram o estado de quiesce. Tais aplicativos também poderiam confirmar ou restaurar as mudanças que foram feitas e, em seguida, finalizar.

Se o gerenciador de filas for forçado a parar (ou seja, parar sem quiesce), os aplicativos receberão o código de razão MQRC_CONNECTION_BROKEN quando eles fazem chamadas MQI. Saia do aplicativo ou, como alternativa, nos sistemas UNIX, Linuxe Windows , emita uma chamada MQDISC

Mensagens contendo dados incorretos

Ao usar unidades de trabalho em seu aplicativo, se um programa não puder processar com êxito uma mensagem que ele recupera de uma fila, a chamada MQGET é restaurada.

O gerenciador de filas mantém uma contagem (no campo *BackoutCount* do descritor de mensagens) do número de vezes que ocorre. Ele mantém essa contagem no descritor de cada mensagem que é afetada. Essa contagem pode fornecer informações valiosas sobre a eficiência de um aplicativo. Mensagens com contagens de restauração que estão aumentando com o tempo estão sendo rejeitadas repetidamente; projete seu aplicativo para que ele analise as razões para isso e manipula tais mensagens, conforme necessário.

No WebSphere MQ para Windows, UNIX e Linux sistemas, a contagem de restaurações sempre sobrevive às reinicializações do gerenciador de filas

Usando as mensagens de relatório para determinação de problemas

O gerenciador de filas remotas não pode relatar erros como uma falha ao colocar uma mensagem em uma fila ao fazer sua chamada MQI, mas ele pode enviar uma mensagem de relatório para dizer como ele processou a mensagem.

No seu aplicativo, é possível criar mensagens de relatório (MQPUT), bem como selecionar a opção para recebê-las (neste caso elas serão enviadas por um outro aplicativo ou por um gerenciador de filas).

Criando mensagens de relatório

Mensagens de relatório ativam um aplicativo para informar outro aplicativo que ele não pode lidar com a mensagem que foi enviada.

No entanto, o campo *Report* deve inicialmente ser analisado para determinar se o aplicativo que enviou a mensagem está interessado em ser informado de qualquer problema. Ao determinar que uma mensagem de relatório é necessária, deve-se decidir:

- Se deseja incluir a mensagem original inteira, incluir apenas os primeiros 100 bytes de dados ou não incluir nenhum da mensagem original.
- O que fazer com a mensagem original. É possível descartá-la ou deixá-la ir para a fila de mensagens não entregues.
- Se o conteúdo do *MsgId* e *CorrelId* campos também são necessários.

Use o campo *Feedback* para indicar a razão para a mensagem de relatório estar sendo gerada. Coloque suas mensagens de relatório na fila de resposta de um aplicativo. Consulte Feedback para obter informações adicionais.

Solicitando e recebendo relatório de mensagens (MQGET)

Ao enviar uma mensagem para outro aplicativo, você não é informado sobre quaisquer problemas, a menos que conclua o campo *Report* para indicar o feedback requerido. Consulte Estrutura do campo de relatório para as opções disponíveis.

Gerenciadores de filas sempre colocam mensagens de relatório na fila de resposta de um aplicativo e é recomendado que seus próprios aplicativos façam o mesmo. Ao usar o recurso de mensagem de relatório, especifique o nome de sua fila de resposta no descritor de mensagens da sua mensagem; caso contrário, a chamada MQPUT falhará.

Seu aplicativo deve conter procedimentos que monitoram sua fila de resposta e processar quaisquer mensagens que cheguem a ela. Lembre-se de que uma mensagem de relatório pode conter toda a mensagem original, os primeiros 100 bytes da mensagem original ou nenhum da mensagem original.

O gerenciador de filas configura o campo *Feedback* da mensagem de relatório para indicar a razão do erro; por exemplo, a fila de destino não existe. Os programas devem fazer o mesmo.

Para obter mais informações sobre mensagens de relatório, consulte “Mensagens de relatório” na página 11.

Erros determinados remotamente

Ao enviar mensagens para uma fila remota, mesmo quando o gerenciador de filas locais tiver processado sua chamada MQI sem encontrar um erro, outros fatores podem influenciar o modo como a mensagem é manipulada por um gerenciador de filas remotas.

Por exemplo, a fila que você está direcionando pode estar cheia ou pode nem sequer existir. Se sua mensagem deve ser tratada por outros gerenciadores de filas intermediários na rota para a fila de destino, qualquer um desses poderia localizar um erro.

Problemas na entrega de uma mensagem

Quando uma chamada MQPUT falhar, é possível tentar colocar a mensagem na fila novamente, retorne-a para o emissor ou coloque-a na fila de mensagens não entregues.

Cada opção tem seus méritos, mas você pode não desejar tentar colocar uma mensagem novamente se a razão pela qual MQPUT falhou tiver sido porque a fila de destino estava cheia. Nesta instância, colocá-la na fila de mensagens não entregues permite que você a entregue à fila de destino correta posteriormente.

Tentar novamente entrega da mensagem

Antes da mensagem ser colocada em uma fila de mensagens não entregues, um gerenciador de filas remotas tenta colocar a mensagem na fila novamente se os atributos *MsgRetryCount* e *MsgRetryInterval* tiverem sido configurados para o canal ou se houver um programa de saída de nova tentativa para que ele use (o nome do qual é retido no campo do atributo do canal *MsgRetryExitId*).

Se o campo *MsgRetryExitId* estiver em branco, os valores nos atributos *MsgRetryCount* e *MsgRetryInterval* são usados.

Se o campo *MsgRetryExitId* não estiver em branco, o programa de saída deste nome será executado. Para obter mais informações sobre como usar seus próprios programas de saída, consulte “Programas de Saída de Canal para Canais de Mensagens” na página 402.

Retornar mensagem ao emissor

Você retorna uma mensagem para o emissor solicitando que uma mensagem de relatório seja gerada para incluir tudo da mensagem original.

Consulte “Mensagens de relatório” na página 11 para obter detalhes sobre opções de mensagem de relatório.

Usando a fila de mensagens não entregues

Quando um gerenciador de filas não puder entregar uma mensagem, ele tentará colocar a mensagem em sua fila de mensagens não entregues. Essa fila deve ser definida quando o gerenciador de filas estiver instalado.

Seus programas podem usar a fila de mensagens não entregues da mesma maneira que o gerenciador de filas a usa. É possível localizar o nome da fila de mensagens não entregues ao abrir o objeto do gerenciador de filas (usando a chamada MQOPEN) e consultar a respeito do atributo *DeadLetterQName* (usando a chamada MQINQ).

Quando o gerenciador de fila coloca uma mensagem nessa fila, ele inclui um cabeçalho na mensagem, cujo formato é descrito pela estrutura do cabeçalho de devoluções (MQDLH); consulte MQDLH-Cabeçalho de devoluções . Este cabeçalho inclui o nome da fila de destino e a razão pela qual a mensagem foi colocada na fila de mensagens não entregues. Ela deve ser removida e o problema deve ser resolvido antes que a mensagem seja colocada na fila pretendida. Além disso, o gerenciador de filas muda o

campo *Format* do descritor de mensagens (MQMD) para indicar que a mensagem contém uma estrutura MQDLH.

Estrutura MQDLH

É recomendado incluir uma estrutura MQDLH em todas as mensagens colocadas na fila de devoluções; no entanto, se você pretende usar o manipulador de devoluções fornecido por determinados produtos WebSphere MQ , ***deve-se*** incluir uma estrutura MQDLH em suas mensagens.

A adição do cabeçalho a uma mensagem pode tornar a mensagem muito longa para a fila de mensagens não entregues, portanto, certifique-se sempre de que suas mensagens sejam mais curtas do que o tamanho máximo permitido para a fila de mensagens não entregues em no mínimo o valor da constante MQ_MSG_HEADER_LENGTH. O tamanho máximo de mensagens permitidas em uma fila é determinado pelo valor do atributo *MaxMsgLength* da fila. Para a fila de mensagens não entregues, certifique-se de que este atributo esteja configurado para o máximo permitido pelo gerenciador de filas. Se o seu aplicativo não puder entregar uma mensagem e a mensagem for muito longa para ser colocada na fila de mensagens não entregues, siga o conselho fornecido na descrição da estrutura MQDLH.

Certifique-se de que a fila de mensagens não entregues seja monitorada e que qualquer mensagem que chega dela seja processada. O manipulador da fila de mensagens não entregues é executado como um utilitário em lote e pode ser usado para executar diversas ações em mensagens selecionadas na fila de mensagens não entregues. Para obter detalhes adicionais, consulte “Processamento de fila de mensagens não entregues” na página 559.

Se uma conversão de dados for necessária, o gerenciador de filas converterá as informações do cabeçalho ao usar a opção MQGMO_CONVERT na chamada MQGET. Se o processo que coloca a mensagem for um MCA, o cabeçalho será seguido por todo o texto da mensagem original.

As mensagens colocadas na fila de mensagens não entregues podem ser truncadas se forem muito longas para esta fila. Um possível indício dessa situação é se as mensagens na fila de mensagens não entregues tiverem o mesmo comprimento que o valor do atributo *MaxMsgLength* da fila.

Processamento de fila de mensagens não entregues

Essas informações contêm informações da interface de programação de uso geral ao usar processamento de fila de mensagens não entregues.

O processamento da fila de mensagens não entregues depende dos requisitos do sistema local, mas considere o seguinte ao definir a especificação:

- A mensagem pode ser identificada como tendo um cabeçalho de fila de mensagens não entregues porque o valor do campo de formato no MQMD é MQFMT_DEAD_LETTER_HEADER.
- No WebSphere MQ para z/OS usando CICS, se um MCA colocar essa mensagem na fila de devoluções, o campo *PutApplType* será MQAT_CICS e o campo *PutApplName* será o *ApplId* do sistema CICS seguido pelo nome da transação do MCA
- A razão para que a mensagem seja roteada para a fila de mensagens não entregues está contida no campo *Reason* do cabeçalho da fila de mensagens não entregues.
- O cabeçalho da fila de mensagens não entregues contém detalhes sobre o nome da fila de destino e o nome do gerenciador de filas.
- O cabeçalho da fila de mensagens não entregues contém campos que precisam ser restabelecidos no descritor de mensagens antes que a mensagem seja colocada na fila de destino. São elas:
  1. *Encoding*
  2. *CodedCharSetId*
  3. *Format*
- O descritor de mensagens é o mesmo que PUT pelo aplicativo original, exceto para os três campos mostrados (Encoding, CodedCharSetId e Format).

O aplicativo de fila de mensagens não entregues deve executar um ou mais dos procedimentos a seguir:

- Examinar o campo *Reason*. Uma mensagem pode ter sido colocada por um MCA pelas razões a seguir:

– A mensagem era mais longa do que o tamanho máximo da mensagem para o canal

  A razão é MQRC_MSG_TOO_BIG_FOR_CHANNEL

– A mensagem não pôde ser colocada em sua fila de destino

  A razão é qualquer código de razão MQRC_* que possa ser retornado por uma operação MQPUT

– Uma saída de usuário solicitou essa ação

  O código de razão é aquele fornecido pela saída de usuário ou o padrão MQRC_SUPPRESSED_BY_EXIT

- Tentar encaminhar a mensagem a seu destino desejado, quando isso for possível.
- Reter a mensagem por um determinado período de tempo antes de descartar quando a razão para o desvio for determinada, mas não imediatamente corrigível.
- Fornecer instruções para administradores corrigirem problemas quando esses tiverem sido determinados.
- Descartar mensagens corrompidas ou que não podem ser processadas.

Há duas maneiras de lidar com as mensagens recuperadas da fila de mensagens não entregues:

1. Se a mensagem for para uma fila local:
   - Execute quaisquer conversões de código necessárias para extrair os dados do aplicativo
   - Execute conversões de código nesses dados se essa for uma função local
   - Coloque a mensagem resultante na fila local com todos os detalhes do descritor de mensagem restaurados
2. Se a mensagem for para uma fila remota, coloque a mensagem na fila.

Para obter informações sobre como as mensagens não entregues são manipuladas em um ambiente de enfileiramento distribuído, consulte O que acontece quando uma mensagem não pode ser entregue?.

Programação multicast

Use estas informações para aprender sobre as tarefas de programação Multicast do WebSphere MQ , como conectar-se a um gerenciador de filas e um relatório de exceção

O WebSphere MQ Multicast foi projetado para ser o mais transparente possível para o usuário e ainda ser compatível com aplicativos existentes. Definir um objeto COMMINFO e configurar os parâmetros **MCAST** e **COMMINFO** do objeto TOPIC significa que os aplicativos WebSphere MQ existentes não requerem regravação substancial para usar multicast. No entanto, pode haver algumas limitações (consulte “Multicast e a Message Queue Interface” na página 560 para obter mais informações) e alguns problemas de segurança a serem considerados (consulte segurança do Multicast para obter mais informações).

Multicast e a Message Queue Interface

Use estas informações para entender os principais conceitos de MQI e como eles estão relacionados ao WebSphere MQ Multicast.

As assinaturas Multicast não são duráveis. Uma vez que não há filas físicas envolvidas, não há lugar para armazenar as mensagens off-line que são criadas pelas assinaturas duráveis.

Após um aplicativo ter inscrito um tópico multicast, ele recebe de volta um manipulador de objetos que ele pode consumir ou do qual pode fazer MQGET, como se fosse um manipulador para uma fila. Isso significa que apenas assinaturas multicast gerenciadas (assinaturas criadas com MQSO_MANAGED) são suportadas, ou seja, não é possível fazer uma assinatura e 'apontar' as mensagens em uma fila. Isso significa que as mensagens devem ser consumidas a partir da manipulação de objetos retornada na chamada de assinatura. No cliente, as mensagens são armazenadas em um buffer de mensagem até que sejam consumidas pelo cliente; veja Sub-rotina MessageBuffer do arquivo de configuração do cliente para obter mais informações. Se o cliente não acompanhar a taxa de publicação, as mensagens serão descartadas, conforme necessário, com as mensagens mais antigas primeiro descartados.

Normalmente é uma decisão administrativa se um aplicativo usa multicast ou não, o que é especificado pela configuração do atributo MCAST de um objeto TOPIC. Se um aplicativo de publicação deve se certificar de que multicast não é usado, ele pode usar a opção `MQOO_NO_MULTICAST`. Da mesma forma, um aplicativo de assinatura pode garantir que o multicast não seja usado ao assinar com a opção `MQSO_NO_MULTICAST`.

WebSphere MQ O Multicast suporta o uso de seletores de mensagem Um seletor é usado por um aplicativo para registrar seu interesse apenas naquelas mensagens com propriedades que satisfazem a consulta SQL92 que a sequência de seleção representa. Para obter mais informações sobre os seletores de mensagem, veja "Seletores" na página 23.

A tabela a seguir lista todos os principais conceitos de MQI e como eles se relacionam com a Multicast:

Tabela 71. conceitos de MQI e como eles se relacionam com multicast

Conceito de MQI	**Ação ao tentar usar multicast**	**Código de razão**
Colocando uma mensagem de comprimento zero	Rejeitado	2005 (07D5) (RC2005): MQRC_BUFFER_LENGTH_ERROR
Agrupamento	Rejeitado	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
Segmentação	Rejeitado	2443 (098B) (RC2443): MQRC_SEGMENTATION_NOT_ALLOWED
Listas de distribuição	Rejeitado	2154 (086A) (RC2154): MQRC_RECS_PRESENT_ERROR
MQINQ	Indicadores para tópicos rejeitados: MQINQ e MQSET de tópicos não são suportado.	2038 (07F6) (RC2038): MQRC_NOT_OPEN_FOR_INQUIRE
	Aceito para manipulador gerenciado. Apenas `Profundidade atual` pode ser consultada.	• Se o valor for de Profundidade Atual, então não há código de razão aplicável. • Se o valor for qualquer outro além da Profundidade Atual, o código de razão é 2067 (0813) (RC2067): MQRC_SELECTOR_ERROR.
MQSET	Rejeitado para todos os identificadores.	2040 (07F8) (RC2040): MQRC_NOT_OPEN_FOR_SET
Transações (XA ou não)	Rejeitado	2072 (0818) (RC2072): MQRC_SYNCPOINT_NOT_AVAILABLE
Mensagem procura	Rejeitado	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
mensagens de bloqueio	Rejeitado	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
Procurar com marca	Rejeitado	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Transmitir contexto	Rejeitado	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR

Tabela 71. conceitos de MQI e como eles se relacionam com multicast (continuação)		
Conceito de MQI	**Ação ao tentar usar multicast**	**Código de razão**
MQPUT1	Rejeitado. É inválido para tentar e MQPUT1 para um único tópico multicast.	2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY
assinatura durável	Rejeitado se o tópico está marcado como "multicast apenas", caso contrário, uma assinatura não multicast é feita.	2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWED
TopicString > 255	Rejeitado. Se a sequência do tópico for maior que 255 caracteres, ela é rejeitada no cliente.	2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR
assinatura não gerenciada feita	Rejeitado se o tópico está marcado como "multicast apenas", caso contrário, uma assinatura não multicast é feita.	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
MQPMO_NOT_OWN_SUBS	Rejeitado	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR

Os seguintes itens expandem-se em alguns dos conceitos MQI da tabela anterior e fornecem informações sobre alguns dos conceitos MQI que não estão na tabela:

Persistência de mensagem
Para assinantes não duráveis de multicast, as mensagens persistentes do publicador são entregues em um modo irrecuperável.

Truncamento da Mensagem
truncamento da mensagem é suportado, o que significa que é possível para um aplicativo:

1. Emita uma chamada MQGET.
2. Obter MQRC_TRUNCATED_MSG_FAILED.
3. Aloque um buffer maior.
4. Emita novamente a chamada MQGET para recuperar a mensagem.

expiração de assinatura
A expiração da assinatura não é suportada. Qualquer tentativa de configurar uma expiração é ignorada.

Alta disponibilidade para multicast

Use estas informações para entender a operação contínua ponto a ponto do WebSphere MQ Multicast; embora o WebSphere MQ se conecte a um gerenciador de filas do WebSphere MQ , as mensagens não fluem através desse gerenciador de filas.

Embora uma conexão com um gerenciador de filas deve ser feita para MQOPEN ou MQSUB, o objeto do tópico de multicast e as próprias mensagens não fluem através do gerenciador de filas. Portanto, após o MQOPEN ou MQSUB ser concluído no objeto de tópico de multicast, será possível continuar a transmitir mensagens multicast, mesmo se a conexão com o gerenciador de filas estiver sido perdida. Existem dois modos de operação:

Uma conexão normal é feita para o gerenciador de filas
A comunicação multicast será possível enquanto a conexão com o gerenciador de filas existir. Se a conexão falhar, as regras de MQI normais são aplicadas, por exemplo; uma MQPUT para o identificador de objeto multicast retorna 2009 (07D9) (RC2009): MQRC_CONNECTION_BROKEN.

Uma conexão do cliente de reconexão é estabelecida com o gerenciador de filas
A comunicação multicast é possível mesmo durante o ciclo de reconexão. Isso significa que, mesmo quando a conexão com o gerenciador de filas foi interrompida, a colocação e o consumo de mensagens multicast não serão afetados. O cliente tentará se reconectar a um gerenciador de filas e se essa reconexão falhar, a manipulação de conexões será interrompida e todas as chamadas MQI, incluindo os multicast, falharão. Para obter mais informações, consulte: Reconexão automatizada do cliente

Se algum aplicativo emitir explicitamente um MQDISC, então todas as assinaturas de multicast e identificadores de objetos serão fechados.

Operação contínua ponto a ponto do multicast

Uma das vantagens da comunicação ponto a ponto entre os clientes é que as mensagens não precisam fluir através do gerenciador de filas; portanto, se a conexão com o gerenciador de filas for interrompida, a transferência de mensagem continuará. As restrições a seguir se aplicam aos requisitos de mensagem contínua deste modo:

- A conexão deve ser feita usando uma das opções MQCNO_RECONNECT_* para operação contínua. Esse processo significa que, embora a sessão de comunicações possa ser interrompida, a manipulação de conexões real não será interrompida e, em vez disso, estará no estado de reconexão. Se a reconexão falhar, a manipulação de conexões agora será interrompida com isso evitará todas as chamadas MQI adicionais.
- Apenas MQPUT, MQGET, MQINQ e Consumo Assíncrono são suportados neste modo. Quaisquer verbos MQOPEN, MQCLOSE ou MQDISC requerem a reconexão com o gerenciador de filas para serem concluídos.
- O status flui para a parada do gerenciador de filas; qualquer estado no gerenciador de filas pode, portanto, ser antigo ou ausente. Isso significa que os clientes podem estar enviando e recebendo mensagens e não há status conhecido no gerenciador de filas. Para obter mais informações, consulte: Monitoramento de aplicativo multicast

Conversão de dados no MQI para o sistema de 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.

Os dados são convertidos no cliente para o tráfego Multicast do WebSphere MQ . Se a opção **MQGMO_CONVERT** for especificada, a conversão de dados será feita conforme solicitada. Os formatos definidos pelo usuário precisam da saída de conversão de dados instalada no cliente; consulte

“Escrevendo saídas de conversão de dados” na página 421 para obter informações sobre quais bibliotecas estão agora nos pacotes do cliente e do servidor.

Para obter informações sobre a administração de conversão de dados, consulte Ativando a conversão de dados para o sistema de mensagens do 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.

Relatório de exceção do Multicast

Use estas informações para aprender sobre WebSphere MQ manipuladores de eventos Multicast e relatar WebSphere MQ Exceções Multicast.

WebSphere MQ Multicast ajuda com a determinação de problemas chamando o manipulador de eventos para relatar eventos multicast que são relatados usando o mecanismo do manipulador de eventos WebSphere MQ padrão.

Um evento Multicast individual pode resultar em mais de um evento do WebSphere MQ sendo chamado porque pode haver vários identificadores de conexão do `MQHCONN` usando o mesmo transmissor ou receptor multicast No entanto, cada exceção multicast faz com que apenas um manipulador de eventos seja chamado por conexão do WebSphere MQ .

A constante WebSphere MQ `MQCBDO_EVENT_CALL` permite que os aplicativos registrem um retorno de chamada para receber apenas eventos do WebSphere MQ e o `MQCBDO_MC_EVENT_CALL` permite que os aplicativos registrem um retorno de chamada para receber apenas eventos multicast Se ambas as constantes forem usadas, ambos os tipos de eventos serão recebidos.

Solicitando eventos do Multicast

WebSphere MQ Eventos multicast usam a constante `MQCBDO_MC_EVENT_CALL` no campo `cbd.Options` . O exemplo a seguir demonstra como solicitar eventos multicast:

```
cbd.CallbackType     = MQCBT_EVENT_HANDLER;
cbd.Options          = MQCBDO_MC_EVENT_CALL;
cbd.CallbackFunction = EventHandler;
MQCB(Hcon,MQOP_REGISTER,&cbd,MQHO_UNUSABLE_HOBJ,NULL,NULL,&CompCode,&Reason);
```

Quando a opção `MQCBDO_MC_EVENT_CALL` é especificada para o campo `cbd.Options` , o manipulador de eventos é enviado apenas WebSphere MQ eventos Multicast em vez de eventos de nível de conexão. Para solicitar que ambos os tipos de eventos sejam enviados ao manipulador de eventos, o aplicativo deve especificar a constante `MQCBDO_EVENT_CALL` no campo `cbd.Options`, bem como a constante `MQCBDO_MC_EVENT_CALL` conforme mostrado no exemplo a seguir:

```
cbd.CallbackType     = MQCBT_EVENT_HANDLER;
cbd.Options          = MQCBDO_EVENT_CALL | MQCBDO_MC_EVENT_CALL
cbd.CallbackFunction = EventHandler;
MQCB(Hcon,MQOP_REGISTER,&cbd,MQHO_UNUSABLE_HOBJ,NULL,NULL,&CompCode,&Reason);
```

Se nenhuma dessas constantes for usada, somente eventos no nível da conexão serão enviadas ao manipulador de eventos.

Para obter mais informações sobre valores para o campo `Options`, consulte Opções (MQLONG).

Formato de evento multicast

WebSphere MQ As exceções multicast incluem algumas informações de suporte que são retornadas no parâmetro **`Buffer`** da função de retorno de chamada. O ponteiro **`Buffer`** aponta para uma matriz de ponteiros e o campo `MQCBC.DataLength` especifica o tamanho, em bytes, da matriz. O primeiro elemento da matriz sempre aponta para uma descrição de texto curta do evento. Mais parâmetros podem ser fornecidos dependendo do tipo de evento. A tabela a seguir lista as exceções:

Tabela 72. Descrições de códigos de eventos multicast		
Código de evento	**Descrição**	**Dados adicionais**
MQMCEV_PACKET_LOSS	Perda de pacote irrecuperável	Número de pacotes perdidos
MQMCEV_HEARTBEAT_TIMEOUT	Ausência longa do pacote de controle de pulsação	N/D
MQMCEV_VERSION_CONFLICT	Recepção de pacotes mais novos de versão de protocolo	N/D
MQMCEV_RELIABILITY	Diferentes modos de confiabilidade do transmissor e do receptor	N/D
MQMCEV_CLOSED_TRANS	A transmissão do tópico é fechada por uma origem	N/D
MQMCEV_STREAM_ERROR	Erro detectado no fluxo	N/D
MQMCEV_NEW_SOURCE	Uma nova origem inicia a transmissão no tópico	Estrutura da origem
MQMCEV_RECEIVE_QUEUE_TRIMMED	Pacotes removidos de PacketQ devido à expiração de tempo ou espaço	Número de pacotes aparados
MQMCEV_PACKET_LOSS_NACK_EXPIRE	Perda de pacote irrecuperável devido à expiração de NACK	Número de pacotes perdidos
MQMCEV_ACK_RETRIES_EXCEEDED	Pacotes removidos do histórico após **max_ack_retries** ter sido excedido	Número de pacotes removidos
MQMCEV_STREAM_SUSPEND_NACK	NACKs foram suspensas em um fluxo aceito por este tópico	Suspender ID do fluxo
		Tempo em milissegundos que o fluxo está suspenso
MQMCEV_STREAM_RESUME_NACK	NACKs foram continuadas após terem sido suspensas em um fluxo	ID do fluxo
MQMCEV_STREAM_EXPELLED	Um fluxo aceito por este tópico foi rejeitado devido a uma solicitação de expulsão	ID do fluxo
MQMCEV_FIRST_MESSAGE	Primeira mensagem de uma origem	Número da mensagem
MQMCEV_LATE_JOIN_FAILURE	Falha ao iniciar sessão se associação postergada	N/D
MQMCEV_MESSAGE_LOSS	Perda de mensagem irrecuperável	Número de mensagens perdidas
MQMCEV_SEND_PACKET_FAILURE	O transmissor multicast falhou ao enviar um pacote multicast	N/D
MQMCEV_REPAIR_DELAY	O receptor multicast não recebeu um pacote de reparo para um NAK pendente	N/D

Tabela 72. Descrições de códigos de eventos multicast (continuação)

Código de evento	Descrição	Dados adicionais
MQMCEV_MEMORY_ALERT_ON	Buffers de recepção do receptor estão ficando cheios	Porcentagem de utilização do buffer pool
MQMCEV_MEMORY_ALERT_OFF	Buffers de recepção do receptor voltaram ao normal	Porcentagem de utilização do buffer pool
MQMCEV_NACK_ALERT_ON	A taxa de solicitações de pacote de reparo do receptor atingiu a marca d'água alta	A taxa de solicitações de reparo atual em pacotes por segundo
MQMCEV_NACK_ALERT_OFF	A taxa de solicitações de pacote de reparo do receptor voltou ao normal	A taxa de solicitações de reparo atual em pacotes por segundo
MQMCEV_REPAIR_ALERT_ON	A taxa de envio de pacote de reparo do transmissor atingiu a marca d'água alta	N/D
MQMCEV_REPAIR_ALERT_OFF	A taxa de envio de pacote de reparo do transmissor voltou ao normal	N/D
MQMCEV_SHM_DEST_UNUSABLE	A região de Memória compartilhada usada por um destino de tópico transmissor foi detectado como inutilizado	N/D
MQMCEV_SHM_PORT_UNUSABLE	A porta de Memória compartilhada usada por uma instância do receptor foi detectada como inutilizada	N/D
MQMCEV_CCT_GETTIME_FAILED	O tempo de get do Tempo do cluster coordenado falhou	N/D
MQMCEV_DEST_INTERFACE_FAILURE	A interface de rede usada por um destino de tópico do transmissor falhou e uma interface de rede de backup está indisponível	
MQMCEV_DEST_INTERFACE_FAILOVER	A interface de rede usada por um destino de tópico do transmissor falhou e um failover bem-sucedido para outra interface foi concluído	
MQMCEV_PORT_INTERFACE-FAILURE	A interface de rede usada por rmmPort de um receptor falhou e uma interface de rede de backup está indisponível (ou também falhou)	configuração de RMM
MQMCEV_PORT_INTERFACE_FAILOVER	A interface de rede usada por rmmPort de um receptor falhou e um failover bem-sucedido para outra interface foi concluído	configuração de RMM

Usando .NET

WebSphere MQ classes para .NET permitem que um programa gravado na estrutura de programação .NET se conecte ao WebSphere MQ como um cliente MQI do WebSphere MQ ou se conecte diretamente a um servidor WebSphere MQ .

Se você tiver aplicativos que usam .NET Framework da Microsoft e desejar tirar vantagem dos recursos do WebSphere MQ, deverá usar classes do WebSphere MQ para .NET.

A interface do WebSphere MQ .NET orientada a objetos é diferente da interface MQI porque ela usa métodos de objetos em vez de usar os verbos MQI.

A interface de programação de aplicativos do WebSphere MQ processual é construída em torno de verbos como aqueles na lista a seguir:

```
MQCONN, MQDISC, MQOPEN, MQCLOSE,
MQINQ, MQSET, MQGET, MQPUT, MQSUB
```

Todos estes verbos utilizam, como um parâmetro, um identificador para o objeto do WebSphere MQ no qual eles devem operar. Como .NET é orientado a objetos, a interface de programação .NET muda isto. Seu programa consiste em um conjunto de objetos do WebSphere MQ, sob o qual você atua chamando métodos nestes objetos. Você pode escrever programas em qualquer linguagem suportada pelo .NET.

Quando você usa a interface processual, desconecta-se de um gerenciador de filas usando a chamada MQDISC(*Hconn*, CompCode, Reason), em que *Hconn* é um identificador para o gerenciador de filas.

Na interface .NET, o gerenciador de filas é representado por um objeto de classe MQQueueManager. Desconecte-se do gerenciador de filas chamando o método disconnect() nessa classe.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.Disconnect();
```

As classes do WebSphere MQ para .NET são um conjunto de classes que permitem aos aplicativos .NET interagir com o WebSphere MQ. Elas representam os vários componentes do WebSphere MQ que seu aplicativo usa, como gerenciadores de filas, filas, canais e mensagens. Para obter detalhes dessas classes, consulte Classes e interfaces do WebSphere MQ .NET.

Antes de poder compilar qualquer aplicativo gravado, é necessário ter um .NET Framework instalado. Para obter instruções sobre como instalar as classes WebSphere MQ para .NET e .NET Framework, consulte “Instalando classes do WebSphere MQ para .NET” na página 568.

Conceitos relacionados

Visão geral técnica

“Opções de Conexão” na página 568
Existem três modos de conectar classes do WebSphere MQ para .NET a um gerenciador de filas. Considere qual tipo de conexão melhor se adequa aos seus requisitos.

“Usando Classes do WebSphere MQ para .NET” na página 579
Esta coleta de tópicos descreve como configurar seu sistema para executar os programas de amostra para verificar sua instalação das classes do WebSphere MQ para .NET e como executar seus próprios programas.

“Resolvendo Problemas do WebSphere MQ .NET” na página 582
Se um programa não for concluído com êxito, execute um dos aplicativos de amostra e siga o conselho fornecido nas mensagens de diagnóstico.

“Gravando e Implementando Programas do WebSphere MQ .NET” na página 582

Para usar classes do WebSphere MQ para .NET para acessar filas do WebSphere MQ, grave programas em qualquer idioma suportado por .NET contendo chamadas que colocam mensagens em, e obtém mensagens de, filas do WebSphere MQ.

"IBM WebSphere MQ canal customizado para Microsoft Windows Communication Foundation (WCF)" na página 602
O canal customizado do Microsoft Windows Communication Foundation (WCF) para o IBM WebSphere MQ envia e recebe mensagens entre clientes e serviços do WCF.

"Decidindo qual linguagem de programação usar" na página 80
Use estas informações para descobrir sobre linguagens de programação e estruturas suportadas pelo IBM WebSphere MQe algumas considerações para usá-las.

"Desenvolvendo Aplicativos" na página 7
O IBM WebSphere MQ fornece várias maneiras nas quais é possível desenvolver aplicativos para enviar e receber mensagens necessárias para suportar seus processos de negócios. É possível também desenvolver os aplicativos para gerenciar os gerenciadores de fila e recursos relacionados.

Introdução às classes WebSphere MQ para .NET

WebSphere MQ classes para .NET permitem que um programa gravado na estrutura de programação .NET se conecte ao WebSphere MQ como um cliente MQI do WebSphere MQ ou se conecte diretamente a um servidor WebSphere MQ .

Opções de Conexão

Existem três modos de conectar classes do WebSphere MQ para .NET a um gerenciador de filas. Considere qual tipo de conexão melhor se adequa aos seus requisitos.

Conexão de Ligações do Cliente

Para usar as classes WebSphere MQ para .NET como um cliente MQI WebSphere MQ , é possível instalá-lo com o cliente MQI WebSphere MQ , na máquina servidor WebSphere MQ ou em uma máquina separada Uma conexão de ligações do cliente pode usar transações XA ou não XA

Conexão de Ligações do Servidor

Quando usadas no modo de ligações do servidor, as classes do WebSphere MQ para .NET usam a API do gerenciador de filas, em vez de se comunicarem através de uma rede. Isto fornece melhor desempenho para aplicativos WebSphere MQ do que usar conexões de rede.

Para usar a conexão de ligações, é necessário instalar classes do WebSphere MQ para .NET no servidor WebSphere MQ.

Conexão do Cliente Gerenciada

Uma conexão feita neste modo se conecta como um cliente WebSphere MQ a um servidor WebSphere MQ em execução na máquina local ou remota.

As classes do WebSphere MQ para .NET que se conectam neste modo permanecem no código gerenciado .NET e não fazem chamadas aos serviços nativos. Para obter informações adicionais sobre código gerenciado, consulte a documentação daMicrosoft.

Existem várias limitações no uso do cliente gerenciado. Para obter mais informações sobre eles, consulte "Conexões do Cliente Gerenciadas" na página 583

Instalando classes do WebSphere MQ para .NET

WebSphere MQ classes para .NET, incluindo amostras, é instalado com o WebSphere MQ. Há um pré-requisito do Microsoft .NET Framework.

A versão mais recente das classes do WebSphere MQ para .NET é instalada por padrão como parte da instalação do WebSphere MQ padrão no recurso *Sistema de Mensagens Java e .NET e Serviços da Web*. Para obter instruções de instalação, consulte Instalando o servidor IBM WebSphere MQ no Windows ou Instalando um cliente IBM WebSphere MQ em Windows sistemas

Em um ambiente de instalação múltipla, se você tiver instalado anteriormente as classes WebSphere MQ para .NET como um pacote de suporte, não será possível instalar o WebSphere MQ a menos que você primeiro desinstale o pacote de suporte. As classes do WebSphere MQ para o recurso .NET instalado com o WebSphere MQ contêm a mesma funcionalidade que o pacote de suporte.

Aplicativos de amostra, incluindo arquivos de origem, também são fornecidos; consulte “Aplicativos de amostra” na página 579.

Para executar as classes WebSphere MQ para .NET em plataformas de 32 bits ou 64 bits, você deve ter instalado o Microsoft .NET Framework V2.0 ou posterior.

Nota: Se o Microsoft .NET Framework v2.0 ou superior não for instalado antes de instalar o WebSphere MQ V7.0.1, a instalação do produto WebSphere MQ continuará sem erro, mas as classes WebSphere MQ para .NET não estarão disponíveis. Se o .NET Framework for instalado após a instalação do WebSphere MQ 7.0.1, as montagens do WebSphere .NET deverão ser registradas executando-se o script *WMQInstallDir*\bin\amqiRegisterdotNet.cmd, em que *WMQInstallDir* é o diretório no qual o WebSphere MQ 7.0.1 é instalado. Este script instala as montagens necessárias no Global Assembly Cache (GAC). Um conjunto de arquivos amqi*.log gravando as ações executadas é criado no diretório %TEMP%.

Para obter informações sobre como usar o canal customizado do WebSphere MQ para o Microsoft WCF com .NET 3, consulte: “IBM WebSphere MQ canal customizado para Microsoft Windows Communication Foundation (WCF)” na página 602

Transações distribuídas no .NET

As transações distribuídas ou transações globais permitem que os aplicativos clientes incluam várias origens diferentes de dados em dois ou mais sistemas em rede em uma transação.

Em transações distribuídas, um gerenciador de transações coordena e gerencia a transação entre dois ou mais gerenciadores de recursos.

Transações podem ser um processamento de single-phase ou two-phase commit. O single-phase commit é um processo no qual somente um gerenciador de recursos participa na transação e o processo two-phase commit é quando há mais de um gerenciador de recursos participando da transação. No processo two-phase commit, o gerenciador de transações envia uma chamada de preparação para verificar se todos os gerenciadores de recursos estão preparados para confirmar. Quando recebe o reconhecimento de todos os gerenciadores de recursos, a chamada de confirmação é emitida. Caso contrário, acontece um retrocesso de toda a transação. Consulte gerenciamento de transações e suporte para obter mais detalhes. Os gerenciadores de recursos devem informar aos gerenciadores de transação de sua participação na transação. Quando o gerenciador de recursos informa ao gerenciador de transações de sua participação, o gerenciador de recursos obtém retornos de chamada do gerenciador de transações quando a transação for confirmada ou retrocedida.

As classes .NET do WebSphere MQ já suportam transações distribuídas em conexões não gerenciadas e no modo de ligação do servidor Nesses modos, as classes WebSphere MQ .NET delegam todas as suas chamadas para o cliente de transação estendida C, que gerencia o processamento de transações em nome do .NET

WebSphere MQ.NET agora suportam transações distribuídas no modo gerenciado em que WebSphere MQ .NET Classes usa o namespace System.Transactions para o suporte de transações distribuídas. A infraestrutura System.Transactions torna a programação transacional simples e eficiente, suportando as transações iniciadas em todos os gerenciadores de recursos, incluindo WebSphere MQ. O aplicativo WebSphere MQ .NET pode colocar e obter mensagens usando a programação de transação implícita .NET ou o modelo de programação de transação explícita. Nas transações implícitas, os limites de transações são criados pelo programa de aplicativo que decide quando confirmar, retroceder (para transações explícitas) ou concluir a transação. Em transações explícitas, é necessário especificar explicitamente se deseja confirmar, retroceder e concluir a transação.

WebSphere MQ.NET usa o Microsoft distributed transaction coordinator (MS DTC) como o gerenciador de transações, que coordena e gerencia a transação entre vários gerenciadores de recurso... O WebSphere MQ é usado como o gerenciador de recursos

WebSphere MQ.NET segue o modelo X/Open Distributed Transaction Processing (DTP). O modelo X/Open Distributed Transaction Processing é um modelo de processamento de transação distribuída proposto pelo Open Group, um consórcio de fornecedores. Esse modelo é um padrão entre a maioria dos fornecedores comerciais nos domínios de banco de dados e processamento de transações. A maioria dos produtos comerciais de gerenciamento de transações suporta o modelo X/DTP.

Modos de transação

- "Transações distribuídas no modo gerenciado" na página 571
- Transações distribuídas para o modo não gerenciado

Coordenando transações em vários cenários

- Uma conexão pode participar de várias transações, mas somente uma transação está ativa a qualquer momento.
- Durante uma transação, a chamada MQQueueManager.Disconnect é honrada. Nesse caso, é solicitado o o retrocesso da transação.
- Durante uma transação, a chamada MQQueue.Close ou MQTopic.Close é honrada. Nesse caso, é solicitado o retrocesso da transação.
- Os limites de transações são criados pelo programa de aplicativo que decide quando confirmar, retroceder (para transações explícitas) ou concluir (para transações implícitas) a transação.
- Se o aplicativo cliente quebrar durante uma transação com um erro inesperado antes de emitir uma chamada Put ou Get em uma chamada de queue ou topic, a transação será retrocedida e uma MQException será lançada.
- Se o código de razão MQCC_FAILED for retornado durante uma chamada Put ou Get em uma chamada queue ou topic, uma MQException será lançada com o código de razão e a transação será movimentada. Se uma chamada de preparação já tiver sido emitida pelo gerenciador de transações, o WebSphere MQ .NET retornará a solicitação de preparação, recuperando forçadamente a transação. Em seguida, o gerenciador da transações DTC causa um retrocesso no trabalho atual com todos os gerenciadores de recursos em transações de ambiente atuais.
- Durante uma transação que envolve diversos gerenciadores de recursos, se alguma razão ambiental fizer com que a chamada Put ou Get seja interrompida indefinidamente, o gerenciador de transações espera um tempo estipulado. Após decorrer esse tempo, causa o retrocesso de todo o trabalho atual com todos os gerenciadores de recursos em transações de ambiente atuais. Se essa espera indefinida ocorrer durante a fase de preparação, o gerenciador de transações pode atingir o tempo limite ou emitir uma chamada em dúvida no recurso, nesse caso, a transação é retrocedida.
- Aplicativos usando transações devem efetuar Put ou Get de mensagens sob SYNC_POINT. Se uma chamada Put ou Get de mensagem for emitida em um contexto transacional que não esteja sob SYNC_POINT, a chamada falhará com o código de razão MQRC_UNIT_OF_WORK_NOT_STARTED.

Diferenças comportamentais entre o suporte a transações do Cliente Gerenciado e Não Gerenciadas usando o namespace Microsoft .NET System.Transactions

As transações aninhadas têm um TransactionScope dentro de outro TransactionScope

- O cliente totalmente gerenciado do WebSphere MQ .NET suporta o TransactionScope aninhado
- O cliente não gerenciado do WebSphere MQ .NET não suporta TransactionScope aninhado

Transações dependentes de System.Transactions

- O cliente totalmente gerenciado do WebSphere MQ .NET suporta o recurso de transações dependentes fornecido pelo System.Transactions

- O cliente não gerenciado do WebSphere MQ .NET não suporta o recurso de transações dependentes fornecido pelo System.Transactions

Amostras do produto

As novas amostras do produto `SimpleXAPute` `SimpleXAGet` estão disponíveis em `WebSphere MQ\tools\dotnet\samples\cs\base` As amostras são aplicativos C#, que demonstram como usar MQPUT e MQGET sob Transações distribuídas usando o namespace SystemTransactions. Para obter mais informações sobre essas amostras, consulte “Criando mensagens simples de put e get em um TransactionScope” na página 574

Transações distribuídas no modo gerenciado

As classes .NET do WebSphere MQ usam o namespace System.Transactions para o suporte de transações distribuídas no modo gerenciado No modo gerenciado, MS DTC coordena e gerencia as transações distribuídas em todos os servidores envolvidos em uma transação.

As classes do WebSphere MQ .NET fornecem um modelo de programação explícito com base na classe System.Transactions.Transaction e um modelo de programação implícito usando a classe System.Transactions.TransactionScope, na qual as transações são gerenciadas automaticamente pela infraestrutura

Transação Implícita

A parte de código a seguir descreve como um aplicativo WebSphere MQ .NET coloca uma mensagem usando a programação de transação implícita .NET.

```
Using (TransactionScope scope = new TransactionScope ())
{
    Q.Put (putMsg,pmo);
    scope.Complete ();
}

Q.close();
qMgr.Disconect();}
```

Explicação do fluxo de código de transação implícita

O código cria *TransactionScope* e coloca a mensagem sob o escopo. Ele, então, chama *Concluído* para informar o coordenador de transação da conclusão da transação. O coordenador de transação agora emite *prepare* e *commit* para concluir a transação. Se um problema for detectado, um *retrocesso* é chamado.

Transação Explícita

O código a seguir descreve como um aplicativo WebSphere MQ .NET coloca mensagens usando o modelo de programação de transação explícito .NET.

```
MQQueueManager qMgr = new MQQueuemanager ("MQQM);
MQQueue Q = QMGR.AccessQueue("Q", MQC.MQOO_OUTPUT+MQC.MQOO_INPUT_SHARED);
MQPutMessageOptions pmo = new MQPutMessageOptions();
pmo.Options = MQC.MQPMO_SYNCPOINT;
MQMessage putMsg1 = new MQMessage();
Using(CommittableTransaction tx = new CommittableTransaction()){
Transaction.Current = tx;
    try
    {
    Q.Put(MSG,pmo);
    tx.commit();
    }
    catch(Exception)
    {tx.rollback();}
    }

Q.close();
qMgr.Disconnect();
}
```

Explicação do fluxo de código de transação explícita

A parte do código cria transação usando a classe *CommitableTransaction*. Ele coloca uma mensagem sob esse escopo e, em seguida, chama explicitamente *commit* para concluir a transação. Se houver qualquer problema, o *retrocesso* é chamado.

Transações distribuídas no modo não gerenciado

WebSphere MQ.NET suportam conexões não gerenciadas (cliente) usando cliente de transação estendido e COM + /MTS como o coordenador de transação, usando o modelo de programação de transação implícito ou explícito. No modo não gerenciado, as classes .NET do WebSphere MQ delegam todas as suas chamadas para o cliente de transação estendida C que gerencia o processamento de transações em nome do .NET

O processamento de transações é controlado por um gerenciador de transações externo, coordenando a unidade de trabalho global sob o controle da API do gerenciador de transações. Os verbos MQBEGIN, MQCMIT e MQBACK estão indisponíveis. As classes .NET do WebSphere MQ expõem esse suporte por meio de seu modo de transporte não gerenciado (cliente C) Consulte Configurando gerenciadores de transações compatíveis com XA

O MTS é desenvolvido como um sistema de processamento de transações (TP) para fornecer os mesmos recursos no Windows NT disponíveis no CICS, Tuxedo e em outras plataformas. Quando o MTS é instalado, um serviço separado é incluído no Windows NT chamado de Microsoft Distributed Transaction Coordinator (MSDTC). O MSDTC coordena as transações que abrangem armazenamentos de dados ou recursos separados. Para funcionar, ele requer que cada armazenamento de dados implemente seu próprio gerenciador de recursos proprietário.

WebSphere MQ se torna compatível com MSDTC implementando uma interface (interface do gerenciador de recursos do proprietário) na qual ele gerencia para mapear chamadas DTC XA para WebSphere MQ(X/Open) chamadas. O WebSphere MQ desempenha a função de um gerenciador de recursos

Quando um componente como COM + solicita acesso a um WebSphere MQ, o COM geralmente verifica com o objeto de contexto MTS apropriado se uma transação for necessária. Se uma transação for necessária, o COM informará o DTC e iniciará automaticamente uma transação integral do WebSphere MQ para essa operação Em seguida, o COM funciona com os dados através do software MQMTS, colocando e obtendo mensagens conforme necessário. A instância do objeto obtida do COM chamará o método SetComplete ou SetAbort, após todas as ações nos dados serem finalizadas. Quando o aplicativo emitir SetComplete, a chamada sinalizará o DTC que o aplicativo concluiu a transação e o DTC poderá prosseguir com o processo de two-phase commit. Em seguida, o DTC emite chamadas para MQMTS que, por sua vez, emite chamadas para o WebSphere MQ para confirmar ou recuperar a transação.

Gravando um aplicativo .NET do WebSphere MQ usando cliente não gerenciado

Para executar dentro do contexto de COM +, uma classe .NET deve herdar do Sistema.EnterpriseServices.ServicedComponent As regras e as recomendações para criarem conjuntos que usam componentes atendidos são as seguintes:

Nota: As etapas a seguir serão relevantes somente se você estiver usando o modo System.EnterpriseServices.

- A classe e o método que estão sendo iniciados em COM+ devem ambos serem públicos (sem classes internas e nenhum método estático ou protegido).
- Os atributos de método e classe: o atributo TransactionOption determina o nível de transação da classe, ou seja, se as transações são desativadas, suportadas ou necessárias. O atributo AutoComplete no método ExecuteUOW() instruirá o COM+ a confirmar a transação, se nenhuma exceção não manipulada for lançada.
- Nomeando fortemente uma montagem: o conjunto deve ser fortemente nomeado e registrado no Global Assembly Cache (GAC). A montagem será registrada em COM+ explicitamente ou por registro lento depois que ela for registrada no GAC.
- Registrando uma montagem em COM: prepare o conjunto a ser exposto aos clientes COM. Em seguida, crie uma biblioteca de tipos usando a ferramenta Assembly Registration, regasm.exe.

```
        regasm UnmanagedToManagedXa.dll
```

- Registre a montagem no GAC `gacutil /i UnmanagedToManagedXa.dll`.
- Registre o conjunto em COM + usando a ferramenta do instalador de serviços do .NET, regsvcs.exe Consulte a biblioteca de tipos criada por regasm.exe:

```
Regsvcs /appname:UnmanagedToManagedXa  /tlb:UnmanagedToManagedXa.tlb
UnmanagedToManagedXa.dll
```

- A montagem é implementada no GAC e posteriormente registrada em COM+ pelo registro lento. O .NET Framework cuida do registro após o código ser executado pela primeira vez.

O fluxo de código de exemplo usando o modelo System.EnterpriseServices e System.Transactions com COM+ são descritos nas seções a seguir:

Fluxo de código de exemplo usando o modelo System.EnterpriseServices

```
using System;
using IBM.WMQ;
using IBM.WMQ.Nmqi;
using System.Transactions;
using System.EnterpriseServices;

namespace UnmanagedToManagedXa
{

[ComVisible(true)]
[System.EnterpriseServices.Transaction(System.EnterpriseServices.TransactionOption.Required)
]
    public class MyXa : System.EnterpriseServices.ServicedComponent
    {

        public MQQueueManager QMGR = null;
        public MQQueueManager QMGR1 = null;
        public MQQueue QUEUE = null;
        public MQQueue QUEUE1 = null;
        public MQPutMessageOptions pmo = null;
        public MQMessage MSG = null;

        public MyXa()
        {
        }

        [System.EnterpriseServices.AutoComplete()]
        public void ExecuteUOW()
        {
            QMGR = new MQQueueManager("usemq");

            QUEUE = QMGR.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE",
                                     MQC.MQOO_INPUT_SHARED +
                                     MQC.MQOO_OUTPUT +
                                     MQC.MQOO_BROWSE);

            pmo = new MQPutMessageOptions();
            pmo.Options = MQC.MQPMO_SYNCPOINT;
            MSG = new MQMessage();
            QUEUE.Put(MSG, pmo);
            QMGR.Disconnect();
        }
    }
}


public void RunNow()
{
   MyXa xa = new MyXa();
   xa.ExecuteUOW();
}
```

Fluxo de código de exemplo usando System.Transactions para interações com COM+

```
[STAThread]
public void ExecuteUOW()
{
Hashtable t1 = new Hashtable();
t1.Add(MQC.CHANNEL_PROPERTY, "SYSTEM.DEF.SVRCONN");
```

```
t1.Add(MQC.HOST_NAME_PROPERTY, "localhost");
t1.Add(MQC.PORT_PROPERTY, 1414);
t1.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_CLIENT);
TransactionOptions opts = new TransactionOptions();

using(TransactionScope scope = new  TransactionScope(TransactionScopeOption.RequiresNew,
                                opts,     EnterpriseServicesInteropOption.Full)
  {

        QMGR = new MQQueueManager("usemq", t1);
        QUEUE = QMGR.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE",
                                          MQC.MQOO_INPUT_SHARED +
                                          MQC.MQOO_OUTPUT +
                                          MQC.MQOO_BROWSE);

         pmo = new MQPutMessageOptions();
         pmo.Options = MQC.MQPMO_SYNCPOINT;
         MSG = new MQMessage();
         QUEUE.Put(MSG, pmo);
         scope.Complete();
  }
 QMGR.Disconnect();
}
```

Criando mensagens simples de put e get em um TransactionScope

Aplicativos C# de amostra do produto estão disponíveis no WebSphere MQ. Estes aplicativos simples demonstram a colocação e obtenção de mensagens em um TransactionScope. Ao final da tarefa, será possível colocar e obter mensagens de uma fila ou tópico.

Antes de começar

O serviço MSDTC deve estar em execução e ativado para transações XA.

Sobre esta tarefa

O exemplo é um aplicativo simples, `SimpleXAPut` e `SimpleXAGet`. Os programas `SimpleXAPut` e `SimpleXAGet` são aplicativos C# disponíveis dentro do WebSphere MQ O `SimpleXAPut` demonstra o uso de MQPUT sob Transações distribuídas usando o namespace SystemTransactions. `SimpleXAGet` demonstra o uso de MQGET sob nomes transações distribuídas usando o namespace SystemTransactions.

`SimpleXAPut` está localizado em `WebSphere MQ\tools\dotnet\samples\cs\base`

Procedimento

Os aplicativos podem ser executados com os parâmetros da linha de comandos de `tools\dotnet\samples\cs\base\bin`

```
SimpleXAPut.exe -d destinationURI [-h host -p port -l channel -tx transaction -tm mode -n
numberOfMsgs]
```

```
SimpleXAGet.exe -d destinationURI [-h host -p port -l channel -tx transaction -tm mode -n
numberOfMsgs]
```

em que os parâmetros são:

`-destinationURI`

Isso pode ser fila ou tópico. Para uma fila, especifique como `queue://queueName` e para um tópico especifique como `topic://topicName`.

`-host`

Isso pode ser um nome de host como host local ou um endereço IP.

`-port`

A porta na qual o gerenciador de filas está em execução.

-channel

O canal de conexão que está sendo usado. O padrão é SYSTEM.DEF.SVRCONN

-transaction

O resultado da transação, por exemplo de confirmação ou retrocesso.

-mode

O modo de transporte, por exemplo, gerenciado ou não gerenciado.

-numberOfMsgs

O número de mensagens. O padrão é 1.

Exemplo

```
SimpleXAPut -d topic://T01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

```
SimpleXAGet -d queue://Q01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

Recuperando transações

Esta seção descreve o processo de recuperação de transações no WebSphere MQ .NET XA usando o modo gerenciado

Visão Geral

No processamento de transações distribuído, as transações podem ser concluídas com sucesso. No entanto, pode haver cenários em que uma transação pode falhar por várias razões. Essas razões podem incluir uma falha do sistema, falha de hardware, erro de rede, dados incorretos ou inválidos, erros de aplicativo ou desastres naturais ou artificiais. Não é possível evitar falhas de transação. O sistema de transação distribuída deve ser capaz de manipular estas falhas. Ele deverá ser capaz de detectar e corrigir erros quando eles ocorrem. Esse processo é conhecido como Recuperação da transação.

Um aspecto importante do Distributed Transaction Processing é recuperar as transações incompletas ou indeterminadas. É essencial para executar a recuperação como parte da Unidade de trabalho de uma transação específica mantida bloqueada até que seja recuperada. O Microsoft .NET de sua biblioteca de classe System.Transactions fornece a opção para recuperar transações incompletas / indeterminadas. Este suporte a recuperação espera o Resource Manager para manter os logs de transações e executar a recuperação quando necessário.

Modelo de recuperação

No modelo de recuperação de transação do Microsoft .NET, o Transaction Manager (System.Transactionsou Microsoft Distributed Transaction coordinator (MS DTC) ou ambos) inicia, coordena e controla a recuperação de transação. Os gerenciadores de recursos baseados em OLE Tx Protocol (protocolo XA da Microsoft) fornecem as opções para configurar o DTC para conduzir, coordenar e controlar a recuperação para eles. Para fazer isso, os gerenciadores de recurso devem registrar XA_Switch com o MS DTC usando a interface nativa.

XA_Switch fornece os pontos de entrada de funções XA como xa_start, xa_end e xa_recover no gerenciador de recursos para o Distributed Transaction Coordinator.

Recuperação usando o Microsoft Distributed Transaction coordinator (DTC):

O coordenador de transações distribuídas da Microsoft fornece dois tipos de processos de recuperação.

Recuperação inativa

A recuperação inativa será executada se o processo do gerenciador da transação falhar enquanto uma conexão com um gerenciador de recursos XA estiver aberta. Quando o gerenciador de transações for

reiniciado, ele lerá os logs do gerenciador de transações e restabelecerá a conexão com o gerenciador de recursos XA e, em seguida, iniciará a recuperação.

Recuperação ativa

A recuperação ativa será executada, se o gerenciador de transações permanecer ativo enquanto a conexão entre o gerenciador de transações e o gerenciador de recursos XA falhar porque o gerenciador de recursos XA ou a rede falhou. Após a falha, o gerenciador de transações tentará periodicamente se reconectar ao gerenciador de recursos XA. Quando a conexão for restabelecida, o gerenciador de transações iniciará a recuperação XA.

O namespace de System.Transactions fornece implementação gerenciada de transações distribuídas baseadas em MS DTC como o gerenciador de transações. Ele fornece recursos semelhantes à interface nativa do MS DTC, mas em um ambiente totalmente gerenciado. A única diferença é sobre a recuperação da transação. System.Transactions espera o gerenciadores de recursos para conduzir a recuperação por si só, então, coordena com o Transaction Managers (MS DTC). O gerenciador de recursos deve solicitar a recuperação de uma transação específica incompleta e, em seguida, o Transaction Manager a aceita e coordena baseado no resultado real dessa transação específica.

Processo de recuperação de transação para o WebSphere MQ .NET

Esta seção descreve como as transações distribuídas podem ser recuperadas com as classes .NET do WebSphere MQ

Visão Geral

Para recuperar uma transação incompleta as informações de recuperação são necessárias. As informações de recuperação da transação devem ser registradas para armazenamento pelos gerenciadores de recursos. As classes .NET do WebSphere MQ seguem um caminho semelhante As informações de recuperação da transação serão registradas em uma fila do sistema chamada SYSTEM.DOTNET.XARECOVERY.QUEUE.

A recuperação de transação no WebSphere MQ .NET é um processo de dois estágios

1. O registro de informações de recuperação da transação.
   - Para cada transação, durante a fase de preparação, uma mensagem persistente contendo as informações de recuperação é incluída em SYSTEM.DOTNET.XARECOVERY.QUEUE.
   - A mensagem é excluída se a chamada de confirmação for bem-sucedida.
2. Recuperando Transações usando um WmqDotnetXAMonitor aplicativo do monitor.
   - WmqDotnetXAMonitor é um aplicativo gerenciado pelo .NET que processa mensagens no SYSTEM.DOTNET.XARECOVERY.QUEUE e recupera transações incompletas

   Se o MCA não conseguir colocar a mensagem na fila de destino, ele gera um relatório de exceções que contém a mensagem original e a coloca em uma fila de transmissão a ser enviada para a fila de resposta especificada na mensagem original. (Se a fila de resposta estiver no mesmo gerenciador de filas que o MCA, a mensagem será colocada diretamente nessa fila, e não em uma fila de transmissão)

SYSTEM.DOTNET.XARECOVERY.QUEUE

Essa é uma fila do sistema que contém informações de recuperação da transação de transações incompletas. Essa fila é criada quando um gerenciador de filas é criado.

Nota: Não é necessário excluir a fila SYSTEM.DOTNET.XARECOVERY.QUEUE.

Aplicativo WMQDotnetXAMonitor

O aplicativo WebSphere MQ .NET XA Monitor monitora um determinado gerenciador de filas e recupera transações incompletas, se houver. A seguir, são consideradas como transações incompletas e são recuperadas:

Transações Incompletas

- Se a transação estiver preparada, mas COMMIT não foi concluído dentro do período de tempo limite.
- Se a transação estiver preparada, mas o gerenciador de filas do WebSphere MQ tiver sido desativado.
- Se a transação estiver preparada, mas, em seguida, o Transaction Manager tiver sido desativado.

  O aplicativo Monitor deve ser executado a partir do mesmo sistema em que o aplicativo cliente WebSphere MQ .NET está em execução. Se houver aplicativos em execução em sistemas múltiplos que se conectam ao mesmo gerenciador de filas, o aplicativo monitor deverá ser executado a partir de todos os sistemas. Embora cada máquina cliente tenha um aplicativo monitor em execução para recuperar o aplicativo, cada monitor deve ser capaz de identificar a mensagem que corresponde à transação que o MS DTC local do monitor atual estava coordenando para que possa realistá-la e concluí-la.

Casos de uso de recuperação de transação para o WebSphere MQ .NET

A seguir estão os diferentes cenários de uso:

- **WebSphere MQ Aplicativo usando uma única instância do DTC e do Gerenciador de Filas:** Neste cenário, quando você se conecta ao gerenciador de filas e executa a Unidade de Trabalho (UoW) sob a transação e se a transação falhar e ficar incompleta, o aplicativo de monitor recupera a transação e a conclui.

  Neste cenário, haverá uma única instância do aplicativo de monitor em execução, pois um único Gerenciador de Filas está associado às transações
- **Vários WebSphere MQ usando uma única instância do DTC e do Gerenciador de Filas:** Neste cenário, há mais de um aplicativo WMQ sob um único DTC e todos estão se conectando ao mesmo gerenciador de filas e executando UoW sob transações

  Se as transações falharem e se tornarem incompletas, o aplicativo de monitor irá recuperá-los e conclui as transações referentes a todos os aplicativos.

  Neste cenário, um único aplicativo de monitor é executado, já que um gerenciador de filas é usado em transações
- **Vários aplicativos WebSphere MQ , diversos DTCs, diferentes instâncias do Gerenciador de filas:** Neste cenário, há mais de um aplicativo WMQ sob diferentes DTCs (ou seja, cada aplicativo está em execução em uma máquina diferente) e se conectando a diferentes gerenciadores de filas.

  Se a falha ocorrer e a transação tornar-se incompleta, o aplicativo de monitor verificará o TransactionManagerWhereabouts na mensagem para determinar o endereço DTC. Se o valor TransactionManagerWhereabouts corresponder ao endereço DTC no qual o monitor está em execução, ele concluirá a recuperação, continuará a procura até que a mensagem correspondente ao DTC seja localizada.

  Neste cenário, haverá apenas uma instância de aplicativo de monitor em execução por cliente (usuário ou computador), pois cada cliente tem seu próprio gerenciador de fila usado em transações...
- **Vários aplicativos WebSphere MQ , DTCs múltiplos, instâncias de Gerenciadores de Filas diferentes:** Neste cenário, há mais de um aplicativo WMQ em DTCs diferentes (que é cada aplicativo está em execução em uma máquina diferente) e todos estão se conectando ao mesmo gerenciador de filas.

  Se a falha ocorrer e a transação se tornar incompleta, o aplicativo de monitor verificará o TransactionManagerWhereabouts na mensagem para verificar se o endereço DTC e o valor correspondem com o DTC no qual o monitor está executando. Se ambos os valores corresponderem, ele concluirá a outra recuperação que continuará a procura até localizar a mensagem correspondente a seu DTC.

  Neste cenário, haverá apenas uma única instância de aplicativo de monitor em execução por cliente (usuário ou computador), pois cada cliente tem sua própria associação de gerenciador de filas usada em transações...
- **Vários WebSphere MQ , DTC único, instâncias diferentes do Gerenciador de Filas:** Neste cenário, há mais de um aplicativo WMQ sob um único DTC (ou seja, em um computador, há mais de um aplicativo WMQ em execução) e se conectando a diferentes gerenciadores de filas..

Se a transação falhar e se tornar incompleta, o aplicativo de monitor recuperará a transação.

Neste cenário, haverá tantas instâncias de aplicativo de monitor em execução quanto gerenciadores de filas conectados, pois cada aplicativo tem seu próprio gerenciador de filas usado em transações e cada um deles deve ser recuperado.

Nota: Se o aplicativo de monitor não estiver sendo executado em segundo plano, será possível iniciá-lo.

Usando o aplicativo WMQDotnetXAMonitor

O aplicativo do monitor XA deve ser executado manualmente Ele pode ser iniciado a qualquer momento. É possível iniciá-lo quando você vir as mensagens no SYSTEM.DOTNET.XARECOVERY.QUEUE ou você pode mantê-lo em execução em segundo plano antes de executar qualquer trabalho transacional com os aplicativos que são gravados usando as classes do WebSphere MQ .NET

Comando para iniciar o aplicativo monitor

```
WmqDotnetXAMonitor.exe -m <QueueManagerName> -n <ConnectionName> -c <Channel> -i
```

Em que

- **n** -Nome de conexão no formato de host (porta). O nome da conexão pode conter mais de um nome de conexão. Vários nomes de conexão devem ser fornecidos na lista separada por vírgula, por exemplo, "localhost (1414), localhost (1415), localhost (1416)". O aplicativo de monitor executa a recuperação para cada um dos nomes de conexão especificados na lista separada por vírgula.
- **c** -Nome do canal
- **m** -Nome do gerenciador de filas. Opcional
- **i** -Conclusão da ramificação heurística Opcional

O aplicativo de monitor executa as ações a seguir:

1. Verifica a profundidade da fila de SYSTEM.DOTNET.XARECOVERY.QUEUE a um intervalo de 100 segundos.
2. Se a profundidade da fila for maior que zero, o monitor de XA procura na fila mensagens e verifica se a mensagem satisfaz os critérios de transação incompleta.
3. Se qualquer uma das mensagens satisfizer os critérios de transação incompleta, o monitor a retira e recupera as informações de recuperação da transação.
4. Em seguida, determina se as informações de recuperação estão relacionadas ao MS DTC local. Se estiverem, então, ele continua para recuperar a transação. Caso contrário, ele volta para procurar a próxima mensagem.
5. Ele então faz chamadas ao gerenciador de filas para recuperar a transação incompleta.

Configurações do arquivo de configuração de aplicativo WmqDotNETXAMonitor

Para monitorar o aplicativo, as entradas também podem ser fornecidas usando o arquivo de configuração de aplicativo. Um arquivo de configuração de aplicativo de amostra é enviado com o WebSphere MQ .NET Esse arquivo pode ser modificado de acordo com seus requisitos.

O arquivo de configuração de aplicativo tem a precedência mais alta ao considerar os valores de entrada. Se ambos os valores de entrada forem fornecidos tanto na linha de comandos como no arquivo de configuração de aplicativo, então os valores da configuração do aplicativo serão consideradas.

Arquivo de configuração de aplicativo de amostra.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<configSections>
<sectionGroup name="IBM.WMQ">
<section name="dnetxa" type="System.Configuration.NameValueFileSectionHandler" />
</sectionGroup>
</configSections>
<IBM.WMQ>
<dnetxa>
<add key="ConnectionName" value=""/>
<add key="ChannelName" value="" />
```

```
<add key="QueueManagerName" value="" />
<add key="UserId" value="" />
<add key="SecurityExit" value="" />
<add key="SecurityExitUserData" value = "">
</dnetxa>
</dnetxa>
</configuration>
```

Log do aplicativo WmqDotNetXAMonitor
O aplicativo de monitor cria um arquivo de log no diretório do aplicativo para a criação de log do progresso do monitor e status de recuperação da transação. A criação de log é iniciada com o nome da conexão e os detalhes do canal para mostrar o gerenciador de filas atual para a qual a recuperação está em execução.

Após a recuperação ser iniciada, o MessageId da mensagem de recuperação da transação, o TransactionId da transação incompleta e o resultado real da transação também pela Coordenação do gerenciador de transação serão registrados.

Arquivo de log de amostra:

```
Time|ProcessId|ThreadId|WMQ .NET XA Recovery Monitor, Running now for
ConnectionName:xxxx, Time|ProcessId|ThreadId|Channel=xxxx
Time|ProcessId|ThreadId|Current QueueDepth = n
Time|ProcessId|ThreadId|Current MessageId = xxxx
Time|ProcessId|ThreadId|Current Incomplete Transaction being recovered = xxxxx
Time|ProcessId|ThreadId|Actual Outcome of the transaction(as per DTC)= Commit/Rollback
Time|ProcessId|ThreadId|Recovery Completed for TransactionId= xxxxx
Time|ProcessId|ThreadId|Current QueueDepth = n
Time|ProcessId|ThreadId|Current MessageId = xxxx
Time|ProcessId|ThreadId|Current Incomplete Transaction being recovered = xxxxx
Time|ProcessId|ThreadId|Actual Outcome of the transaction(as per DTC)= Commit/Rollback
Time|ProcessId|ThreadId| Recovery Completed for TransactionId= xxxxx
```

Usando Classes do WebSphere MQ para .NET

Esta coleta de tópicos descreve como configurar seu sistema para executar os programas de amostra para verificar sua instalação das classes do WebSphere MQ para .NET e como executar seus próprios programas.

Configurando seu Gerenciador de Filas para Aceitar Conexões do Cliente TCP/IP

Para configurar um gerenciador de filas para aceitar pedidos de conexão recebidos dos clientes:

1. Defina um canal de conexão do servidor:
   a. Inicie o gerenciador de filas.
   b. Defina um canal de amostra chamado NET.CHANNEL[3]:

```
DEF CHL('NET.CHANNEL') CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ') +
DESCR('Sample channel for WebSphere MQ classes for .NET')
```

2. Inicie um listener:

```
runmqlsr -t tcp [-m qmnqme] [-p portnum]
```

   Nota: Os colchetes indicam parâmetros opcionais; *qmname* não é necessário para o gerenciador de filas padrão e o número da porta *portnum* não é necessário se você estiver usando o padrão (1414).

Aplicativos de amostra

Para executar seus próprios aplicativos .NET, use as instruções para os programas de verificação, substituindo seu nome do aplicativo no local dos aplicativos de amostra.

Cinco aplicativos de amostra são fornecidos:

[3] Nesta amostra, não estamos considerando implicações de segurança. Para um sistema de produção, considere usar SSL ou uma saída de segurança. Consulte Segurança para obter mais informações..

- Um aplicativo de entrada de mensagem
- Um aplicativo de obtenção de mensagem
- Um aplicativo 'hello world'
- Um aplicativo de publicação/assinatura
- Um aplicativo que usa propriedades de mensagem

Todos esses aplicativos de amostra são fornecidos na linguagem C, e alguns também são fornecidos em C++ e Visual Basic. Você pode escrever aplicativos em qualquer idioma suportado pelo .NET.

Programa de "entrada de mensagem" SPUT (nmqsput.cs, mmqsput.cpp, vmqsput.vb)
Este programa mostra como colocar uma mensagem em uma fila nomeada. O programa possui três parâmetros:

- O nome de uma fila (necessário), por exemplo, SYSTEM.DEFAULT.LOCAL.QUEUE
- O nome de um gerenciador de filas (opcional)
- A definição de um canal (opcional), por exemplo, SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Se nenhum nome do gerenciador de filas for fornecido, o gerenciador de filas será padronizado com o gerenciador de filas locais padrão. Se um canal for definido, ele tem o mesmo formato que a variável de ambiente MQSERVER.

Programa de "obtenção de mensagem" SGET (nmqsget.cs, mmqsget.cpp, vmqsget.vb)
Este programa mostra como obter uma mensagem de uma fila nomeada. O programa possui três parâmetros:

- O nome de uma fila (necessário), por exemplo, SYSTEM.DEFAULT.LOCAL.QUEUE
- O nome de um gerenciador de filas (opcional)
- A definição de um canal (opcional), por exemplo, SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Se nenhum nome do gerenciador de filas for fornecido, o gerenciador de filas será padronizado com o gerenciador de filas locais padrão. Se um canal for definido, ele tem o mesmo formato que a variável de ambiente MQSERVER.

Programa "Hello World" (nmqwrld.cs, mmqwrld.cpp, vmqwrld.vb)
Este programa mostra como colocar e obter uma mensagem. O programa possui três parâmetros:

- O nome de uma fila (opcional), por exemplo, SYSTEM.DEFAULT.LOCAL.QUEUE ou SYSTEM.DEFAULT.MODEL.QUEUE
- O nome de um gerenciador de filas (opcional)
- Uma definição de canal (opcional), por exemplo, SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Se nenhum nome de fila for fornecido, o nome padrão será SYSTEM.DEFAULT.LOCAL.QUEUE. Se nenhum nome do gerenciador de filas for fornecido, o gerenciador de filas será padronizado com o gerenciador de filas locais padrão.

Programa de "publicação/assinatura" (MQPubSubSample.cs)
Este programa mostra como usar publicação/assinatura do WebSphere MQ. Ele é fornecido somente em C#. O programa possui dois parâmetros:

- O nome de um gerenciador de filas (opcional)
- Uma definição de canal (opcional)

Programa de "propriedades de mensagem" (MQMessagePropertiesSample.cs)
Este programa mostra como usar propriedades de mensagem. Ele é fornecido somente em C#. O programa possui dois parâmetros:

- O nome de um gerenciador de filas (opcional)
- Uma definição de canal (opcional)

É possível verificar sua instalação compilando e executando estes aplicativos.

Os aplicativos de amostra estão instalados nos seguintes locais, de acordo com a linguagem na qual são gravados.*MQ_INSTALLATION_PATH* Representa o diretório de alto nível no qual o WebSphere MQ está instalado

C#

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqswrld.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqsput.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqsget.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\MQPubSubSample.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\MQMessagePropertiesSample.cs

C++ Gerenciado

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqswrld.cpp

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqsput.cpp

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqsget.cpp

Visual Basic

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqswrld.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqsput.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqsget.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqswrld.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqsput.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqsget.vb

Para construir os aplicativos de amostra, um arquivo em lote foi fornecido para cada linguagem.

C#

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\bldcssamp.bat

O arquivo bldcssamp.bat contém uma linha para cada amostra, que é tudo o que é necessário para construir este programa de amostra:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib:MQ_INSTALLATION_PATH\bin
/out:nmqwrld.exe nmqwrld.cs
```

C++ Gerenciado

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\bldmcpsamp.bat

O arquivo bldmcpsamp.bat contém uma linha para cada amostra, que é tudo o que é necessário para construir este programa de amostra:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Se você desejar compilar estes aplicativos no Microsoft Visual Studio 2003/.NET SDKv1.1, substitua o comando de compilação:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

com

```
cl /clr MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Visual Basic

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\bldvbsamp.bat

O arquivo bldvbsamp.bat contém uma linha para cada amostra, que é tudo o que é necessário para construir este programa de amostra:

```
vbc /r:System.dll /r:MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:vmqwrld.exe vmqwrld.vb
```

Resolvendo Problemas do WebSphere MQ .NET

Se um programa não for concluído com êxito, execute um dos aplicativos de amostra e siga o conselho fornecido nas mensagens de diagnóstico.

Estes aplicativos de amostra são descritos em "Usando Classes do WebSphere MQ para .NET" na página 579.

Se os problemas continuarem e for necessário contatar a equipe de serviço da IBM, poderá ser solicitado que você ative o recurso de rastreio.

Rastreando o Aplicativo de Amostra

Para obter instruções sobre como usar o recurso de rastreio, consulte "Rastreando Programas WebSphere MQ .NET" na página 602.

Mensagens de erros

Você poderá ver a seguinte mensagem de erro comum:

Uma exceção não manipulada do tipo 'System.IO.FileNotFoundException' ocorreu no módulo desconhecido

Se este erro ocorrer para amqmdnet.dll ou amqmdxcs.dll, assegure que ambos estejam registrados no 'Cache de Montagem Global' ou crie uma arquivo de configuração que aponte para as montagens amqmdnet.dll e amqmdxcs.dll. É possível examinar e alterar o conteúdo do cache de montagem usando mscorcfg.msc, que é fornecido como parte da estrutura .NET.

Se a estrutura .NET estava indisponível quando o WebSphere MQ foi instalado, as classes podem não estar registradas no cache do conjunto global É possível reexecutar manualmente o processo de registro usando o comando

```
amqidnet -c MQ_INSTALLATION_PATH\bin\amqidotn.txt -l logfile.txt
```

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

Informações sobre esta instalação são gravadas no arquivo de log especificado (`logfile.txt` neste exemplo).

Gravando e Implementando Programas do WebSphere MQ .NET

Para usar classes do WebSphere MQ para .NET para acessar filas do WebSphere MQ, grave programas em qualquer idioma suportado por .NET contendo chamadas que colocam mensagens em, e obtém mensagens de, filas do WebSphere MQ.

A documentação do WebSphere MQ contém informações apenas sobre as linguagens C#, C++ e Visual Basic

Esta coleta de tópicos fornece informações para ajudar na gravação de aplicativos para interagir com sistemas WebSphere MQ. Para obter detalhes de classes individuais, consulte As classes e interfaces do WebSphere MQ .NET

Diferenças de Conexão

A maneira como você programa o WebSphere MQ .NET tem algumas dependências nos modos de conexão que você deseja usar.

Conexões do Cliente Gerenciadas

Quando as classes WebSphere MQ para .NET são usadas como um cliente gerenciado, há várias diferenças de um cliente MQI padrão do WebSphere MQ .

Os recursos a seguir não estão disponíveis para um cliente gerenciado:

- Compactação de canal
- Suporte SSL
- Encadeamento de saída do canal

Se você tentar usar estes recursos com um cliente gerenciado, isto retornará uma MQException. Se o erro for detectado na extremidade do cliente de uma conexão, ele usará o código de razão MQRC_ENVIRONMENT_ERROR. Se ele for detectado na extremidade do servidor, o código de razão retornado pelo servidor será usado.

Saídas de canal gravadas para um cliente não gerenciado não funcionam. É necessário gravar novas saídas especificamente para o cliente gerenciado. Verifique se não há saídas de canal inválidas especificadas em sua tabela de client channel definition table (CCDT).

O nome de uma saída do canal gerenciada pode ter até 999 caracteres de comprimento. Entretanto, se você usar a CCDT para especificar o nome de saída do canal, ele será limitado a 128 caracteres.

Comunicação é suportada somente sobre TCP/IP.

Quando você para um gerenciador de filas usando o comando **endmqm**, um canal de conexão do servidor para um cliente gerenciado .NET pode demorar mais para fechar do que os canais de conexão do servidor para outros clientes.

Se você tiver configurado *NMQ_MQ_LIB* para `gerenciado` para usar diagnósticos de problemas gerenciados do WebSphere MQ, nenhum dos parâmetros -i, -p, -s, -b ou -c do comando **strmqtrc** será suportado.

Um aplicativo .NET gerenciado usando transações XA não funcionará com um gerenciador de filas z/OS . Um cliente gerenciado. Net tentando conectar a um gerenciador de filas do z/OS falha com um erro, MQRC_UOW_ENLISTMENT_ERROR (mqrc=2354), na chamada MQOPEN. Entretanto, um aplicativo .NET gerenciado usando transações XA funcionará com o gerenciador de filas distribuído.

Definindo qual Tipo de Conexão Usar

O tipo de conexão é determinado pela configuração do nome de conexão, nome de canal, do valor de customização NMQ_MQ_LIB e da propriedade MQC.TRANSPORT_PROPERTY.

É possível especificar o nome de conexão conforme a seguir:

- Explicitamente em um construtor MQQueueManager:

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- Configurando as propriedades MQC.HOST_NAME_PROPERTY e, opcionalmente, MQC.PORT_PROPERTY em uma entrada hashtable em um construtor MQQueueManager:

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Como valores de MQEnvironment explícitos

```
MQEnvironment.Hostname
```

`MQEnvironment.Port`(opcional).

- Configurando as propriedades MQC.HOST_NAME_PROPERTY e, opcionalmente, MQC.PORT_PROPERTY na hashtable MQEnvironment.properties.

É possível especificar o nome do canal conforme a seguir:

- Explicitamente em um construtor MQQueueManager:

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- Configurando a propriedade MQC.CHANNEL_PROPERTY em uma entrada de hashtable em um construtor MQQueueManager:

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Como um valor de MQEnvironment explícito

```
MQEnvironment.Channel
```

- Configurando a propriedade MQC.CHANNEL_PROPERTY na hashtable MQEnvironment.properties.

É possível especificar a propriedade de transporte conforme a seguir:

- Configurando a propriedade MQC.TRANSPORT_PROPERTY em uma entrada de hashtable em um construtor MQQueueManager:

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Configurando a propriedade MQC.TRANSPORT_PROPERTY na hashtable MQEnvironment.properties.

Selecione o tipo de conexão que você requer usando um dos seguintes valores:

MQC.TRANSPORT_MQSERIES_BINDINGS - conectar como servidor
MQC.TRANSPORT_MQSERIES_CLIENT - conectar como cliente não XA
MQC.TRANSPORT_MQSERIES_XACLIENT - conectar como cliente XA
MQC.TRANSPORT_MQSERIES_MANAGED - conectar como cliente gerenciado não XA

É possível configurar o valor de customização NMQ_MQ_LIB para escolher explicitamente o tipo de conexão conforme mostrado na tabela a seguir

Valor de NMQ_MQ_LIB	**Tipo de conexão**
mqic.dll	Conectar como um cliente não XA
mqicxa.dll	Conectar como um cliente XA
mqm.dll	Conectar como um servidor ou como um cliente não XA
managed	Conectar como um cliente gerenciado não XA
Nota: Valores de mqic32.dll e mqic32xa.dll são aceitos como sinônimos de mqic.dll e mqicxa.dll para compatibilidade com liberações anteriores. No entanto, mqm.dll e mqm.pdb são apenas parte do pacote do cliente da versão 7.1 em diante.	

Se você escolher um tipo de conexão indisponível em seu ambiente, por exemplo, você especificar mqic32xa.dll e não tiver suporte XA, WebSphere MQ .NET lançará uma exceção.

Configurar NMQ_MQ_LIB como "gerenciado" faz com que o cliente use testes de diagnóstico de problemas gerenciados do WebSphere MQ , conversão de dados .NET e outras funções gerenciadas de baixo nível do WebSphere MQ .

Todos os outros valores para NMQ_MQ_LIB fazem com que o processo .NET use testes de diagnóstico de problemas e conversão de dados não gerenciados do WebSphere MQ e outras funções de baixo nível não gerenciadas do WebSphere MQ (assumindo que um cliente ou servidor MQI do WebSphere MQ esteja instalado no sistema).

WebSphere MQ .NET escolhe o tipo de conexão conforme a seguir:

1. Se MQC.TRANSPORT_PROPERTY for especificado, ele conecta-se de acordo com o valor de MQC.TRANSPORT_PROPERTY.

   Observe, entretanto, que configurar MQC.TRANSPORT_PROPERTY com MQC.TRANSPORT_MQSERIES_MANAGED não garante que o processo do cliente seja executado gerenciado. Mesmo com esta configuração, o cliente não é gerenciado nos seguintes casos:

   - Se outro encadeamento no processo tiver se conectado com MQC.TRANSPORT_PROPERTY configurada como algo diferente de MQC.TRANSPORT_MQSERIES_MANAGED.
   - Se NMQ_MQ_LIB não estiver configurado como "gerenciado", os testes de diagnóstico de problemas, a conversão de dados e outras funções de baixo nível não serão totalmente gerenciados (assumindo que um cliente ou servidor MQI do WebSphere MQ esteja instalado no sistema).
2. Se um nome de conexão tiver sido especificado sem um nome de canal, ou um nome de canal tiver sido especificado sem um nome de conexão, ele lançará um erro.
3. Se um nome de conexão e um nome de canal tiverem sido especificados:
   - Se NMQ_MQ_LIB for configurado como mqic32xa.dll, ele se conectará como um cliente XA.
   - Se NMQ_MQ_LIB for configurado como gerenciado, ele se conectará como um cliente gerenciado.
   - Caso contrário, ele se conectará como um cliente não XA.
4. Se NMQ_MQ_LIB for especificado, ele se conectará de acordo com o valor de NMQ_MQ_LIB.
5. Se um servidor WebSphere MQ estiver instalado, ele se conectará como um servidor.
6. Se um cliente MQI do WebSphere MQ estiver instalado, ele se conectará como um cliente não XA
7. Caso contrário, ele se conectará como um cliente gerenciado.

Arquivos de Configuração para Classes do WebSphere MQ para .NET

Um aplicativo cliente .NET pode usar um arquivo de configuração do cliente MQI do WebSphere MQ e se você estiver usando o tipo de conexão gerenciada, um arquivo de configuração do aplicativo .NET. As configurações no arquivo de configuração de aplicativo têm prioridade.

Arquivo de Configuração do Cliente

Uma classe WebSphere MQ para o aplicativo cliente .NET pode usar um arquivo de configuração do cliente da mesma maneira que qualquer outro cliente MQI do WebSphere MQ . Este arquivo geralmente é chamado de mqclient.ini, mas é possível especificar um nome do arquivo diferente. Para obter mais informações sobre o arquivo de configuração do cliente, consulte Configurando um cliente usando um arquivo de configuração WebSphere MQ arquivo de configuração do cliente MQI

Apenas os atributos a seguir em um arquivo de configuração do cliente MQI do WebSphere MQ são relevantes para as classes WebSphere MQ para .NET. Se você especificar outros atributos, isto não terá efeito.

Sub-rotina	**Atribuir**
CHANNELS	CCSID
CHANNELS	ChannelDefinitionDirectory
CHANNELS	ChannelDefinitionFile

Sub-rotina	Atribuir
CHANNELS	ServerConnectionParms
ClientExitPath	Caminho Padrão das Saídas
ClientExitPath	ExitsDefaultPath64
MessageBuffer	MaximumSize
MessageBuffer	PurgeTime
MessageBuffer	UpdatePercentage
TCP	ClntRcvBufSize
TCP	ClntSndBufSize
TCP	IPAddressVersion
TCP	KeepAlive

É possível substituir qualquer um destes atributos usando a variável de ambiente apropriada.

Arquivo de Configuração de Aplicativo

Se você estiver executando com o tipo de conexão gerenciada, também poderá substituir o arquivo de configuração do cliente WebSphere MQ e as variáveis de ambiente equivalentes usando o arquivo de configuração do aplicativo .NET.

As definições do arquivo de configuração do aplicativo .NET são atuadas somente ao executar com o tipo de conexão gerenciado e são ignoradas para outros tipos de conexão.

O arquivo de configuração do aplicativo .NET e seu formato são definidos pela Microsoft para uso geral na estrutura .NET, mas os nomes de seção, as chaves e os valores específicos mencionados nesta documentação são específicos do Websphere MQ.

O formato do arquivo de configuração de aplicativo .NET é um número de *seções*. Cada seção contém uma ou mais *chaves* e cada chave tem um *valor* associado. O exemplo a seguir mostra as seções, chaves e valores usados em um arquivo de configuração de aplicativo .NET para controlar a propriedade TCP/IP KeepAlive:

```
<configuration>
  <configSections>
    <section name="TCP" type="System.Configuration.NameValueSectionHandler"/>
  </configSections>
  <TCP>
    <add key="KeepAlive" value="true"></add>
  </TCP>
<configuration>
```

As palavras-chave usadas nos nomes e chaves de seção do arquivo de configuração de aplicativo .NET correspondem exatamente às palavras-chave para as Sub-rotinas e os Atributos definidos no arquivo de configuração do cliente.

Consulte sua documentação da Microsoft para obter informações adicionais.

Fragmento de Código de Exemplo

O fragmento de código C# a seguir demonstra um aplicativo que executa três ações:

1. Conecta-se a um gerenciador de filas
2. Coloca uma mensagem em SYSTEM.DEFAULT.LOCAL.QUEUE
3. Obtém a mensagem de volta

Ele também mostra como alterar o tipo de conexão.

```
// ============================================================================
// Licensed Materials - Property of IBM
// 5724-H72
// (c) Copyright IBM Corp. 2003, 2024
// ============================================================================
using System;
using System.Collections;

using IBM.WMQ;

class MQSample
{
  // The type of connection to use, this can be:-
  // MQC.TRANSPORT_MQSERIES_BINDINGS for a server connection.
  // MQC.TRANSPORT_MQSERIES_CLIENT for a non-XA client connection
  // MQC.TRANSPORT_MQSERIES_XACLIENT for an XA client connection
  // MQC.TRANSPORT_MQSERIES_MANAGED for a managed client connection
  const String connectionType = MQC.TRANSPORT_MQSERIES_CLIENT;

  // Define the name of the queue manager to use (applies to all connections)
  const String qManager = "your_Q_manager";

  // Define the name of your host connection (applies to client connections only)
  const String hostName = "your_hostname";

  // Define the name of the channel to use (applies to client connections only)
  const String channel = "your_channelname";



  /// <summary>
  /// Initialise the connection properties for the connection type requested
  /// </summary>
  /// <param name="connectionType">One of the MQC.TRANSPORT_MQSERIES_ values</param>
  static Hashtable init(String connectionType)
  {
    Hashtable connectionProperties = new Hashtable();

    // Add the connection type
    connectionProperties.Add(MQC.TRANSPORT_PROPERTY, connectionType);

    // Set up the rest of the connection properties, based on the
    // connection type requested
    switch(connectionType)
    {
      case MQC.TRANSPORT_MQSERIES_BINDINGS:
        break;
      case MQC.TRANSPORT_MQSERIES_CLIENT:
      case MQC.TRANSPORT_MQSERIES_XACLIENT:
      case MQC.TRANSPORT_MQSERIES_MANAGED:
        connectionProperties.Add(MQC.HOST_NAME_PROPERTY, hostName);
        connectionProperties.Add(MQC.CHANNEL_PROPERTY, channel);
        break;
    }

    return connectionProperties;
  }
  /// <summary>
  /// The main entry point for the application.
  /// </summary>
  [STAThread]
  static int Main(string[] args)
  {
    try
    {
      Hashtable connectionProperties = init(connectionType);

      // Create a connection to the queue manager using the connection
      // properties just defined
      MQQueueManager qMgr = new MQQueueManager(qManager, connectionProperties);

      // Set up the options on the queue we want to open
      int openOptions = MQC.MQOO_INPUT_AS_Q_DEF | MQC.MQOO_OUTPUT;

      // Now specify the queue that we want to open,and the open options
      MQQueue system_default_local_queue =
        qMgr.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE", openOptions);
```

```
      // Define a WebSphere MQ message, writing some text in UTF format
      MQMessage hello_world = new MQMessage();
      hello_world.WriteUTF("Hello World!");

      // Specify the message options
      MQPutMessageOptions pmo = new MQPutMessageOptions(); // accept the defaults,
                                                           // same as MQPMO_DEFAULT

      // Put the message on the queue
      system_default_local_queue.Put(hello_world, pmo);



      // Get the message back again

      // First define a WebSphere MQ message buffer to receive the message
      MQMessage retrievedMessage =new MQMessage();
      retrievedMessage.MessageId =hello_world.MessageId;

      // Set the get message options
      MQGetMessageOptions gmo =new MQGetMessageOptions(); //accept the defaults
                                                          //same as MQGMO_DEFAULT

      // Get the message off the queue
      system_default_local_queue.Get(retrievedMessage,gmo);

      // Prove we have the message by displaying the UTF message text
      String msgText = retrievedMessage.ReadUTF();
      Console.WriteLine("The message is: {0}", msgText);

      // Close the queue
      system_default_local_queue.Close();

      // Disconnect from the queue manager
      qMgr.Disconnect();
    }

    //If an error has occurred,try to identify what went wrong.

    //Was it a WebSphere MQ error?
    catch (MQException ex)
    {
      Console.WriteLine("A WebSphere MQ error occurred: {0}", ex.ToString());
    }

    catch (System.Exception ex)
    {
      Console.WriteLine("A System error occurred: {0}", ex.ToString());
    }

    return 0;
  }//end of start
}//end of sample
```

Operações nos Gerenciadores de Filas

Esta seção descreve como conectar-se a, e desconectar-se de, um gerenciador de filas usando classes do WebSphere MQ para .NET.

Configurando o Ambiente do WebSphere MQ

Antes de usar a conexão do cliente para se conectar a um gerenciador de filas, é necessário configurar o ambiente do WebSphere MQ.

Nota: Esta etapa não é necessária ao usar classes do WebSphere MQ para .NET no modo de ligações do servidor.

A interface de programação .NET permite que você use o valor de customização NMQ_MQ_LIB mas também inclui uma classe MQEnvironment. Esta classe permite que você especifique detalhes que devem ser usados durante a tentativa de conexão, tal como os itens na lista a seguir:

- Nome do canal
- Nome do host

- Número da Porta
- Saídas do canal
- Parâmetros de SSL
- ID do Usuário e Senha

Para obter informações completas sobre a classe MQEnvironment, consulte MQEnvironment Classe .NET

Para especificar o nome do canal e o nome do host, use o código a seguir:

```
MQEnvironment.Hostname = "host.domain.com";
MQEnvironment.Channel  = "client.channel";
```

Por padrão, os clientes tentam se conectar a um listener do WebSphere MQ na porta 1414. Para especificar uma porta diferente, use o código:

```
MQEnvironment.Port = nnnn;
```

Conectando-se a um gerenciador de filas

Agora você está pronto para conectar-se a um gerenciador de filas criando uma nova instância da classe MQQueueManager:

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Para desconectar-se de um gerenciador de filas, chame o método `Disconnect` no gerenciador de filas:

```
queueManager.Disconnect();
```

Você deve ter autoridade de consulta (`inq`) no gerenciador de filas ao tentar se conectar ao gerenciador de filas. Sem autoridade de consulta, a tentativa de conexão falhará.

Se você chamar o método `Disconnect`, todas as filas e processos abertos que você acessou usando esse gerenciador de filas serão encerrados. Entretanto, é uma boa prática de programação fechar estes recursos explicitamente quando você terminar de usá-los. Para fechar os recursos, use o método `Close` no objeto associado a cada recurso.

Os métodos `Commit` e `Backout` em um gerenciador de filas substituem as chamadas MQCMIT e MQBACK que são usadas com a interface processual.

Acessando Filas e Tópicos

É possível acessar filas e tópicos usando métodos de MQQueueManager ou construtores apropriados.

Para acessar filas, use os métodos da classe MQQueueManager. O MQOD (estrutura do descritor de objeto) é reduzido nos parâmetros destes métodos. Por exemplo, para abrir uma fila em um gerenciador de filas representado por um objeto MQQueueManager chamado queueManager, use o seguinte código:

```
MQQueue queue = queueManager.AccessQueue("qName",
                                         MQC.MQOO_OUTPUT,
                                         "qMgrName",
                                         "dynamicQName",
                                         "altUserId");
```

O parâmetro *options* é o mesmo que o parâmetro Options na chamada MQOPEN.

O método AccessQueue retorna um novo objeto da classe MQQueue.

Quando você tiver concluído o uso da fila, use o método Close() para fechá-la, como no exemplo a seguir:

```
queue.Close();
```

Com o WebSphere MQ .NET, também é possível criar uma fila usando o construtor MQQueue. Os parâmetros são exatamente os mesmos que para o método accessQueue, com a adição de um parâmetro do gerenciador de filas especificando o objeto MQQueueManager instanciado para uso. Por exemplo:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            MQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserId");
```

Construir um objeto de fila desta maneira permite que você grave suas próprias subclasses de MQQueue.

De modo semelhante, também é possível acessar tópicos usando os métodos da classe MQQueueManager. Use um método AccessTopic() para abrir um tópico. Isto retorna um novo objeto da classe MQTopic. Quando você tiver concluído o uso do tópico, use o método Close() de MQTopic para fechá-lo.

Também é possível criar um tópico usando um construtor MQTopic. Há vários construtores para tópicos; para obter mais informações, consulte MQTopic .NET class.

Manipulando Mensagens

As mensagens são tratadas usando os métodos das classes de fila ou tópico. Para criar uma nova mensagem, crie um novo MQMessageobject.

Coloque mensagens nas filas ou tópicos usando o método Put() da classe MQQueue ou MQTopic. Obtenha mensagens das filas ou tópicos usando o método Get() da classe MQQueue ou MQTopic. Diferente da interface processual, em que MQPUT e MQGET colocam e obtêm matrizes de bytes, as classes do WebSphere MQ para .NET colocam e obtêm instâncias da classe MQMessage. A classe MQMessage encapsula o buffer de dados que contém os dados da mensagem reais, junto com todos os parâmetros MQMD (descritor de mensagens) que descrevem essa mensagem.

Para criar uma nova mensagem, crie uma nova instância da classe MQMessage e use os métodos WriteXXX para colocar dados no buffer de mensagem.

Quando a nova instância de mensagem for criada, todos os parâmetros MQMD serão configurados automaticamente para seus valores padrão, conforme definido em Valores iniciais e declarações de idioma para MQMD . O método Put() de MQQueue também utiliza uma instância da classe MQPutMessageOptions como um parâmetro. Esta classe representa a estrutura MQPMO. O exemplo a seguir cria uma mensagem e a coloca em uma fila:

```
// Build a new message containing my age followed by my name
MQMessage myMessage = new MQMessage();
myMessage.WriteInt(25);

String name = "Charlie Jordan";
myMessage.WriteUTF(name);

// Use the default put message options...
MQPutMessageOptions pmo = new MQPutMessageOptions();

// put the message!
queue.Put(myMessage,pmo);
```

O método Get() de MQQueue retorna uma nova instância de MQMessage, que representa a mensagem recém-obtida da fila. Ele também utiliza uma instância da classe MQGetMessageOptions como um parâmetro. Esta classe representa a estrutura de MQGMO.

Não é necessário especificar um tamanho de mensagem máximo, porque o método Get() ajusta automaticamente o tamanho de seu buffer interno para ajustar a mensagem recebida. Use os métodos ReadXXX da classe MQMessage para acessar os dados na mensagem retornada.

O exemplo a seguir mostra como obter uma mensagem de uma fila:

```
// Get a message from the queue
MQMessage theMessage    = new MQMessage();
MQGetMessageOptions gmo = new MQGetMessageOptions();
queue.Get(theMessage,gmo);  // has default values

// Extract the message data
int age = theMessage.ReadInt();
String name1 = theMessage.ReadUTF();
```

É possível alterar o formato numérico que os métodos de leitura e gravação usam configurando a variável de membro *encoding*.

É possível alterar o conjunto de caracteres para usar para ler e gravar sequências configurando a variável de membro *characterSet*.

Consulte MQMessage .NET class para obter mais detalhes

Nota: O método WriteUTF() de MQMessage codifica automaticamente o comprimento da sequência bem como os bytes Unicode que ela contém. Quando sua mensagem for lida por outro programa .NET (usando ReadUTF()), está é a maneira mais simples de enviar informações de cadeia.

Manipulando Propriedades de Mensagem

As propriedades de mensagem permitem que você selecione mensagens ou recupere informações sobre uma mensagem sem acessar seus cabeçalhos. A classe MQMessage contém métodos para obter e configurar propriedades.

É possível usar propriedades de mensagem para permitir que um aplicativo selecione mensagens para processar ou recupere informações sobre uma mensagem sem acessar cabeçalhos MQMD ou MQRFH2. Eles também facilitam a comunicação entre WebSphere MQ e aplicativos JMS. Para obter mais informações sobre as propriedades de mensagem no WebSphere MQ, consulte Propriedades de mensagens

A classe MQMessage fornece vários métodos para obter e configurar propriedades, de acordo com o tipo de dado da propriedade. Os métodos get possuem nomes no formato Get*Property e os métodos set possuem nomes no formato Set*Property, em que o asterisco (*) representa uma das sequências a seguir:

- Booleana
- Byte
- bytes
- Double
- Float
- Int
- Int2
- Int4
- Int8
- Long
- Object
- Curta
- Sequência

Por exemplo, para obter a propriedade myproperty do WebSphere MQ (uma sequência de caracteres), use a chamada `message.GetStringProperty('myproperty')`. Opcionalmente, é possível transmitir um descritor de propriedades, que WebSphere MQ concluirá.

Manipulando Erros

Manipule erros que surgem das classes do WebSphere MQ para .NET usando blocos `try` e `catch`.

Métodos na interface .NET não retornam um código de conclusão e código de razão. Em vez disso, eles lançam uma exceção sempre que o código de conclusão e código de razão resultantes de um chamada do WebSphere MQ não são ambos zero. Isto simplifica a lógica do programa de forma que não seja necessário verificar os códigos de retorno após cada chamada para o WebSphere MQ. É possível decidir em quais pontos em seu programa você deseja lidar com a possibilidade de falha. Nestes pontos, é possível cercar seu código com blocos `try` e `catch`, como no exemplo a seguir:

```
try
{
  myQueue.Put(messageA,PutMessageOptionsA);
  myQueue.Put(messageB,PutMessageOptionsB);
}
catch (MQException ex)
{
  // This block of code is only executed if one of
  // the two put methods gave rise to a non-zero
  // completion code or reason code.
  Console.WriteLine("An error occurred during the put operation:" +
                        "CC = " + ex.CompletionCode +
                        "RC = " + ex.ReasonCode);
  Console.WriteLine("Cause exception:" + ex );
}
```

Obtendo e Configurando Valores de Atributos

As classes MQManagedObject, MQQueue, and MQQueueManager contêm métodos que permitem que você obtenha e configure seus valores de atributos. Observe que, para MQQueue, os métodos funcionam somente se você especificar a consulta apropriada e configurar sinalizadores quando abrir a fila.

Para obter atributos comuns, as classes MQQueueManager e MQQueue herdam de uma classe chamada MQManagedObject. Esta classe define as interfaces Inquire() e Set().

Quando você cria um novo objeto do gerenciador de filas usando o operador *new*, ele é aberto automaticamente para consulta. Quando você usa o método AccessQueue() para acessar um objeto de fila, esse objeto *não* é automaticamente aberto para operações de consulta ou configuração, isto poderia causar problemas com alguns tipos de filas remotas. Para usar os métodos Inquire e Set e para configurar propriedades em uma fila, é necessário especificar os sinalizadores de consulta e configuração apropriados no parâmetro openOptions do método AccessQueue().

Os métodos inquire e set utilizam três parâmetros:

- matriz de seletores
- matriz intAttrs
- matriz charAttrs

Não são necessários os parâmetros SelectorCount, IntAttrCount e CharAttrLength, que estão localizados em MQINQ, porque o comprimento de uma matriz é sempre conhecido. O exemplo a seguir mostra como fazer uma consulta em uma fila:

```
//inquire on a queue
int [ ] selectors = new int [2] ;
int [ ] intAttrs = new int [1] ;
byte [ ] charAttrs = new byte [MQC.MQ_Q_DESC_LENGTH];
selectors [0] = MQC.MQIA_DEF_PRIORITY;
selectors [1] = MQC.MQCA_Q_DESC;
queue.Inquire(selectors,intAttrs,charAttrs);
ASCIIEncoding enc = new ASCIIEncoding();
String s1 = "";
s1 = enc.GetString(charAttrs);
```

Todos os atributos destes objetos podem ser consultados. Um subconjunto de atributos é exposto como as propriedades de um objeto. Para obter uma lista dos atributos do objeto, veja Atributos de objetos. Para propriedades de objetos, consulte a descrição da classe apropriada.

Programas Multiencadeados

O ambiente de tempo de execução .NET é inerentemente multiencadeado. As classes do WebSphere MQ para .NET permitem que um objeto do gerenciador de filas seja compartilhado em múltiplos encadeamentos, mas asseguram que todo acesso ao gerenciador de filas de destino seja sincronizado.

Considere um programa simples que se conecta a um gerenciador de filas e abre uma fila na inicialização. O programa exibe um único botão na tela. Quando um usuário clica nesse botão, o programa busca uma mensagem da fila. Nesta situação, a inicialização do aplicativo ocorre em um encadeamento e o código que é executado em resposta ao pressionamento do botão é executado em um encadeamento separado (o encadeamento da interface com o usuário).

A implementação do WebSphere MQ .NET assegura que, para uma determinada conexão (instância do objetoMQQueueManager), todo o acesso ao gerenciador de filas WebSphere MQ de destino seja sincronizado. O comportamento padrão é que um encadeamento que deseja emitir uma chamada a um gerenciador de filas seja bloqueado até todas as outras chamadas em andamento para essa conexão sejam concluídas. Se você requerer acesso simultâneo ao mesmo gerenciador de filas a partir de múltiplos encadeamentos em seu programa, crie um novo objeto MQQueueManager para cada encadeamento que requer acesso simultâneo. (Isto é equivalente a emitir uma chamada MQCONN separada para cada encadeamento.)

Se as opções de conexão padrão forem substituídas por MQC.MQCNO_HANDLE_SHARE_NONE ou MQC.MQCNO_SHARE_NO_BLOCK, o gerenciador de filas não será mais sincronizado.

Usando uma tabela de definição de canal do cliente com .NET

É possível usar uma tabela de definição de canal do cliente (CCDT) com as classes .NET para o WebSphere MQ. Você especifica o local da CCDT de diferentes maneiras, dependendo de você estar usando uma conexão gerenciada ou uma conexão não gerenciada.

Tipo de conexão do cliente não gerenciada XA ou não XA

Com um tipo de conexão não gerenciado, é possível especificar o local da CCDT de duas maneiras:

- Usando as variáveis de ambiente MQCHLLIB para especificar o diretório no qual a tabela está localizada e MQCHLTAB para especificar o nome do arquivo da tabela.
- Usando o arquivo de configuração do cliente. Na sub-rotina CHANNELS, use os atributos ChannelDefinitionDirectory para especificar o diretório no qual a tabela está localizada e ChannelDefinitionFile para especificar o nome do arquivo.

Se o local for especificado das duas maneiras, no arquivo de configuração do cliente e usando variáveis de ambiente, as variáveis de ambiente terão prioridade. É possível usar este recurso para especificar um local padrão no arquivo de configuração do cliente e substituí-lo usando variáveis de ambiente quando necessário.

Tipo de conexão do cliente gerenciado

Com um tipo de conexão gerenciado, é possível especificar o local da CCDT de três maneiras:

- Usando o arquivo de configuração de aplicativo .NET. Na seção CHANNELS, use os ChannelDefinitionDirectory chaves para especificar o diretório no qual a tabela está localizada e ChannelDefinitionFile para especificar o nome do arquivo.
- Usando as variáveis de ambiente MQCHLLIB para especificar o diretório no qual a tabela está localizada e MQCHLTAB para especificar o nome do arquivo da tabela.
- Usando o arquivo de configuração do cliente. Na sub-rotina CHANNELS, use os atributos ChannelDefinitionDirectory para especificar o diretório no qual a tabela está localizada e ChannelDefinitionFile para especificar o nome do arquivo.

Se o local for especificado de mais de uma dessas maneiras, as variáveis de ambiente terão prioridade sobre o arquivo de configuração do cliente e o Arquivo de Configuração de Aplicativo .NET terá prioridade sobre os outros dois métodos. É possível usar este recurso para especificar um local padrão no arquivo

de configuração do cliente e substituí-lo usando variáveis de ambiente ou o arquivo de configuração de aplicativo quando necessário.

Como um Aplicativo .NET Determina qual Definição de Canal Usar

No ambiente do cliente .NET do WebSphere MQ, a definição de canal a ser usada pode ser especificada de várias maneiras diferentes. Podem existir várias especificações da definição de canal. Um aplicativo deriva a definição de canal de uma ou mais origens.

Se existir mais de uma definição de canal, aquela usada será selecionada na seguinte ordem de prioridade:

1. Propriedades especificadas no construtor MQQueueManager, seja explicitamente ou incluindo *MQC.CHANNEL_PROPERTY* na hashtable de propriedades
2. Uma propriedade *MQC.CHANNEL_PROPERTY* na hashtable MQEnvironment.properties
3. A propriedade *Channel* em MQEnvironment
4. O arquivo de configuração de aplicativo .NET, nome de seção CHANNELS, chave ServerConnectionParms (aplica-se apenas a conexões gerenciadas)
5. A variável de ambiente *MQSERVER*
6. O arquivo de configuração do cliente, sub-rotina CHANNELS, Atributo ServerConnectionParms
7. A Client Channel Definition Table (CCDT). O local da CCDT é especificado no arquivo de configuração de aplicativo .NET (aplica-se apenas a conexões gerenciadas)
8. A Client Channel Definition Table (CCDT). O local da CCDT é especificado usando as variáveis de ambiente *MQCHLIB* e *MQCHLTAB*
9. A Client Channel Definition Table (CCDT). O local da CCDT é e especificado usando o arquivo de configuração do cliente

Para os itens 1-3, a definição de canal é construída campo por campo a partir de valores fornecidos pelo aplicativo. Esses valores podem ser fornecidos usando diferentes interfaces e podem existir vários valores para cada uma. Os valores do campo são incluídos na definição de canal seguindo a ordem de prioridade fornecida:

1. O valor de *connName* no construtor MQQueueManager
2. Valores de propriedades da hashtable MQQueueManager.properties
3. Valores de propriedades da hashtable MQEnvironment.properties
4. Valores configurados como campos MQEnvironment (por exemplo, MQEnvironment.Hostname, MQEnvironment.Port)

Para os itens 4-6, a definição de canal inteira é fornecida como o valor. Campos não especificados na definição de canal assumem os padrões do sistema. Nenhum valor de outros métodos de definição de canais e seus campos é fundido com estas especificações.

Para os itens 7-9, a definição de canal inteira é tomada a partir da CCDT. Os campos que não foram explicitamente especificados quando o canal foi definido tomam os padrões do sistema. Nenhum valor de outros métodos de definição de canais e seus campos é fundido com estas especificações.

Usando saídas do canal no IBM WebSphere MQ .NET

Se você usar ligações do cliente, poderá usar saídas do canal como para qualquer outra conexão do cliente. Se você usar ligações gerenciadas, deverá gravar um programa de saída que implementa uma interface apropriada.

Ligações do Cliente

Se você usar ligações do cliente, poderá usar saídas do canal conforme descrito em Saídas do canal. Não é possível usar saídas do canal gravadas para ligações gerenciadas.

Ligações Gerenciadas

Se você usar uma conexão gerenciada, para implementar uma saída, defina uma nova classe .NET que implemente a interface apropriada. Três interfaces de saída são definidas no pacote do WebSphere MQ:

- MQSendExit
- MQReceiveExit
- MQSecurityExit

Nota: As saídas de usuário gravadas usando estas interfaces não são suportadas como saídas do canal no ambiente não gerenciado.

A amostra a seguir define uma classe que implementa todas as três:

```
class MyMQExits : MQSendExit, MQReceiveExit, MQSecurityExit
{
   // This method comes from the send exit
  byte[] SendExit(MQChannelExit       channelExitParms,
                  MQChannelDefinition channelDefinition,
                  byte[]              dataBuffer
                  ref int             dataOffset
                  ref int             dataLength
                  ref int             dataMaxLength)
  {
     // complete the body of the send exit here
  }


  // This method comes from the receive exit
  byte[] ReceiveExit(MQChannelExit       channelExitParms,
                     MQChannelDefinition channelDefinition,
                     byte[]              dataBuffer
                     ref int             dataOffset
                     ref int             dataLength
                     ref int             dataMaxLength)
  {
     // complete the body of the receive exit here
  }


  // This method comes from the security exit
  byte[] SecurityExit(MQChannelExit       channelExitParms,
                      MQChannelDefinition channelDefParms,
                      byte[]              dataBuffer
                      ref int             dataOffset
                      ref int             dataLength
                      ref int             dataMaxLength)
  {
     // complete the body of the security exit here
  }

}
```

Cada saída passa por uma instância de objeto MQChannelExit e MQChannelDefinition. Estes objetos representam as estruturas MQCXP e MQCD definidas na interface processual.

Os dados a serem enviados por uma saída de envio e os dados recebidos em uma saída de segurança ou recebimento são especificados usando os parâmetros de saída.

Na entrada, os dados no deslocamento *dataOffset* com comprimento *dataLength* na matriz de byte *dataBuffer* são os dados que estão prestes a serem enviados por uma saída de envio e os dados recebidos em uma saída de segurança ou recebimento. O parâmetro *dataMaxLength* fornece o comprimento máximo (de *dataOffset*) disponível para a saída em *dataBuffer*. Nota: Para uma saída de segurança, é possível que o dataBuffer seja nulo se esta for a primeira vez que a saída é chamada ou se a extremidade do parceiro elegeu para não enviar dados.

No retorno, o valor de *dataOffset* e *dataLength* deve ser configurado para apontar para o deslocamento e comprimento na matriz de byte retornada que as classes .NET devem estão usar. Para uma saída de envio, isto indica os dados que ela deve enviar e para uma saída de segurança ou recebimento, os dados que devem ser interpretados. A saída normalmente deve retornar uma matriz de byte; exceções são uma saída de segurança que poderia escolher não enviar dados e qualquer saída chamada com as razões INIT

ou TERM. A forma mais simples de saída que pode ser gravada, portanto, é uma que não faz nada mais que retornar dataBuffer:

O corpo de saída mais simples possível é:

```
{
  return dataBuffer;
}
```

Classe MQChannelDefinition

V 7.5.0.6 No Version 7.5.0, Fix Pack 6, o ID do usuário e a senha que são especificados com o aplicativo cliente .NET gerenciado são configurados na classe MQChannelDefinition do IBM WebSphere MQ .NET que é transmitida para a saída de segurança do cliente. A saída de segurança copia o ID do usuário e a senha nos campos MQCD.RemoteUserIdentifier e MQCD.RemotePassword (veja "Escrevendo uma saída de segurança" na página 412).

Especificando Saídas do Canal (Cliente Gerenciado)

Se você especificar um nome de canal e nome de conexão ao criar seu objeto MQQueueManager (no construtor MQEnvironment ou MQQueueManager) será possível especificar saídas do canal de duas maneiras.

Na ordem de precedência, eles são:

1. Transmitir propriedades hashtable MQC.SECURITY_EXIT_PROPERTY, MQC.SEND_EXIT_PROPERTY ou MQC.RECEIVE_EXIT_PROPERTY no construtor MQQueueManager.
2. Configurar as propriedades MQEnvironment SecurityExit, SendExit ou ReceiveExit.

Se você não especificar um nome de canal e nome de conexão, as saídas do canal a usar vêm da definição de canal selecionada a partir de uma tabela de client channel definition table (CCDT). Não é possível substituir os valores armazenados na definição de canal. Veja Client Channel Definition Table e "Usando uma tabela de definição de canal do cliente com .NET" na página 593 para obter informações adicionais sobre tabelas de definição de canal.

Em cada caso, a especificação tem a forma de uma sequência com o formato a seguir:

```
Assembly_name(Class_name)
```

Class_name é o nome completo, incluindo a especificação de espaço de nomes, de uma classe .NET que implementa a interface IBM.WMQ.MQSecurityExit, IBM.WMQ.MQSendExit ou IBM.WMQ.MQReceiveExit (conforme apropriado). *Assembly_name* é o local completo, incluindo extensão de arquivo, da montagem que hospeda a classe. O comprimento da sequência é limitado a 999 caracteres se você usar as propriedades de MQEnvironment ou MQQueueManager. Entretanto, se o nome de saída do canal for especificado no CCDT, ele será limitado a 128 caracteres. Quando necessário, o código do cliente .NET carrega e cria uma instância da classe especificada analisando a especificação de cadeia.

Especificando Dados do Usuário de Saída do Canal (Cliente Gerenciado)

As saídas do canal podem ter dados do usuário associados a elas. Se você especificar um nome de canal e nome de conexão ao criar seu objeto MQQueueManager (no construtor MQEnvironment ou MQQueueManager), poderá especificar os dados do usuário de duas maneiras.

Na ordem de precedência, eles são:

1. Transmitir propriedades hashtable MQC.SECURITY_USERDATA_PROPERTY, MQC.SEND_USERDATA_PROPERTY ou MQC.RECEIVE_USERDATA_PROPERTY no construtor MQQueueManager.
2. Configurar as propriedades MQEnvironment SecurityUserData, SendUserData ou ReceiveUserData.

Se você não especificar um nome de canal e um nome de conexão, os valores dos dados do usuário de saída para usar virão da definição de canal selecionada na client channel definition table (CCDT). Não é

possível substituir os valores armazenados na definição de canal. Veja Client Channel Definition Table e “Usando uma tabela de definição de canal do cliente com .NET” na página 593 para obter informações adicionais sobre tabelas de definição de canal.

Em cada caso, a especificação é uma sequência, limitada a 32 caracteres.

Reconexão automática do cliente no .NET

É possível fazer com que seu cliente se reconecte automaticamente a um gerenciador de filas durante uma quebra de conexão inesperada.

Um cliente pode se desconectar inesperadamente a partir de um gerenciador de filas se, por exemplo, o gerenciador de filas parar ou se a rede ou o servidor falhar.

Sem a reconexão automática do cliente, um erro será produzido quando a conexão falhar. É possível usar o código de erro para ajudá-lo a restabelecer a conexão.

Um cliente que usa o recurso de reconexão de cliente automático é denominado um cliente reconectável. Para criar um cliente reconectável, especifique certas opções denominadas opções de reconexão enquanto se conecta ao gerenciador de filas.

Se o aplicativo cliente for um cliente .NET do WebSphere MQ , ele poderá optar por obter a reconexão automática do cliente especificando um valor apropriado para CONNECT_OPTIONS_PROPERTY ao usar a classe `MQQueueManager` para criar um gerenciador de filas. Consulte Opções de reconexão para obter detalhes dos valores CONNECT_OPTIONS_PROPERTY.

É possível selecionar se o aplicativo cliente sempre se conecta e reconecta a um gerenciador de filas do mesmo nome, com o mesmo gerenciador de filas ou a qualquer conjunto de gerenciadores de filas que foi definido com o mesmo QMNAME na tabela de conexão do cliente (consulte Grupos do gerenciadores de filas na CCDT para obter detalhes).

Suporte a Secure Sockets Layer (SSL)

A seção a seguir não se aplica ao cliente gerenciado.

As classes do WebSphere MQ para aplicativos clientes .NET suportam criptografia Secure Sockets Layer (SSL). SSL fornece criptografia de comunicação, autenticação e integridade da mensagem. É geralmente usada para comunicações seguras entre qualquer dois pontos na Internet ou em uma intranet.

Ativando SSL

SSL é suportado somente para conexões do cliente. Para ativar SSL, é necessário especificar o CipherSpec para uso ao se comunicar com o gerenciador de filas, e isto deve corresponder a CipherSpec configurado no canal de destino.

Para ativar SSL, especifique CipherSpec usando a variável de membro estático SSLCipherSpec de MQEnvironment. O exemplo a seguir conecta-se a um canal SVRCONN denominado SECURE.SVRCONN.CHANNEL, que foi configurado para requerer SSL com um CipherSpec igual a NULL_MD5:

```
MQEnvironment.Hostname         = "your_hostname";
MQEnvironment.Channel          = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.SSLCipherSpec    = "NULL_MD5";
MQEnvironment.SSLKeyRepository = "C:\mqm\key";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

Veja Especificando CipherSpecs para obter uma lista de CipherSpecs.

A propriedade SSLCipherSpec também pode ser configurada usando MQC.SSL_CIPHER_SPEC_PROPERTY na hashtable das propriedades da conexão.

Para conectar-se com êxito usando SSL, o keystore do cliente deve ser configurado com cadeia de certificados raiz de Autoridade de Certificação a partir da qual o certificado apresentado pelo gerenciador de filas pode ser autenticado. Similarmente, se SSLClientAuth no canal SVRCONN tiver sido configurado

como MQSSL_CLIENT_AUTH_REQUIRED, o keystore do cliente deverá conter um certificado pessoal de identificação que seja confiável pelo gerenciador de filas.

Usando o nome distinto do gerenciador de filas

O gerenciador de filas se identifica usando um certificado SSL, que contém um *Nome Distinto* (DN).

Um aplicativo cliente do WebSphere MQ .NET pode usar este DN para assegurar que ele esteja se comunicando com o gerenciador de filas correto. Um padrão de DN é especificado usando a variável sslPeerName de MQEnvironment. Por exemplo, configurar:

```
MQEnvironment.SSLPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

permite que a conexão seja bem-sucedida somente se o gerenciador de filas apresentar um certificado com um Nome Comum iniciando QMGR., e pelo menos dois nomes de Unidade Organizacional, o primeiro dos quais deve ser IBM , e o segundo WEBSPHERE,

A propriedade SSLPeerName também pode ser configurada usando MQC.SSL_PEER_NAME_PROPERTY na hashtable das propriedades da conexão. Para obter mais informações sobre Nomes Distintos e regras para configurar nomes de mesmo nível, consulte Segurança.

Se SSLPeerName estiver configurado, as conexões serão bem-sucedidas somente se ele estiver configurado com um padrão válido e o gerenciador de filas apresentar um certificado correspondente.

Identificador de Erros ao Usar SSL

Os códigos de razão a seguir podem ser emitidos pelas classes do WebSphere MQ para .NET ao conectar-se a um gerenciador de filas usando SSL:

MQRC_SSL_NOT_ALLOWED
A propriedade SSLCipherSpec foi configurada, mas a conexão de ligações foi usada. Somente a conexão do cliente suporta SSL.

MQRC_SSL_PEER_NAME_MISMATCH
O padrão de DN especificado na propriedade SSLPeerName não correspondia ao DN apresentado pelo gerenciador de filas.

MQRC_SSL_PEER_NAME_ERROR
O padrão de DN especificado na propriedade SSLPeerName não era válido.

Usando o .NET Monitor

Consulte Recursos que podem ser usados somente com a instalação primária no Windows para obter informações importantes

O .NET Monitor é um aplicativo semelhante a um monitor acionador do WebSphere MQ. É possível criar componentes .NET que são instanciados sempre que uma mensagem é recebida em uma fila monitorada e, em seguida, processar essa mensagem. O .NET Monitor é iniciado pelo comando `runmqdnm` e parado pelo comando `endmqdnm`. Para obter detalhes desses comandos, veja runmqdnm e endmqdnm.

Para usar o .NET Monitor, grave um componente que implementa a interface IMQObjectTrigger, que está definida em amqmdnm.dll.

Componentes podem ser transacionais ou não transacionais. Um componente transacional deve herdar de System.EnterpriseServices.ServicedComponent e ser registrado como RequiresTransaction ou SupportsTransaction. Ele não deve ser registrado como RequiresNew porque o .NET Monitor já iniciou uma transação.

O componente recebe objetos MQQueueManager, MQQueue e MQMessage de `runmqdnm`. Ele também deve receber uma sequência de Parâmetros do Usuário se uma tiver sido especificada usando a opção da linha de comandos *-u* quando `runmqdnm` foi iniciado. Observe que seu componente recebe o conteúdo de uma mensagem que chegou na fila monitorada em um objeto MQMessage. Ele não precisa se conectar ao gerenciador de filas, abrir a fila ou obter a mensagem em si. O componente deve então processar a mensagem conforme apropriado e retornar o controle ao .NET Monitor.

Se seu componente tiver sido gravado como um componente transacional, ele registrará para confirmar ou retroceder a transação usando os recursos fornecidos pelo System.EnterpriseServices.ServicedComponent.

Como o componente recebe os objetos MQQueueManager e MQQueue bem como a mensagem, ele possui informações de contexto completas para essa mensagem e pode, por exemplo, abrir outra fila no mesmo gerenciador de filas sem precisar conectar-se separadamente ao WebSphere MQ.

Fragmentos de Código de Exemplo

Este tópico contém dois exemplos de componentes que obtêm uma mensagem do .NET Monitor e a imprimem, um usando processamento transacional e o outro processamento não transacional. Um terceiro exemplo mostra rotinas do utilitário comuns, aplicáveis aos dois primeiros exemplos. Todos os exemplos estão em C#.

Exemplo 1: Processamento Transacional

```
/********************************************************************/
/* Licensed materials, property of IBM                              */
/* 63H9336                                                          */
/* (C) Copyright IBM Corp. 2005, 2024.                              */
/********************************************************************/
using System;
using System.EnterpriseServices;

using IBM.WMQ;
using IBM.WMQMonitor;

[assembly: ApplicationName("dnmsamp")]

// build:
//
// csc -target:library -reference:amqmdnet.dll;amqmdnm.dll TranAssembly.cs
//
// run (with dotnet monitor)
//
// runmqdnm -m <QMNAME> -q <QNAME> -a dnmsamp.dll -c Tran

namespace dnmsamp
{
  [TransactionAttribute(TransactionOption.Required)]
  public class Tran : ServicedComponent, IMQObjectTrigger
  {
    Util util = null;

    [AutoComplete(true)]
    public void Execute(MQQueueManager qmgr, MQQueue queue,
        MQMessage message, string param)
    {
      util = new Util("Tran");

      if (param != null)
        util.Print("PARAM: '" +param.ToString() + "'");

      util.PrintMessage(message);

      //System.Console.WriteLine("SETTING ABORT");
      //ContextUtil.MyTransactionVote = TransactionVote.Abort;

      System.Console.WriteLine("SETTING COMMIT");
      ContextUtil.SetComplete();
      //ContextUtil.MyTransactionVote = TransactionVote.Commit;
    }
  }
}
```

Exemplo 2: Processamento Não-transacional

```
/********************************************************************/
/* Licensed materials, property of IBM                              */
/* 63H9336                                                          */
/* (C) Copyright IBM Corp. 2005, 2024.                              */
```

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

using System;

using IBM.WMQ;
using IBM.WMQMonitor;

// build:
//
// csc -target:library -reference:amqmdnet.dll;amqmdnm.dll NonTranAssembly.cs
//
// run (with dotnet monitor)
//
// runmqdnm -m <QMNAME> -q <QNAME> -a dnmsamp.dll -c NonTran
namespace dnmsamp
{
  public class NonTran : IMQObjectTrigger
  {
    Util util = null;

    public void Execute(MQQueueManager qmgr, MQQueue queue,
        MQMessage message, string param)
    {
      util = new Util("NonTran");

      try
      {
        util.PrintMessage(message);
      }

      catch (Exception ex)
      {
        System.Console.WriteLine(">>> NonTran\n{0}", ex.ToString());
      }
    }
  }
}
```

Exemplo 3: Rotinas Comuns

```
/*********************************************************************/
/* Licensed materials, property of IBM                               */
/* 63H9336                                                           */
/* (C) Copyright IBM Corp. 2005, 2024.                               */
/*********************************************************************/

using System;

using IBM.WMQ;

namespace dnmsamp
{
    /// <summary>
    /// Summary description for Util.
    /// </summary>
    public class Util
    {
    /* ---------------------------------------------------------------- */
    /* Default prefix string of the namespace.                           */
    /* ---------------------------------------------------------------- */
    private string prefixText = "dnmsamp";

    /* ---------------------------------------------------------------- */
    /* Constructor that takes the replacement prefix string to use.     */
    /* ---------------------------------------------------------------- */
    public Util(String text)
    {
      prefixText = text;
    }

    /* ---------------------------------------------------------------- */
    /* Display an arbitrary string to the console.                       */
    /* ---------------------------------------------------------------- */
    public void Print(String text)
    {
      System.Console.WriteLine("{0} {1}\n", prefixText, text);
    }
```

```
 /* ------------------------------------------------------------------ */
    /* Display the content of the message passed to the console.          */
    /* ------------------------------------------------------------------ */
    public void PrintMessage(MQMessage message)
    {
      if (message.Format.CompareTo(MQC.MQFMT_STRING) == 0)
      {
        try
        {
          string messageText = message.ReadString(message.MessageLength);

          Print(messageText);
        }

        catch(Exception ex)
        {
          Print(ex.ToString());
        }
      }
      else
      {
        Print("UNRECOGNISED FORMAT");
      }
    }

    /* ------------------------------------------------------------------ */
    /* Convert the byte array into a hex string.                          */
    /* ------------------------------------------------------------------ */
    static public string ToHexString(byte[] byteArray)
    {
      string hex = "0123456789ABCDEF";

      string retString = "";

      for(int i = 0; i < byteArray.Length; i++)
      {
        int h = (byteArray[i] & 0xF0)>>4;
        int l = (byteArray[i] & 0x0F);

        retString += hex.Substring(h,1) + hex.Substring(l,1);
      }

      return retString;
    }

    }
}
```

Compilando Programas do WebSphere MQ .NET

Comandos de amostra para compilar aplicativos .NET gravados em várias linguagens.

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

Para construir um aplicativo C# usando as classes do WebSphere MQ para .NET, use o comando a seguir:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib:MQ_INSTALLATION_PATH\bin /out:MyProg.exe MyProg.cs
```

Para construir um aplicativo Visual Basic usando classes do WebSphere MQ para .NET, use o comando a seguir:

```
vbc /r:System.dll /r:MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:MyProg.exe MyProg.vb
```

Para construir um aplicativo C++ Gerenciado usando classes do WebSphere MQ para .NET, use o comando a seguir:

```
cl /clr MQ_INSTALLATION_PATH\bin Myprog.cpp
```

Para outros idiomas, consulte a documentação fornecida pelo fornecedor do idioma.

Rastreando Programas WebSphere MQ .NET

No WebSphere MQ .NET, você inicia e controla o recurso de rastreio como em programas WebSphere MQ usando o MQI.

Entretanto, os parâmetros -i e -p do comando strmqtrc, que permitem que você especifique identificadores de processo e encadeamento, e processos nomeados, não possuem efeito.

Normalmente é necessário usar o recurso de rastreio somente no pedido de serviço da IBM.

Consulte Usando rastreio no Windows para obter informações sobre comandos de rastreamento

IBM WebSphere MQ canal customizado para Microsoft Windows Communication Foundation (WCF)

O canal customizado do Microsoft Windows Communication Foundation (WCF) para o IBM WebSphere MQ envia e recebe mensagens entre clientes e serviços do WCF.

Conceitos relacionados

"Introdução ao uso do canal customizado do WebSphere MQ para WCF com .NET 3" na página 602
Visão geral das informações disponíveis para programadores usando o canal customizado do WebSphere MQ para o Windows Communication Foundation (WCF) com .NET 3.

"Usando canais customizados do WebSphere MQ para WCF" na página 606
Visão geral das informações disponíveis para programadores usando canais customizados do WebSphere MQ V7 para o Windows Communication Foundation (WCF).

"Usando as Amostras do WCF" na página 623
As amostras do Windows Communication Foundation (WCF) fornecem alguns exemplos simples de como o canal customizado WebSphere MQ pode ser usado.

"Determinação de problema no canal customizado do WCF para WebSphere MQ" na página 629
É possível usar o rastreio de WebSphere MQ para coletar informações detalhadas sobre o que várias partes do WebSphere MQ estão fazendo. Ao usar o Windows Communication Foundation (WCF), uma saída de rastreio separada é gerada para o rastreio de canal customizado do WCF integrado ao rastreio de infraestrutura do WCF Microsoft .

Introdução ao uso do canal customizado do WebSphere MQ para WCF com .NET 3

Visão geral das informações disponíveis para programadores usando o canal customizado do WebSphere MQ para o Windows Communication Foundation (WCF) com .NET 3.

Qual é o canal customizado do WebSphere MQ para WCF?

O canal customizado para o WebSphere MQ é um canal de transporte que usa o modelo de programação unificada Microsoft Windows Communication Foundation (WCF).

A estrutura Microsoft Windows Communication Foundation, introduzida no Microsoft .NET 3, permite que aplicativos e serviços .NET sejam desenvolvidos independentemente do transporte e dos protocolos usados para conectá-los, permitindo que transportes ou configurações alternativos sejam usados de acordo com o ambiente no qual o serviço ou aplicativo é implementado.

As conexões são gerenciadas no tempo de execução pelo WCF construindo uma pilha de canais que contém a combinação necessária de:

- Elementos de protocolo: Um conjunto opcional de elementos em que nenhum, um ou mais podem ser incluídos para suportar protocolos como os padrões WS-*.
- Codificador de mensagem: Um elemento obrigatório na pilha que controla a serialização da mensagem em seu formato de ligação.

- Canal de transporte: Um elemento obrigatório na pilha responsável por transportar a mensagem serializada para seu terminal.

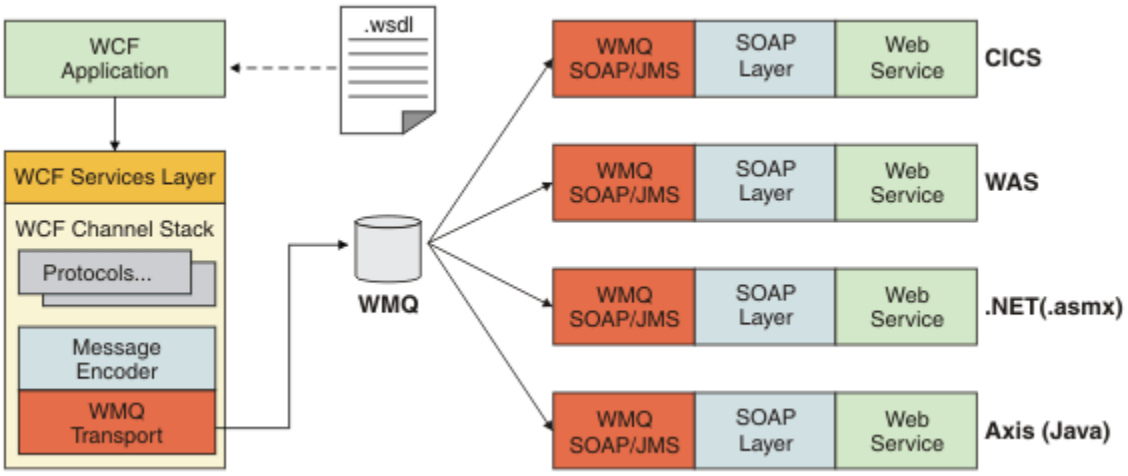
O canal customizado para o WebSphere MQ é um canal de transporte e, como tal, deve ser unido a um codificador de mensagem e protocolos opcionais conforme requerido pelo aplicativo usando uma ligação customizada do WCF. Dessa forma, os aplicativos que foram desenvolvidos para usar o WCF podem usar o canal customizado para o WebSphere MQ para enviar e receber dados da mesma forma que usam os transportes integrados fornecidos pela Microsoft, permitindo a integração simples com as funções do sistema de mensagens assíncronas, escaláveis e confiáveis do WebSphere MQ. Para obter uma lista completa das funções suportadas, consulte: "Recursos de Capacidades do Canal Customizado WCF" na página 606.

Quando e por que eu uso o canal customizado do WebSphere MQ para WCF?

O canal customizado do WebSphere MQ pode ser usado para enviar e receber mensagens entre clientes e serviços do WCF da mesma maneira que os transportes integrados fornecidos pela Microsoft, permitindo que os aplicativos acessem os recursos do WebSphere MQ no modelo de programação unificado do WCF.

Um cenário de padrão de uso típico do canal customizado do WebSphere MQ para WCF é como uma interface para serviços da web hospedados em WebSphere MQ (SOAP/JMS)

As mensagens são transportadas usando o formato de mensagem SOAP sobre JMS do WebSphere MQ, permitindo que os clientes e serviços WCF também chamem ou sejam chamados por outros aplicativos ou ambientes de hosting do WebSphere MQ que são compatíveis com esse formato, incluindo serviços da web e clientes em execução no WebSphere Application Server, CICS, Axis v1 (Java), e .asmx (.NET), conforme mostrado no diagrama a seguir:



Para obter detalhes sobre SOAP sobre JMS, consulte: "Transporte do WebSphere MQ para SOAP" na página 959

Um exemplo de um cenário típico do diagrama seria:

1. Um Serviço da Web hospedado no WebSphere Application Server e exposto sobre WebSphere MQ usando o suporte para SOAP sobre JMS no WebSphere Application Server
2. O documento WSDL que descreve o serviço pode, então, ser usado pela ferramenta WCF para gerar um proxy de cliente e uma configuração que poderá criar uma pilha de canais do WCF apropriada, incluindo o canal customizado.
3. O aplicativo cliente pode, então, usar o proxy para iniciar o serviço da Web da mesma maneira que qualquer outro serviço da Web.

O canal, geralmente, pode ser usado com um codificador de mensagem de texto/SOAP do WCF, mas o canal pode ser unido a outros codificadores de mensagem do WCF se necessário. Usar codificadores alternativos também pode fornecer integração limitada com aplicativos nativos do WebSphere MQ que não suportam SOAP sobre JMS, mas essa não é a função principal do canal.

Os principais benefícios de usar o canal customizado em um ambiente WCF são:

- Chamada assíncrona: Suportar o disparo e esquecer operações do cliente nas quais o cliente é separado da disponibilidade do serviço e dos recursos, como rotear novamente as respostas e o multi-hop.
- Características de escala confiável: Sistema de mensagens baseado em fila permite que a capacidade seja previsivelmente incluída em um sistema.
- Qualidade de serviço: As mensagens são tangíveis e rastreáveis e podem ser facilmente gerenciadas e administradas.

Requisitos de software e instruções de instalação para o canal customizado do WebSphere MQ para WCF

Este tópico descreve os requisitos de software e informações de instalação para o canal customizado do WebSphere MQ para WCF.

O canal customizado do WebSphere MQ para WCF pode conectar-se apenas aos gerenciadores de filas WebSphere MQ V7 ou superiores.

Requisitos de software para o canal customizado do WCF para WebSphere MQ

Estas informações listam os requisitos de software para o canal customizado do WCF para WebSphere MQ.

Ambiente de tempo de execução

- O Microsoft .NET Framework v3.0 ou superior deve ser instalado na máquina host.
- *Java e .NET Messaging e Web Services* são instalados por padrão como parte do instalador do WebSphere MQ 7.0.1. Instala os conjuntos .NET necessários para o canal customizado no Cache de Conjunto Global.

Nota: Se o Microsoft .NET Framework v2.0 ou superior não for instalado antes de instalar o WebSphere MQ V7.0.1, a instalação do produto WebSphere MQ continuará sem erro, mas o canal customizado WebSphere MQ estará indisponível. Se o .NET Framework for instalado após a instalação do WebSphere MQ 7.0.1, o canal customizado do WebSphere MQ deverá ser ativado executando-se o script *WMQInstallDir*\bin\amqiRegisterdotNet.cmd, em que *WMQInstallDir* é o diretório no qual o WebSphere MQ 7.0.1 é instalado. Este script instala as montagens necessárias no Global Assembly Cache (GAC). Um conjunto de arquivos amqi*.log gravando as ações executadas é criado no diretório %TEMP%. Não é necessário executar novamente o script amqiRegisterdotNet.cmd se .NET for atualizado para v3.0 ou superior de uma versão anterior, por exemplo, de .NET v2.0.

Ambiente de desenvolvimento

- Microsoft Visual Studio 2008 ou Windows Software Development Kit para .NET 3.0 ou posterior.
- O Microsoft .NET Framework V3.5 ou superior deve ser instalado na máquina host para construir os arquivos de soluções de amostra

Nota: Se o Microsoft .NET Framework v2.0 ou superior não for instalado antes de instalar o WebSphere MQ V7.0.1, a instalação do produto WebSphere MQ continuará sem erro, mas o canal customizado WebSphere MQ estará indisponível. Se o .NET Framework for instalado após a instalação do WebSphere MQ 7.0.1, o canal customizado do WebSphere MQ deverá ser ativado executando-se o script *WMQInstallDir*\bin\amqiRegisterdotNet.cmd, em que *WMQInstallDir* é o diretório no qual o WebSphere MQ 7.0.1 é instalado. Este script instala as montagens necessárias no Global Assembly Cache (GAC). Um conjunto de arquivos amqi*.log gravando as ações executadas é criado no diretório %TEMP%. Não é necessário executar novamente o script amqiRegisterdotNet.cmd se .NET for atualizado para v3.0 ou superior de uma versão anterior, por exemplo, de .NET v2.0.

Canal Customizado do WebSphere MQ para WCF: O que É Instalado?

O canal customizado para o WebSphere MQ é um canal de transporte usando o modelo de programação unificado do Microsoft Windows Communication Foundation (WCF). O canal customizado é instalado, por padrão, como parte da instalação do WebSphere MQ 7.0.1.

Canal Customizado do WebSphere MQ para WCF

O canal customizado do WebSphere MQ para WCF é instalado por padrão como parte da instalação do WebSphere MQ 7.0.1 ; o canal customizado e suas dependências estão contidos no componente `Java and .NET Messaging and Web Services` , que é instalado por padrão. Ao fazer upgrade para o WebSphere MQ 7.0.1 a partir de uma versão anterior, a atualização instalará o canal customizado do WebSphere MQ para WCF por padrão se o componente `Java and .NET Messaging and Web Services` tiver sido instalado anteriormente em uma instalação anterior.

O componente `Java and .NET Messaging and Web Services` contém o arquivo `IBM.XMS.WCF.dll` e o arquivo `IBM.XMS.WCF.dll` é o conjunto do canal customizado principal, que contém as classes de interfaces WCF.. Esse arquivo é instalado no Global Assembly Cache (GAC) e também está disponível no seguinte diretório: *MQ_INSTALLATION_PATH*`\bin` , em que *MQ_INSTALLATION_PATH* é o diretório no qual o WebSphere MQ 7.0.1 está instalado.

As classes de chave necessárias para usar o canal customizado estão no *Espaço de nomes: IBM.XMS.WCF* e:

Nome de Ligação de Transporte	IBM.XMS.WCF.SoapJmsIbmTransportBindingElement
Transport Binding Importer:	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementImporter
Configuração de Ligação de Transporte	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementConfig

Amostras do Canal Customizado do WebSphere MQ

As amostras fornecem alguns exemplos simples de como o canal customizado do WebSphere MQ para WCF pode ser usado. As amostras e seus arquivos associados estão localizados no diretório *MQ_INSTALLATION_PATH*`\tools\wcf\samples\` , em que *MQ_INSTALLATION_PATH* é o diretório de instalação do WebSphere MQ. Para obter informações adicionais sobre as amostras do canal customizado do WebSphere MQ, consulte: “Usando as Amostras do WCF” na página 623

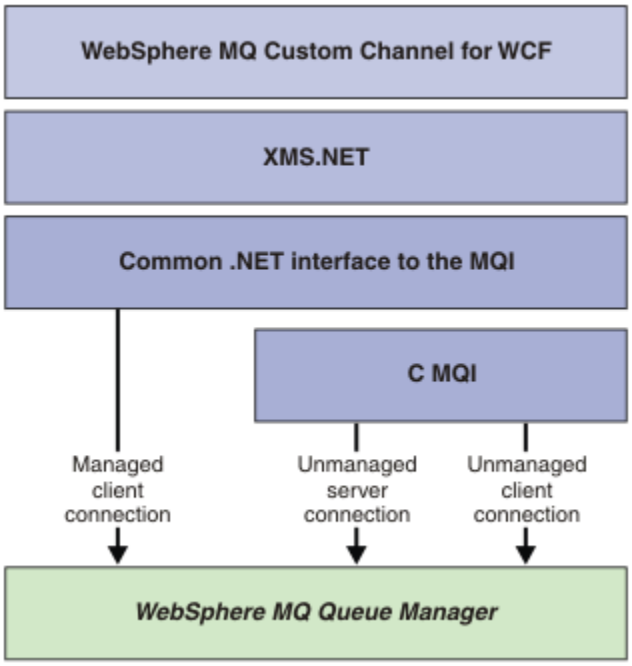
svcutil.exe.config

O `svcutil.exe.config` é um exemplo das definições de configuração necessárias para possibilitar que a ferramenta de geração de proxy de cliente `svcutil` do Microsoft WCF reconheça o canal customizado. O arquivo `svcutil.exe.config` está localizado no diretório *MQ_INSTALLATION_PATH*`\tools\wcf\docs\examples\` , em que *MQ_INSTALLATION_PATH* é o diretório de instalação do WebSphere MQ. Para obter informações adicionais sobre como usar o `svcutil.exe.config`, consulte: “Gerando um Proxy de Cliente do WCF e Arquivos de Configuração do Aplicativo Usando a Ferramenta svcutil com Metadados de um Serviço em Execução” na página 620.

Arquitetura do WCF

O canal customizado do WebSphere MQ para WCF é integrado sobre a API do `IBM Message Service Client for .NET (XMS .NET)` .

A arquitetura do WCF é conforme mostrada no diagrama a seguir:



Todos os componentes necessários são instalados por padrão com a instalação do WebSphere MQ V7.0.1 .

As três conexões são: conexões do cliente gerenciadas, conexões do servidor não gerenciadas e conexões do cliente não gerenciadas. Para obter mais informações sobre essas conexões, consulte “Opções de Conexão do WCF” na página 610.

Usando canais customizados do WebSphere MQ para WCF

Visão geral das informações disponíveis para programadores usando canais customizados do WebSphere MQ V7 para o Windows Communication Foundation (WCF).

O Microsoft Windows Communication Foundation sustenta os serviços da Web e o suporte ao sistema de mensagens no Microsoft .NET Framework 3 O WebSphere MQ V7 agora pode ser usado como um canal customizado no WCF no .NET Framework 3 da mesma maneira que os canais integrados oferecidos pela Microsoft.

As mensagens transportadas pelo canal customizado são formatadas de acordo com a implementação SOAP sobre JMS do WebSphere MQ V7. Os aplicativos podem então se comunicar com serviços hospedados pelo WCF ou pela infraestrutura de serviço SOAP sobre JMS do WebSphere . Para obter detalhes sobre SOAP sobre JMS, consulte: “Transporte do WebSphere MQ para SOAP” na página 959

Recursos de Capacidades do Canal Customizado WCF

Use os seguintes tópicos para obter informações relativas aos recursos e capacidades do canal customizado do WCF.

Formas de Canal Customizado WCF

Visão geral das formas de canal customizado que o WebSphere MQ pode ser usado como dentro dos canais customizados do Microsoft Windows Communication Foundation (WCF).

O canal customizado do WebSphere MQ para WCF suporta duas formas de canal:

- Unidirecional
- Pedido-resposta

O WCF automaticamente seleciona a forma de canal de acordo com o contrato de serviço que está sendo hospedado.

Os contratos que incluem métodos que usam apenas o parâmetro **IsOneWay** são servidos pela forma de canal unidirecional, por exemplo:

```
    [OperationContract(IsOneWay = true)]
    void printString(String text);
```

Os contratos que incluem uma mistura dos métodos unidirecional e solicitação e resposta ou todos os métodos de solicitação e resposta são atendidos pela forma de canal solicitação e resposta. Por exemplo:

```
    [OperationContract]
    int subtract(int a, int b);

    [OperationContract(IsOneWay = true)]
    void printString(string text);
```

Nota: Ao misturar os métodos unidirecional e pedido-resposta no mesmo contrato, deve-se assegurar que o comportamento seja o pretendido, especialmente ao trabalho em um ambiente misto porque métodos unidirecionais esperarão até que recebam uma resposta nula do serviço.

Canal unidirecional

O canal customizado unidirecional do WebSphere MQ para WCF é usado, por exemplo, para enviar mensagens de um cliente WCF usando um formato de canal unidirecional. O canal pode enviar mensagens apenas em uma direção, por exemplo, de um gerenciador de filas do cliente para uma fila em um serviço do WCF.

Canal Pedido-Resposta

O canal customizado de solicitação / resposta do WebSphere MQ para WCF é usado, por exemplo, para enviar mensagens em duas direções de forma assíncrona; a mesma instância do cliente deve ser usada para o sistema de mensagens assíncronas. O canal pode enviar mensagens em uma direção, por exemplo, de um gerenciador de filas do cliente para uma fila em um serviço do WCF e, então, enviar uma mensagem de resposta do WCF para uma fila no gerenciador de filas do cliente.

Nomes e Valores de Parâmetros da URI do WCF

connectionFactory
O parâmetro connectionFactory é necessário. Para a sintaxe desse parâmetro, consulte Sintaxe de URI e parâmetros para implementação de serviço da web

initialContextFactory
O parâmetro Factory initialContexté necessário e deve ser configurado como "com.ibm.mq.jms.Nojndi" para compatibilidade com o WebSphere Application Server e outros produtos (consulte “Implementando um serviço no WebSphere Application Server para usar o WebSphere Transporte para SOAP” na página 1017).

Entrega Assegurada do Canal Customizado WCF

A Entrega Assegurada garante que um pedido ou resposta de serviço é acionada e não perdida.

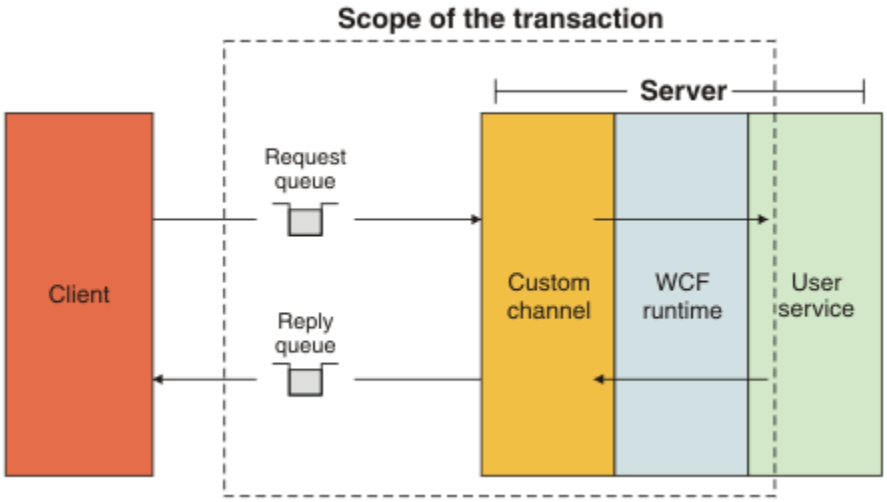
Uma mensagem de pedido é recebida e qualquer mensagem de resposta é enviada em um ponto de sincronização da transação local, que pode ser retrocedido no caso de falha de tempo de execução. Exemplos dessas falhas são: Uma exceção não manipulada lançada pelo serviço, falha para executar o dispatch da mensagem para o serviço ou falha para entregar a mensagem de resposta.

`AssuredDelivery` é o atributo de entrega assegurada que pode ser especificado em um contrato de serviço para garantir que as mensagens de pedido recebidas por um serviço e a mensagem de resposta enviada de um serviço não sejam perdida no caso de uma falha de tempo de execução.

Para assegurar que as mensagens também sejam preservadas no caso de falha do sistema ou queda de energia, elas devem ser enviadas como persistentes. Para usar as mensagens persistentes, o aplicativo

cliente deve ter esta opção especificada no URI do terminal. Para obter informações adicionais sobre como configurar propriedades do URI, veja: Sintaxe de URI e os parâmetros para implementação do serviço da web.

As transações distribuídas não são suportadas e o escopo da transação não se estende além do processamento de mensagens de solicitação e resposta executado pelo WebSphere MQ. Qualquer trabalho executado dentro do serviço pode ser executado novamente como resultado de uma falha que faz com que a mensagem seja recebida outra vez. O diagrama a seguir mostra o escopo da transação:



Entrega assegurada é ativada aplicando o atributo `AssuredDelivery` à classe de serviço conforme mostrado no exemplo a seguir:

```
[AssuredDelivery]
class TestCalculatorService : IWMQSampleCalculatorContract
{
    public int add(int a, int b)
    {
        int ans = a + b;
        return ans;
    }
}
```

Ao usar o atributo `AssuredDelivery`, você deve estar ciente dos seguintes pontos:

- Quando um canal determina que uma falha provavelmente ocorrerá outra vez se uma mensagem tiver sido revertida e recebida novamente, a mensagem será tratada como uma mensagem suspeita e não é retornada à fila de pedidos para reprocessamento. Por exemplo: Se a mensagem recebida não está formatada corretamente ou não pode ter o dispatch executado para um serviço. As exceções não manipuladas lançadas de uma operação de serviço são sempre reenviadas até que a mensagem tenha sido entregue novamente o número máximo de vezes especificado pela propriedade de limite de restauração da fila de solicitações. Para obter informações adicionais, consulte: “Mensagens Suspeitas do Canal Customizado WCF” na página 609
- O canal executa a leitura, o processamento e a resposta de cada mensagem de pedido como uma operação atômica usando um único encadeamento de execução para reforçar a integridade transacional. Para possibilitar que as operações de serviço sejam executadas simultaneamente, o canal permite que o WCF crie várias instâncias do canal. O número de instâncias do canal disponíveis para pedidos de processamento é controlado pela propriedade de ligação `MaxConcurrentCalls`. Para obter informações adicionais, consulte: “Opções de Configuração de Ligação do WCF” na página 616
- A função de entrega assegurada usa os pontos de extensibilidade do WCF IOperationInvoker e IErrorHandler. Se esses pontos de extensibilidade forem usados externamente por um aplicativo, o aplicativo deverá assegurar que qualquer ponto de extensibilidade registrado anteriormente seja chamado. A falha em fazer isso para IErrorHandler pode resultar em erros não serem relatados. A falha em fazer isso para IOperationInvoker pode fazer com que o WCF pare de responder.

Segurança do Canal Customizado WCF

O canal customizado do WebSphere MQ para WCF suporta o uso de SSL apenas para conexões do cliente não gerenciadas para o gerenciador de filas.

O SSL pode ser especificado de duas maneiras:

- Especificar o SSL diretamente na URI do SOAP sobre JMS. Para obter uma descrição completa das opções SSL, consulte SSL e o transporte do WebSphere MQ para SOAP
- Especificar o SSL usando uma entrada na Client Channel Definition Table (CCDT). Para obter mais informações sobre CCDTs, consulte Tabela de definição de canal de cliente

Client Channel Definition Tables (CCDT) do WCF

O canal customizado do WebSphere MQ para WCF suporta o uso de tabelas de definições de canais do cliente (CCDT) para configurar as informações de conexão para conexões do cliente..

As CCDTs são controladas através dessas duas variáveis de ambiente:

- *MQCHLLIB* especifica o diretório no qual a tabela está localizada.
- *MQCHLTAB* especifica o nome do arquivo da tabela.

Não é possível especificar a tabela de definição de canal diretamente na URI do SOAP sobre JMS. Se essas variáveis de ambiente estão definidas, elas terão prioridade sobre qualquer detalhe de conexão do cliente especificado no URI.

Para obter mais informações sobre tabelas de definições de canal de cliente, consulte: Tabela de definição de canal de cliente .

Conceitos relacionados

Tabela de Definições de Canal do Cliente

Mensagens Suspeitas do Canal Customizado WCF

Quando um serviço falha em processar uma mensagem de pedido ou falha em entregar uma mensagem de resposta para uma fila de resposta, a mensagem é tratada como uma mensagem suspeita.

Mensagens de Pedido Suspeitas

Se uma mensagem de pedido não puder ser processada, ela será tratada como uma mensagem suspeita. Esta ação evita que o serviço receba a mesma mensagem não processável novamente. Para uma mensagem de pedido não processável ser tratada como uma mensagem suspeita, uma das seguintes situações deve ser verdadeira:

- A contagem de restaurações de mensagens excedeu o limite de restauração especificado na fila de pedidos, o que só ocorre se entrega garantida tiver sido especificada para o serviço. Para obter mais informações sobre a entrega garantida, veja: “Entrega Assegurada do Canal Customizado WCF” na página 607
- A mensagem não foi formatada corretamente e não pôde ser interpretada como uma mensagem SOAP sobre JMS.

Mensagens de Resposta Suspeitas

Se um serviço falhar em entregar uma mensagem de resposta para a fila de resposta, a mensagem de resposta será tratada como uma mensagem suspeita. Para mensagens de resposta, esta ação possibilita que as mensagens de resposta sejam recuperadas posteriormente para auxiliar na determinação de problema.

Manipulação de Mensagens Suspeitas

A ação tomada para uma mensagem suspeita depende da configuração do gerenciador de filas e dos valores configurados nas opções de relatório da mensagem. Para SOAP sobre JMS, as seguintes opções de relatório são configuradas nas mensagens de pedido por padrão e não são configuráveis:

- MQRO_EXCEPTION_WITH_FULL_DATA

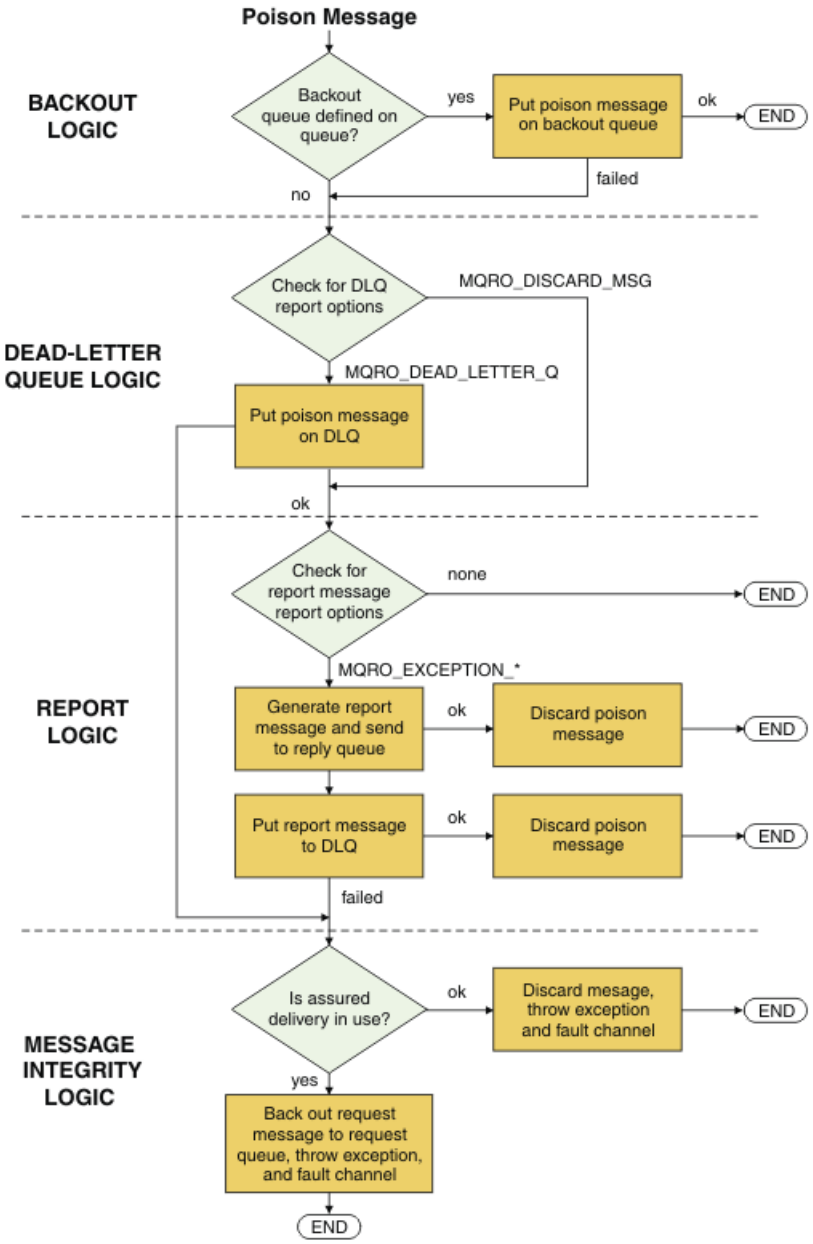
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_DISCARD_MSG

Para SOAP sobre JMS, a seguinte opção de relatório é configurada em mensagens de resposta por padrão e não é configurável:

- MQRO_DEAD_LETTER_Q

Se as mensagens vierem de uma origem não WCF, consulte a documentação para essa origem.

O diagrama a seguir mostra as possíveis ações e as etapas realizadas se a manipulação de uma mensagem suspeita falhar:



Opções de Conexão do WCF

Há três modos de conectar um canal customizado do WebSphere MQ para WCF a um gerenciador de filas. Considere qual tipo de conexão melhor se adequa aos seus requisitos.

Para obter informações adicionais sobre opções de conexão, consulte: “Diferenças de Conexão” na página 582

Para obter informações adicionais sobre arquitetura WCF, veja: “Arquitetura do WCF” na página 605

Conexão do Cliente Não Gerenciada

Uma conexão feita neste modo conecta-se como um cliente do WebSphere MQ a um servidor WebSphere MQ em execução na máquina local ou em uma máquina remota.

Para usar o canal customizado do WebSphere MQ para WCF como um cliente WebSphere MQ , é possível instalá-lo, com o cliente MQI do WebSphere MQ , no servidor WebSphere MQ ou em uma máquina separada.

Conexão do Servidor Não Gerenciada

Quando usada no modo de ligações do servidor, o canal customizado do WebSphere MQ para WCF usa a API do gerenciador de filas em vez de se comunicar através de uma rede. Usar conexões de ligações fornece melhor desempenho para aplicativos WebSphere MQ do que usar conexões de rede.

Para usar a conexão de ligações, você deve instalar o canal customizado do WebSphere MQ para WCF no servidor WebSphere MQ .

Conexão do Cliente Gerenciada

Uma conexão feita neste modo conecta-se como um cliente do WebSphere MQ a um servidor WebSphere MQ em execução na máquina local ou em uma máquina remota.

As classes do canal customizado do WebSphere MQ para .NET 3 que se conectam neste modo permanecem no código gerenciado .NET e não fazem chamadas para serviços nativos. Para obter mais informações sobre o código gerenciado, consulte a documentação da Microsoft

Existem várias limitações no uso do cliente gerenciado. Para obter informações adicionais sobre estas limitações, veja “Conexões do Cliente Gerenciadas” na página 583.

Criando e configurando o canal customizado do WebSphere MQ para WCF

Os canais customizados do WebSphere MQ V7 para WCF funcionam da mesma maneira que os canais WCF de transporte oferecidos pela Microsoft. O canal customizado do WebSphere MQ para WCF pode ser criado de uma das duas maneiras.

Sobre esta tarefa

O canal customizado do WebSphere MQ se integra ao WCF como um canal de transporte WCF e, como tal, deve ser emparelhado com um codificador de mensagens e canais de protocolo opcionais, para que ele possa criar uma pilha de canais completa que possa ser usada por um aplicativo. Dois elementos são necessários para que uma pilha completa de canais para seja criada com êxito:

1. Uma definição de ligação: Especifica quais elementos são necessários para construir a pilha de canais de aplicativos, incluindo canal de transporte, codificador de mensagens e quaisquer protocolos além das definições gerais de configuração. Para o canal customizado, a definição de ligação deve ser criada na forma de uma ligação customizada WCF.
2. Uma definição de terminal: Vincula o contrato de serviço à definição de ligação e também fornece o URI de conexão real que descreve onde o aplicativo pode se conectar. Para o canal customizado, o URI está no formato de um URI SOAP sobre JMS.

Estas definições podem ser criadas de duas maneiras diferentes:

- Administrativamente: As definições são criadas fornecendo os detalhes em um arquivo de configuração de aplicativo (por exemplo: `app.config`).
- Programaticamente: As definições são criadas diretamente do código do aplicativo.

A decisão sobre qual método usar para criar as definições deve ser baseada nos requisitos do aplicativo conforme segue:

- O método administrativo para configuração fornece a flexibilidade para alterar os detalhes do serviço e pós-implementação de cliente sem reconstruir o aplicativo.
- O método programático para configuração fornece maior proteção contra erros de configuração e a capacidade de gerar dinamicamente uma configuração no tempo de execução.

Criando um Canal Customizado do WCF Administrativamente Fornecendo Informações de Ligação e Terminal em um Arquivo de Configuração do Aplicativo

O canal customizado WebSphere MQ para WCF é um canal WCF de nível de transporte. Um terminal e ligação devem ser definidos para usar o canal customizado e estas definições podem ser feitas fornecendo as informações de ligação e terminal em um arquivo de configuração do aplicativo.

Para configurar e usar o canal customizado do WebSphere MQ para WCF, que é um canal WCF de nível de transporte, uma ligação e uma definição de terminal devem ser definidas. A ligação contém as informações de configuração para o canal e a definição de terminal contém os detalhes da conexão. Estas definições podem ser criados de duas formas:

- Programaticamente diretamente a partir do código do aplicativo, conforme descrito aqui: "Criando um Canal Customizado do WCF Fornecendo Informações de Ligação e Terminal Programaticamente" na página 614
- Administrativamente, fornecendo os detalhes em um arquivo de configuração do aplicativo, conforme descrito no procedimento a seguir.

O arquivo de configuração do cliente ou do aplicativo de serviço é comumente denominado *yourappname*`.exe.config` em que *yourappname* é o nome de seu aplicativo. O arquivo de configuração do aplicativo é modificado mais facilmente usando a ferramenta do editor de configuração de serviço Microsoft chamada `SvcConfigEditor.exe` da seguinte maneira:

- Inicie a ferramenta do editor de configuração `SvcConfigEditor.exe`. O local de instalação padrão para a ferramenta é: *`Drive:`*`\Program Files\Microsoft SDKs\Windows\v6.0\Bin\SvcConfigEditor.exe`, em que *Drive:* é o nome da unidade de instalação.

Etapa 1: Incluir uma Extensão de Elemento de Ligação para Ativar o WCF para Localizar o Canal Customizado

1. Clique com o botão direito em **Avançado** > **Extensão** > **elemento de ligação** para abrir o menu e selecione **Novo**
2. Preencha os campos conforme mostrado nesta tabela:

Tabela 73. Novos Campos do Elemento de Ligação

Campo	**Value**
Nome	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`
Tipo	Navegue para `IBM.XMS.WCF.dll` no Global Assembly Cache (GAC) e selecione `IBM.XMS.WCFSoapJmsIbmTransportBindingElementConfig`

Etapa 2: Criar uma Definição de Ligação Customizada que Une o Canal Customizado a um Codificador de Mensagem do WCF

1. Clique com o botão direito em **Ligações** para abrir o menu e selecione **Nova configuração de ligação**
2. Preencha os campos conforme mostrado nesta tabela:

Tabela 74. Novos Campo de Configuração de Ligação	
Campo	**Value**
Nome	`CustomBinding_WMQ`
BindingElement 1	`textMessageEncoding (MessageVersion: Soap11)`
BindingElement 2	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`

Etapa 3: Especificar as Propriedades de Ligação

1. Selecione a ligação de transporte de *IBM.XMS.WCF.SoapJmsIbmTransportChannel* a partir da ligação que você criou em: "Etapa 2: Criar uma Definição de Ligação Customizada que Une o Canal Customizado a um Codificador de Mensagem do WCF" na página 612
2. Faça qualquer alteração necessária nos valores-padrão das propriedades conforme descrito em: "Opções de Configuração de Ligação do WCF" na página 616

Step 4: Criar uma Definição de Terminal

Crie uma definição de terminal que faça referência à ligação customizada criada em "Etapa 2: Criar uma Definição de Ligação Customizada que Une o Canal Customizado a um Codificador de Mensagem do WCF" na página 612 e forneça os detalhes da conexão do serviço. A maneira como estas informações são especificadas é dependente de se a definição é para um aplicativo cliente ou um aplicativo de serviço.

Para um aplicativo cliente, inclua uma definição de terminal na seção do cliente conforme a seguir:

1. Clique com o botão direito em **Cliente** > **Terminais** para abrir o menu e selecione **Novo Terminal do Cliente**
2. Preencha os campos conforme mostrado nesta tabela:

Tabela 75. Novos Campos de Terminal do Cliente	
Campo	**Value**
Nome	`Endpoint_WMQ`
Endereço	*O URI SOAP/JMS que descreve os detalhes de conexão do WMQ necessários para acessar o serviço.* Para obter detalhes adicionais, consulte: "WebSphere MQ canal customizado para o formato de endereço do URI do terminal WCF" na página 615
Ligação	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Contrato	*O nome de sua interface de contrato do serviço*

Para um aplicativo de serviço, inclua uma definição de serviço na seção de serviços conforme a seguir:

1. Clique com o botão direito em **Serviços** para abrir o menu e selecione **Novo Serviço** e, em seguida, selecione a classe de serviço a ser hospedada.
2. Inclua uma definição de terminal à seção **Terminais** ao seu novo serviço e preencha os campos conforme mostrado nesta tabela:

Tabela 76. Novos Campos do Terminal em Serviço	
Campo	**Value**
Nome	`Endpoint_WMQ`

Tabela 76. Novos Campos do Terminal em Serviço (continuação)	
Campo	**Value**
Endereço	*O URI SOAP/JMS que descreve os detalhes de conexão do WMQ necessários para acessar o serviço.* Para obter detalhes adicionais, consulte: "WebSphere MQ canal customizado para o formato de endereço do URI do terminal WCF" na página 615
Ligação	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Contrato	*O nome de sua classe de implementação de serviço*

Criando um Canal Customizado do WCF Fornecendo Informações de Ligação e Terminal Programaticamente

O canal customizado WebSphere MQ para WCF é um canal WCF de nível de transporte. Um terminal e uma ligação devem ser definidos para usar o canal customizado e estas definições podem ser feitas programaticamente diretamente a partir do código do aplicativo.

Para configurar e usar o canal customizado do WebSphere MQ para WCF, que é um canal WCF de nível de transporte, uma ligação e uma definição de terminal devem ser definidas. A ligação contém as informações de configuração para o canal e a definição de terminal contém os detalhes da conexão. Para obter mais informações, consulte: "Usando as Amostras do WCF" na página 623

Estas definições podem ser criados de duas formas:

- Administrativamente, fornecendo os detalhes em um arquivo de configuração do aplicativo, conforme descrito aqui: "Criando um Canal Customizado do WCF Administrativamente Fornecendo Informações de Ligação e Terminal em um Arquivo de Configuração do Aplicativo" na página 612
- Programaticamente diretamente do código do aplicativo, conforme descrito no exemplo a seguir.

Etapa 1: Criar uma instância do elemento de ligação de transporte do canal

Inclua o seguinte código em seu aplicativo:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
```

Etapa 2: Configurar Propriedades de Ligação

Configure quaisquer propriedades de ligação necessárias, por exemplo incluindo o código a seguir em seu aplicativo para configurar o `ClientConnectionMode`

```
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.AS_URI;
```

Etapa 3: Criar uma Ligação Customizada que Une o Canal de Transporte a um Codificador de Mensagem

Crie uma ligação customizada incluindo o seguinte código no aplicativo:

```
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
```

Etapa 4: Criando o URI SOAP/JMS

O URI SOAP/JMS que descreve os detalhes da conexão do WebSphere MQ necessários para acessar o serviço deve ser fornecido como o endereço do terminal. Isso depende se o canal está sendo usado para um aplicativo de serviço ou um aplicativo cliente.

Para aplicativos clientes, o URI SOAP/JMS deve ser criado como um `EndpointAddress` conforme a seguir:

```
EndpointAddress address = new EndpointAddress("jms:/queue?
destination=SampleQ@QM1&connectionFactory=connectQueueManager(QM1)&initialContextFactory=com.ibm
.mq.jms.Nojndi");
```

Para aplicativos de serviços, o URI SOAP/JMS deve ser criado como um URI conforme a seguir:

```
Uri address = new Uri("jms:/queue?
destination=SampleQ@QM1&connectionFactory=connectQueueManager(QM1)&initialContextFactory=com.ibm
.mq.jms.Nojndi");
```

Para obter mais informações sobre o endereço do terminal: “WebSphere MQ canal customizado para o formato de endereço do URI do terminal WCF” na página 615

WebSphere MQ canal customizado para o formato de endereço do URI do terminal WCF

Um Universal Resource Identifier (URI) fornece detalhes de local e conexão para especificar um serviço da web. Este formato de URI permite um grau abrangente de controle sobre os parâmetros e opções específicos do SOAP/ WebSphere MQao acessar serviços de destino.

Um serviço da Web é especificado usando um Universal Resource Identifier (URI). Esta seção especifica o formato de URI que é suportado no transporte do WebSphere MQ para SOAP. Este formato de URI permite um grau abrangente de controle sobre os parâmetros e opções específicos do SOAP/WebSphere MQao acessar serviços de destino. Este formato é compatível com o WebSphere Application Server (WAS) e com o CICS facilitando a integração do WebSphere MQ com ambos os produtos.

A sintaxe de URI é a seguinte:

```
jms:/queue?name=value&name=value...
```

em que `name` é um nome de parâmetro e *value* é um valor apropriado e o elemento `name`=*value* pode ser repetido várias vezes com a segunda e as ocorrências subsequentes sendo precedidas por um e comercial (&).

Para obter informações adicionais sobre como configurar propriedades de URI, consulte: Sintaxe de URI e Parâmetros para Implementação de Serviço da Web

Os nomes de parâmetros fazem distinção entre maiúsculas e minúsculas, assim como os nomes dos objetos WebSphere MQ . Se qualquer parâmetro for especificado mais de uma vez, a ocorrência final do parâmetro entrará em vigor, o que significa que os aplicativos clientes podem substituir os valores de parâmetro anexando no URI. Se quaisquer parâmetros não reconhecidos adicionais forem incluídos, eles serão ignorados

Se você armazenar um URI em uma sequência de XML, deverá representar o caractere de e comercial como "&". Da mesma forma, se um URI for codificado em um script, tome cuidado para escapar caracteres como **&** que, de outra forma, seriam interpretados pelo shell.

Esse é um exemplo de um URI simples para um serviço do Axis:

```
jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Aqui está um exemplo de um URI simples para um serviço .NET:

```
jms:/queue?destination=myQ&connectionFactory=()&targetService=MyService.asmx
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Somente os parâmetros necessários são fornecidos (`targetService` é necessário apenas para serviços .NET) e `connectionFactory` não tem opções.

Neste exemplo de Eixo, `connectionFactory` contém várias opções:

```
jms:/queue?destination=myQ@myRQM&connectionFactory=connectQueueManager(myconnQM)
binding(client)clientChannel(myChannel)clientConnection(myConnection)
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Neste exemplo de Axis, a opção `sslPeerName` de `connectionFactory` também foi especificada. O valor de sslPeerName em si contém pares de nome-valor e espaços em branco integrados significativos:

```
jms:/queue?destination=myQ@myRQM&connectionFactory=connectQueueManager(myconnQM)
binding(client)clientChannel(myChannel)clientConnection(myConnection)
sslPeerName(CN=MQ Test 1,O=IBM,S=Hampshire,C=GB)
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Opções de Configuração de Ligação do WCF

Este tópico descreve como as opções de configuração podem ser aplicadas às informações de ligação de canais customizados e lista as opções disponíveis.

As opções de configuração de ligação podem ser configuradas de uma de duas maneiras diferentes:

1. Administrativamente: As configurações da propriedade de ligação devem ser especificadas na seção de transporte da definição de ligação customizada no arquivo de configuração de aplicativos, por exemplo: `app.config`
2. Programaticamente: O código do aplicativo deve ser modificado para especificar a propriedade durante a inicialização da ligação customizada.

Configurando as Propriedades de Ligação Administrativamente

As configurações da propriedade de ligação também podem ser especificadas no arquivo de configuração de aplicativo, por exemplo: `app.config`. O arquivo de configuração é gerado por **svcutil**, por exemplo:

```
<customBinding>

...

     <IBM.XMS.WCF.SoapJmsIbmTransportChannel maxBufferPoolSize="524288"
      maxMessageSize="4000000" clientConnectionMode="0" maxConcurrentCalls="16"/>

...

</customBinding>
```

Configurando as Propriedades de Ligação Programaticamente

Para incluir uma propriedade de ligação de WCF para especificar o modo de conexão do cliente, você deve modificar o código de serviço para especificar a propriedade durante a inicialização da ligação customizada.

Use o exemplo a seguir para especificar o modo de conexão do cliente não gerenciado:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.CLIENT_UNMANAGED;

Binding sampleBinding = new CustomBinding(new TextMessageEncodingBindingElement(),
                                                    transportBindingElement);
```

Propriedades de Ligação do WCF

Nome da Propriedade	Aplicativo Cliente ou de Serviço	Valor Administrativo	Valor Programático	Descrição
maxBufferPoolSize	Ambos	0 para 64 bit número inteiro assinado	0 para 64 bit número inteiro assinado	Especifica o tamanho máximo da memória que pode ser utilizada para armazenar os buffers de mensagem do WCF para uma instância do canal.
maxMessageSize	Ambos	1 para 32 bit número inteiro assinado	1 para 32 bit número inteiro assinado	Especifica a memória máxima que pode ser utilizada para uma mensagem do WCF individual.
clientConnectionMode	Ambos	0 (valor padrão) 1	AS_URI (valor padrão) CLIENT_UNMANAGED	Especifica o modo de conexão do cliente do canal de transporte. 0 significa que o modo de conexão do cliente é conforme especificado no URI. Usado somente se a conexão do cliente for usada. Especifica que o modo de conexão do cliente é conforme especificado no URI. 0 é o valor padrão se nenhum modo de conexão do cliente estiver configurado. 1 significa que o modo de conexão do cliente é um cliente não gerenciado. Usado somente se a conexão do cliente for usada.
MaxConcurrentCalls	Client	O intervalo é de 0–2 147 483 647 16 é o valor padrão	O intervalo é de 0–2 147 483 647 16 é o valor padrão	Essa propriedade define o número máximo de operações simultâneas que podem ocorrer em um proxy de cliente individual a qualquer momento. Se mais operações forem iniciadas, elas serão enfileiradas até que uma operação em andamento seja concluída ou expire. Esta configuração pode ser usada para controlar o máximo de encadeamentos e recursos que podem ser consumidos por um proxy individual. 0 remove este limite, possibilitando que todas as operações sejam tentadas simultaneamente.

Nome da Propriedade	Aplicativo Cliente ou de Serviço	Valor Administrativo	Valor Programático	Descrição
MaxConcurrentCalls	Serviço	O intervalo é de 1–2 147 483 647 16 é o valor padrão	O intervalo é de 1–2 147 483 647 16 é o valor padrão	Esta propriedade é usada apenas se o recurso de entrega assegurada for ativado (para obter mais informações sobre a entrega assegurada, veja “Entrega Assegurada do Canal Customizado WCF” na página 607). Ela especifica o número máximo de operações simultâneas que podem estar em andamento ao mesmo tempo para o terminal determinado. É necessário tomar cuidado ao alterar esta configuração. Cada operação simultânea requer recursos adicionais, em particular uma nova instância do canal customizado e os encadeamentos associados do conjunto de encadeamentos para acionar os pedidos. A alocação em excesso pode ser contraproducente e afetar o desempenho gravemente. A configuração apropriada do conjunto de encadeamentos deve ser feita para suportar esta propriedade.

Construindo e Hospedando Serviços para WCF

Visão geral dos serviços do Microsoft Windows Communication Foundation (WCF) explicando como criar e configurar serviços do WCF.

O canal customizado do IBM WebSphere MQ para WCF e serviços do WCF que o usam pode ser hospedado pelos seguintes métodos:

- Auto-hosting
- Windows Serviço

O canal customizado do IBM WebSphere MQ para WCF não pode ser hospedado no Windows Process Activation Service.

Os tópicos a seguir fornecem alguns exemplos de auto-hosting simples para demonstrar as etapas envolvidas. A documentação on-line do WCF do Microsoft, que contém informações adicionais e os detalhes mais recentes, pode ser localizada no website MSDN do Microsoft em https://msdn.microsoft.com.

Construindo Aplicativos de Serviço WCF Usando o Método 1: Auto-hospedando Administrativamente Usando um Arquivo de Configuração do Aplicativo

Depois de criar um arquivo de configuração do aplicativo, abra uma instância do serviço e incluir o código especificado para seu aplicativo.

Antes de começar

Crie ou edite um arquivo de configuração do aplicativo para o serviço, conforme descrito em: "Criando um Canal Customizado do WCF Administrativamente Fornecendo Informações de Ligação e Terminal em um Arquivo de Configuração do Aplicativo" na página 612

Sobre esta tarefa

1. Instancie e abra uma instância do serviço no host do serviço. O tipo de serviço deve ser o mesmo que o tipo de serviço especificado no arquivo de configuração do serviço.
2. Inclua o seguinte código em seu aplicativo:

```
ServiceHost service = new ServiceHost(typeof(MyService));
service.Open();
...
service.Close();
```

Construindo Aplicativos de Serviço do WCF Usando o Método 2: Auto-hospedando Programaticamente Diretamente a partir do Aplicativo

Inclua as propriedades de ligação, crie o host de serviço com uma instância da classe de serviço necessária e abrir o serviço.

Antes de começar

1. Inclua uma referência ao arquivo `IBM.XMS.WCF.dll` do canal customizado no projeto. O `IBM.XMS.WCF.dll` está em *`WMQInstallDir`*`\bin` em que *WMQInstallDir* é o diretório em que o WebSphere MQ 7 está instalado.
2. Inclua uma instrução *using* no namespace `IBM.XMS.WCF`, por exemplo: `using IBM.XMS.WCF`
3. Crie uma instância do elemento de ligação de canais e terminal conforme descrito em: "Criando um Canal Customizado do WCF Fornecendo Informações de Ligação e Terminal Programaticamente" na página 614

Sobre esta tarefa

Se as mudanças para as propriedades de ligação do canal forem necessárias, conclua as seguintes etapas:

1. Inclua as propriedades de ligação em `transportBindingElement` conforme mostrado no exemplo a seguir:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
Uri address = new Uri("jms:/queue?destination=SampleQ@QM1&connectionFactory=

connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
```

2. Crie o host de serviço com uma instância da classe de serviço necessária:

```
ServiceHost service = new ServiceHost(typeof(MyService));
```

3. Abra o serviço:

```
service.AddServiceEndpoint(typeof(IMyServiceContract), binding, address);
service.Open();
    ...
service.Close();
```

Expondo Metadados Usando um Terminal HTTP

Instruções para expor os metadados de um serviço configurado para usar o canal customizado do WebSphere MQ para WCF.

Sobre esta tarefa

Se os metadados de serviços precisarem ser expostos (para que ferramentas como `svcutil` possam acessá-los diretamente a partir do serviço em execução, em vez de a partir de um arquivo WSDL off-line, por exemplo), isso deverá ser feito expondo os metadados de serviços com um terminal HTTP. As etapas a seguir podem ser usadas para incluir este terminal adicional.

1. Inclua o endereço de base de onde os metadados devem ser expostos no ServiceHost, por exemplo:

```
ServiceHost service = new ServiceHost(typeof(TestService),
                      new Uri("http://localhost:8000/MyService"));
```

2. Inclua o seguinte código no ServiceHost antes do serviço ser aberto:

```
ServiceMetadataBehavior metadataBehavior = new ServiceMetadataBehavior();
  metadataBehavior.HttpGetEnabled = true;
  service.Description.Behaviors.Add(metadataBehavior);
  service.AddServiceEndpoint(typeof(IMetadataExchange),
     MetadataExchangeBindings.CreateMexHttpBinding(), "mex");
```

Resultados

Os metadados estão agora disponíveis no endereço a seguir: `http://localhost:8000/MyService`

Construindo Aplicativos Clientes para WCF

Visão geral da geração e construção de aplicativos clientes Microsoft Windows Communication Foundation (WCF).

Um aplicativo cliente pode ser criado para um serviço WCF; os aplicativos clientes são, geralmente, gerados usando o Microsoft ServiceModel Metadata Utility Tool (Svcutil.exe) para criar os arquivos de configuração e de proxy necessários que podem ser usados diretamente pelo aplicativo.

Gerando um Proxy de Cliente do WCF e Arquivos de Configuração do Aplicativo Usando a Ferramenta svcutil com Metadados de um Serviço em Execução

Instruções para usar a ferramenta Microsoft svcutil.exe para gerar um cliente para um serviço configurado para usar o canal customizado do WebSphere MQ para WCF.

Antes de começar

Há três pré-requisitos para usar a ferramenta svcutil para criar arquivos de configuração e proxy necessários que podem ser usados diretamente pelo aplicativo:

- O serviço do WCF deve estar em execução antes da ferramenta svcutil ser iniciada.
- O serviço WCF deve expor seus metadados usando uma porta HTTP além das referências de terminal de canal customizado do WebSphere MQ para gerar um cliente diretamente a partir de um serviço em execução.
- O canal customizado deve ser registrado nos dados de configuração para svcutil.

Sobre esta tarefa

As etapas a seguir explicam como gerar um cliente para um serviço configurado para usar o canal customizado do WebSphere MQ , mas também expõe seus metadados no tempo de execução por meio de uma porta HTTP separada:

1. Inicie o serviço do WCF (O serviço deve estar em execução antes da ferramenta svcutil ser iniciada).
2. Inclua os detalhes do arquivo de configuração `svcutil.exe` da raiz da instalação no arquivo de configuração svcutil ativo, geralmente `C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\svcutil.exe.config` para que svcutil reconheça o canal customizado WebSphere MQ .
3. Execute svcutil a partir de um prompt de comandos, por exemplo:

```
svcutil /language:C# /r:<installlocation>\bin\IBM.XMS.WCF.dll
                /config:app.config http://localhost:8000/IBM.XMS.WCF/samples
```

4. Copie os arquivos `app.config` e `YourService.cs` gerados no projeto do cliente Microsoft Visual Studio.

Como proceder a seguir

Se os metadados de serviços não puderem ser recuperados diretamente, svcutil poderá ser usada para gerar os arquivos de cliente a partir de wsdl. Para obter informações adicionais, consulte: "Gerando um Proxy de Cliente do WCF e Arquivos de Configuração de Aplicativo Usando a Ferramenta svcutil com WSDL" na página 621

Gerando um Proxy de Cliente do WCF e Arquivos de Configuração de Aplicativo Usando a Ferramenta svcutil com WSDL

Instruções para gerar clientes WCF a partir do WSDL se os metadados do serviço está indisponível.

Se não for possível recuperar os metadados do serviço diretamente para gerar um cliente a partir dos metadados de um serviço em execução, svcutil poderá ser usado para gerar os arquivos de cliente a partir do WSDL. As modificações a seguir devem ser feitas no WSDL para especificar que o canal customizado do WebSphere MQ deve ser usado:

1. Inclua as seguintes definições de espaço de nomes e informações de política:

```
<wsdl:definitions
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">

        <wsp:Policy wsu:Id="CustomBinding_IWMQSampleContract_policy">
            <wsp:ExactlyOne>
                <wsp:All>
                    <xms:xms xmlns:xms="http://sample.schemas.ibm.com/policy/xms" />
                </wsp:All>
            </wsp:ExactlyOne>
        </wsp:Policy>

...

</wsdl:definitions>
```

2. Modifique a seção de ligações para fazer referência à nova seção de política e remova qualquer definição de `transport` do elemento de ligação subjacente:

```
<wsdl:definitions ...>

        <wsdl:binding ...>
                <wsp:PolicyReference URI="#CustomerBinding_IWMQSampleContract_policy" />
                <[soap]:binding ... transport="" />
                ...
        </wsdl:binding>
</wsdl:definitions>
```

3. Execute svcutil a partir de um prompt de comandos, por exemplo:

```
svcutil /language:C# /r:MQ_INSTALLATION_PATH\bin\IBM.XMS.WCF.dll
        /config:app.config
MQ_INSTALLATION_PATH\src\samples\WMQAxis\default\service\soap.server.stockQuoteAxis_Wmq.wsdl
```

   Em que *MQ_INSTALLATION_PATH* é o diretório de instalação do WebSphere MQ

Construindo Aplicativos Clientes WCF Usando um Proxy de Cliente com um Arquivo de Configuração de Aplicativo

Antes de começar

Crie ou edite um arquivo de configuração de aplicativo para o cliente, conforme descrito em: “Criando um Canal Customizado do WCF Administrativamente Fornecendo Informações de Ligação e Terminal em um Arquivo de Configuração do Aplicativo” na página 612

Sobre esta tarefa

Instancie e abra uma instância do proxy de cliente. O parâmetro passado para o proxy gerado deve ser o mesmo que o nome de terminal especificado no arquivo de configuração de cliente, por exemplo `Endpoint_WMQ`:

```
MyClientProxy myClient = new MyClientProxy("Endpoint_WMQ");
            try {
                   myClient.myMethod("HelloWorld!");
                   myClient.Close();
            }
            catch (TimeoutException e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
            catch (CommunicationException e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
            catch (Exception e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
```

Construindo Aplicativos Cliente WCF Usando um Proxy de Cliente com Configuração Programática

Antes de começar

1. Inclua uma referência ao arquivo `IBM.XMS.WCF.dll` do canal customizado no projeto. O `IBM.XMS.WCF.dll` está no diretório *`WMQInstallDir`*`\bin` em que *WMQInstallDir* é o diretório no qual o WebSphere MQ 7 está instalado.
2. Inclua uma instrução *using* no namespace `IBM.XMS.WCF`, por exemplo: `using IBM.XMS.WCF`
3. Crie uma instância do elemento de ligação th' e terminal do canal conforme descrito em: “Criando um Canal Customizado do WCF Fornecendo Informações de Ligação e Terminal Programaticamente” na página 614

Sobre esta tarefa

Se as mudanças para as propriedades de ligação do canal forem necessárias, conclua as seguintes etapas:

1. Inclua as propriedades de ligação em `transportBindingElement` conforme mostrado na figura a seguir:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
EndpointAddress address =
  new EndpointAddress("jms:/queue?destination=SampleQ@QM1&connectionFactory=

connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
```

2. Crie o proxy de cliente conforme mostrado na figura a seguir, em que *binding* e *endpoint address* são a ligação e o endereço do terminal configurados na etapa “1” na página 622 e transmitidos:

```
 MyClientProxy myClient = new MyClientProxy(binding, endpoint address);
            try {
                   myClient.myMethod("HelloWorld!");
                   myClient.Close();
            }
```

```
            catch (TimeoutException e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
            catch (CommunicationException e) {
                Console.Out.WriteLine(e);
                                  myClient.Abort();
            }
            catch (Exception e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
```

Usando as Amostras do WCF

As amostras do Windows Communication Foundation (WCF) fornecem alguns exemplos simples de como o canal customizado WebSphere MQ pode ser usado.

Para construir os projetos de amostra, o Microsoft .NET 3.5 SDK ou Microsoft Visual Studio 2008 é necessário.

Amostra do WCF de Cliente e Servidor Unidirecional Simples

Esta amostra demonstra o canal customizado do WebSphere MQ sendo usado para iniciar um serviço Windows Communication Foundation (WCF) a partir de um cliente WCF usando um formato de canal unidirecional.

Sobre esta tarefa

O serviço implementa um único método que envia uma sequência para o console. O cliente foi gerado usando a ferramenta `svcutil` para recuperar os metadados de serviço de um terminal HTTP exposto separadamente conforme descrito em "Gerando um Proxy de Cliente do WCF e Arquivos de Configuração do Aplicativo Usando a Ferramenta svcutil com Metadados de um Serviço em Execução" na página 620

A amostra foi configurada com nomes de recurso específicos conforme descrito no procedimento a seguir. Se precisar mudar os nomes do recurso, então, deverá também mudar o valor correspondente no aplicativo cliente no arquivo *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\app.config` e no aplicativo de serviço no arquivo *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\TestServices.cs`, em que *MQ_INSTALLATION_PATH* é o diretório de instalação para o IBM WebSphere MQ. Para obter mais informações sobre como formatar o URI do terminal JMS, consulte *WebSphere MQ Transport for SOAP* na documentação do produto WebSphere MQ Se for necessário modificar a solução e a origem de amostra, será necessário um IDE, por exemplo, Microsoft Visual Studio 8 ou superior

Procedimento

1. Crie um gerenciador de filas chamado *QM1*
2. Crie um destino de fila chamado *SampleQ*
3. Inicie o serviço de forma que o listener esteja aguardando mensagens: execute o arquivo *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\bin\Release\TestService.exe`, em que *MQ_INSTALLATION_PATH* é o diretório de instalação para o IBM WebSphere MQ.
4. Execute o cliente uma vez: execute o arquivo *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\bin\Release\TestClient.exe`, em que *MQ_INSTALLATION_PATH* é o diretório de instalação para o IBM WebSphere MQ.

   O aplicativo cliente faz loop cinco vezes enviando cinco mensagens para *SampleQ*

Resultados

O aplicativo de serviço obtém as mensagens de *SampleQ* e exibe `Hello World` na tela cinco vezes.

Como proceder a seguir

Amostra do WCF do Cliente e Servidor de Pedido-Resposta Simples

Essa amostra demonstra o canal customizado do WebSphere MQ sendo usado para iniciar um serviço Windows Communication Foundation (WCF) a partir de um cliente WCF usando um formato de canal de solicitação-resposta.

Sobre esta tarefa

Este serviço fonece alguns métodos calculadores simples para adicionar e subtrair dois números e, em seguida, retornar o resultado. O cliente foi gerado usando a ferramenta `svcutil` para recuperar os metadados de serviço de um terminal HTTP exposto separadamente conforme descrito em "Gerando um Proxy de Cliente do WCF e Arquivos de Configuração do Aplicativo Usando a Ferramenta svcutil com Metadados de um Serviço em Execução" na página 620

A amostra foi configurada com nomes de recurso específicos como no procedimento descrito a seguir. Se precisar alterar os nomes de recursos, também será necessário alterar o valor correspondente no aplicativo cliente no arquivo *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\client\app.config` e no aplicativo de serviço no arquivo *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\service\RequestReplyService.cs` , em que *MQ_INSTALLATION_PATH* é o diretório de instalação para o WebSphere MQ. Para obter mais informações sobre como formatar o URI do terminal JMS, consulte *WebSphere MQ Transport for SOAP* na documentação do produto WebSphere MQ Se for necessário modificar a solução e a origem de amostra, será necessário um IDE, por exemplo, Microsoft Visual Studio 8 ou superior

Procedimento

1. Crie um gerenciador de filas chamado *QM1*
2. Crie um destino de fila chamado *SampleQ*
3. Crie um destino de fila chamado *SampleReplyQ*
4. Inicie o serviço para que o listener esteja esperando mensagens: Execute o arquivo *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\service\bin\Release\SimpleRequestReply_Service.exe` , em que *MQ_INSTALLATION_PATH* é o diretório de instalação para o WebSphere MQ.
5. Execute o cliente uma vez: Execute o arquivo *MQ_INSTALLATION_PATH*`\Tools\wcf\samples\WCF\requestreply\client\bin\Release\SimpleRequestReply_Client.exe` , em que *MQ_INSTALLATION_PATH* é o diretório de instalação do WebSphere MQ.

Resultados

Quando o cliente tiver sido executado, o seguinte processo será iniciado e repetirá quatro vezes para que um total de cinco mensagens serão enviadas cada maneira:

1. O cliente coloca uma mensagem de pedido em *SampleQ* e aguarda uma resposta.
2. O serviço obtém a mensagem de pedido de *SampleQ*.
3. O serviço adiciona e subtrai alguns valores usando o conteúdo da mensagem.
4. O serviço então coloca os resultados em uma mensagem em *SampleReplyQ* e aguarda o cliente colocar uma nova mensagem.
5. O cliente obtém a mensagem de *SampleReplyQ* e exibe os resultados na tela.

Como proceder a seguir

Cliente WCF para um serviço .NET hospedado pela amostra WebSphere MQ

Aplicativos clientes de amostra e aplicativos do proxy de serviço de amostra são fornecidos para .NET e para Java. As amostras são baseadas em um serviço de Cota de Ações que obtém um pedido para uma cota de ações e, em seguida, fornece a cota de ações.

Antes de começar

A amostra requer que o ambiente de hospedagem do serviço .NET SOAP sobre JMS esteja instalado e configurado corretamente no WebSphere MQ e esteja acessível a partir de um gerenciador de filas locais Para obter informações sobre como instalar e configurar o ambiente, consulte: "Instalando o transporte da web do WebSphere MQ para SOAP" na página 969

Quando o ambiente de hosting do serviço .NET SOAP sobre JMS estiver instalado e configurado corretamente no WebSphere MQ e estiver acessível a partir de um gerenciador de fila local, as etapas de configuração adicionais deverão ser concluídas

1. Configure a variável de ambiente WMQSOAP_HOME para o diretório de instalação do WebSphere MQ , por exemplo: `C:\Program Files\IBM\WebSphere MQ` .
2. Assegure que o compilador Java `javac` esteja disponível e no PATH.
3. Copie o arquivo `axis.jar` a partir do diretório `prereqs/axis` do CD de instalação do WebSphere para o diretório de produção do WebSphere MQ , por exemplo: `C:\Program Files\IBM\WebSphere MQ\java\lib\soap`
4. Inclua no PATH: *MQ_INSTALLATION_PATH*`\Java\lib` em que *MQ_INSTALLATION_PATH* representa o diretório no qual o WebSphere MQ está instalado, por exemplo: `C:\Program Files\IBM\WebSphere MQ`
5. Assegure-se de que o local de .NET esteja especificado corretamente em *MQ_INSTALLATION_PATH*`\bin\amqwcallWSDL.cmd` , em que *MQ_INSTALLATION_PATH* representa o diretório no qual o WebSphere MQ está instalado, por exemplo: `C:\Program Files\IBM\WebSphere MQ`. O local de .NET pode ser especificado por exemplo: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Quando as etapas anteriores forem concluídas, teste e execute o serviço:

1. Navegue até seu diretório de trabalho SOAP sobre JMS.
2. Insira um dos comandos a seguir para executar o teste de verificação e deixe o listener de serviço em execução:
   - Para .NET: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt dotnet hold` em que *MQ_INSTALLATION_PATH* representa o diretório no qual o WebSphere MQ está instalado.
   - Para AXIS:*MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt Dotnet2AxisClient hold` em que *MQ_INSTALLATION_PATH* representa o diretório no qual o WebSphere MQ está instalado.

O argumento `hold` mantém os listeners em execução após o teste ser concluído.

Caso sejam relatados erros durante esta configuração, é possível remover todas as mudanças para que o procedimento possa ser reiniciado da seguinte maneira:

1. Exclua o diretório SOAP sobre JMS gerado.
2. Exclua o gerenciador de filas.

Sobre esta tarefa

Esta amostra demonstra uma conexão de um cliente WCF com o serviço de amostra .NET SOAP sobre JMS fornecido no WebSphere MQ usando um formato de canal unidirecional.. O serviço implementa um exemplo simples de StockQuote, que envia uma sequência de texto para o console.

O cliente foi gerado usando o WSDL para gerar arquivos de clientes conforme descrito em "Gerando um Proxy de Cliente do WCF e Arquivos de Configuração de Aplicativo Usando a Ferramenta svcutil com WSDL" na página 621

A amostra foi configurada com nomes de recurso específicos conforme descrito no procedimento a seguir. Se você precisar mudar os nomes de recurso, também deverá mudar o valor correspondente no aplicativo cliente no arquivo *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQNET\default\client\app.config e no aplicativo de serviço no arquivo *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQNET\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl , em que *MQ_INSTALLATION_PATH* representa o diretório de instalação para WebSphere MQ. Para obter mais informações sobre como formatar o URI do terminal JMS, consulte *WebSphere MQ Transport for SOAP* na documentação do produto WebSphere MQ

Procedimento

Execute o cliente uma vez: Execute o arquivo *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQNET\default\client\bin\Release\TestClient.exe , em que *MQ_INSTALLATION_PATH* representa o diretório de instalação para WebSphere MQ.

O aplicativo cliente faz loop cinco vezes enviando cinco mensagens à fila de amostra.

Resultados

O aplicativo de serviço recebe as mensagens da fila de amostra e exibe `Hello World` cinco vezes na tela.

Cliente WCF para um serviço Java Axis hospedado pela amostra WebSphere MQ

Aplicativos clientes de amostra e aplicativos do proxy de serviço de amostra são fornecidos para Java e para .NET. As amostras são baseadas em um serviço de Cota de Ações que obtém um pedido para uma cota de ações e, em seguida, fornece a cota de ações.

Antes de começar

Essa amostra requer que o ambiente de hosting do serviço .NET SOAP sobre JMS esteja corretamente instalado e configurado no WebSphere MQ e esteja acessível a partir de um gerenciador de filas locais Para obter informações sobre como instalar e configurar o ambiente, consulte: “Instalando o transporte da web do WebSphere MQ para SOAP” na página 969

Quando o ambiente de hosting do serviço .NET SOAP sobre JMS estiver instalado e configurado corretamente no WebSphere MQ e estiver acessível a partir de um gerenciador de fila local, as etapas de configuração adicionais deverão ser concluídas

1. Configure a variável de ambiente WMQSOAP_HOME para o diretório de instalação do WebSphere MQ , por exemplo: `C:\Program Files\IBM\WebSphere MQ` .
2. Assegure que o compilador Java `javac` esteja disponível e no PATH.
3. Copie o arquivo `axis.jar` do diretório `prereqs/axis` do CD de instalação do WebSphere para o diretório de instalação do WebSphere MQ .
4. Inclua no PATH: *MQ_INSTALLATION_PATH*\Java\lib em que *MQ_INSTALLATION_PATH* representa o diretório no qual o WebSphere MQ está instalado, por exemplo: `C:\Program Files\IBM\WebSphere MQ`
5. Assegure-se de que o local de .NET esteja especificado corretamente em *MQ_INSTALLATION_PATH*\bin\amqwcallWSDL.cmd , em que *MQ_INSTALLATION_PATH* representa o diretório no qual o WebSphere MQ está instalado, por exemplo: `C:\Program Files\IBM\WebSphere MQ`. O local de .NET pode ser especificado por exemplo: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Quando as etapas anteriores forem concluídas, teste e execute o serviço:

1. Navegue até seu diretório de trabalho SOAP sobre JMS.
2. Insira um dos comandos a seguir para executar o teste de verificação e deixe o listener de serviço em execução:

- Para .NET: *MQ_INSTALLATION_PATH*\Tools\soap\samples\runivt dotnet hold em que *MQ_INSTALLATION_PATH* representa o diretório no qual o WebSphere MQ está instalado.
- Para AXIS:*MQ_INSTALLATION_PATH*\Tools\soap\samples\runivt Dotnet2AxisClient hold em que *MQ_INSTALLATION_PATH* representa o diretório no qual o WebSphere MQ está instalado.

O argumento hold mantém os listeners em execução após o teste ser concluído.

Caso sejam relatados erros durante esta configuração, é possível remover todas as mudanças de modo que o procedimento seja reiniciado da seguinte maneira:

1. Exclua o diretório SOAP sobre JMS gerado.
2. Exclua o gerenciador de filas.

Sobre esta tarefa

A amostra demonstra uma conexão de um cliente WCF com o serviço de amostra Axis Java SOAP over JMS fornecido no WebSphere MQ usando um formato de canal unidirecional. O serviço implementa um exemplo de StockQuote simples, que envia uma sequência de texto para um arquivo que é salvo no diretório atual.

O cliente foi gerado usando o WSDL para gerar arquivos de clientes conforme descrito em "Gerando um Proxy de Cliente do WCF e Arquivos de Configuração de Aplicativo Usando a Ferramenta svcutil com WSDL" na página 621

A amostra foi configurada com nomes de recurso específicos conforme descrito neste parágrafo. Se você precisar alterar os nomes dos recursos, também deverá alterar o valor correspondente no aplicativo cliente no arquivo *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQAxis\default\client\app.config e no aplicativo de serviço no arquivo *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQAxis\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl , em que *MQ_INSTALLATION_PATH* representa o diretório de instalação para WebSphere MQ.

Procedimento

Execute o cliente uma vez: Execute o arquivo *MQ_INSTALLATION_PATH*\tools\wcf\samples\WMQAxis\default\client\bin\Release\TestClient.exe , em que *MQ_INSTALLATION_PATH* representa o diretório de instalação para WebSphere MQ.

O aplicativo cliente faz loop cinco vezes enviando cinco mensagens à fila de amostra.

Resultados

O aplicativo de serviço recebe as mensagens da fila de amostra e inclui Hello World cinco vezes em um arquivo no diretório atual.

Referências relacionadas

"Manipulando Diferentes Nomes de Elemento de Resposta do SOAP" na página 635

O WCF espera que o nome de um valor retornado esteja em um formato específico por padrão, mas um serviço pode não retornar um elemento com seu nome no formato esperado.

Cliente WCF para serviço Java hospedado pela amostra do WebSphere Application Server

Aplicativos clientes de amostra e aplicativos proxy de serviço de amostra são fornecidos para o WebSphere Application Server (WAS) 6. Um pedido de serviço pedido-resposta também é fornecido.

Antes de começar

Essa amostra requer que a seguinte configuração do WebSphere MQ seja usada:

Tabela 77. WebSphere MQ configuração necessária	
Object	**Nome necessário**
Gerenciador de Filas	QM1
Fila local	HelloWorld
Fila local	HelloWorldReply

Essa amostra também requer que um ambiente de hospedagem do WebSphere Application Server V6 esteja instalado e configurado corretamente O WebSphere Application Server V6 usa uma conexão de modo de ligações para conectar ao WebSphere MQ por padrão. Portanto, o WebSphere Application Server V6 deve ser instalado na mesma máquina que o gerenciador de filas

Depois que o ambiente do WAS for configurado, as seguintes etapas de configuração adicionais deverão ser concluídas:

1. Crie os seguintes objetos JNDI no repositório JNDI do WebSphere Application Server:
   a. Um destino de fila JMS chamado `HelloWorld`
      - Configure o nome JNDI como `jms/HelloWorld`
      - Configure o nome da fila como `HelloWorld`
   b. Um factory de conexão da fila JMS chamado `HelloWorldQCF`
      - Configure o nome JNDI como `jms/HelloWorldQCF`
      - Configure o nome do gerenciador de filas como `QM1`
   c. Um factory de conexão da fila JMS chamado `WebServicesReplyQCF`
      - Configure o nome JNDI como `jms/WebServicesReplyQCF`
      - Configure o nome do gerenciador de filas como `QM1`
2. Crie uma Porta do Listener de Mensagens chamada `HelloWorldPort` no WebSphere Application Server com a seguinte configuração:
   - Configure o nome JNDI do connection factory como `jms/HelloWorldQCF`
   - Configure o nome JNDI de destino como `jms/HelloWorld`
3. Instale o aplicativo `HelloWorldEJBEAR.ear` de serviço da web em seu WebSphere Application Server conforme a seguir:
   a. Clique em **Aplicativos** > **Novo aplicativo** > **Novo aplicativo corporativo**.
   b. Navegue até *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear` , em que *MQ_INSTALLATION_PATH* é o diretório de instalação do WebSphere MQ
   c. Não altere nenhuma das opções padrão no assistente e reinicie o servidor de aplicativos depois que o aplicativo tiver sido instalado.

Quando a configuração do WAS tiver sido concluída, teste o serviço executando-o uma vez:

1. Navegue até seu diretório de trabalho Soap sobre JMS.
2. Insira este comando para executar a amostra: *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\TestClient.exe` em que *MQ_INSTALLATION_PATH* é o diretório de instalação do WebSphere MQ.

Sobre esta tarefa

A amostra demonstra uma conexão de um cliente WCF com o serviço de amostra SOAP sobre JMS do WebSphere Application Server fornecido nas amostras WCF incluídas no WebSphere MQ V7, usando um formato de canal de solicitação-resposta. O fluxo de mensagens entre o WCF e o WebSphere Application Server usando WebSphere MQ filas. O serviço implementa o método `HelloWorld(...)` , que toma uma sequência e retorna uma saudação para o cliente

O cliente foi gerado usando a ferramenta svcutil para recuperar os metadados de serviço a partir de um terminal HTTP exposto separadamente conforme descrito em “Gerando um Proxy de Cliente do WCF e Arquivos de Configuração do Aplicativo Usando a Ferramenta svcutil com Metadados de um Serviço em Execução” na página 620

A amostra foi configurada com nomes de recurso específicos conforme descrito no procedimento a seguir. Se você precisar alterar os nomes de recursos, também deverá alterar o valor correspondente no aplicativo cliente no arquivo *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\default\client\app.config` e no aplicativo de serviço no *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear` em que *MQ_INSTALLATION_PATH* é o diretório de instalação do WebSphere MQ. Para obter mais informações sobre como formatar o URI do terminal JMS do, consulte Sintaxe do URI e Parâmetros para Implementação de Serviço da Web .

O serviço e o cliente são baseados no serviço e no cliente descritos no IBM Artigo do Desenvolvedor *Construindo um Serviço da Web JMS Usando SOAP sobre JMS e WebSphere Studio* Se desejar saber mais sobre como desenvolver serviços da web SOAP sobre JMS que são compatíveis com o canal customizado do WebSphere MQ WCF, o artigo relevante poderá ser localizado em: https://www.ibm.com/developerworks/websphere/library/techarticles/0402_du/0402_du.html.

Procedimento

Execute o cliente uma vez: Execute o arquivo *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\default\client\bin\Release\TestClient.exe` , em que *MQ_INSTALLATION_PATH* é o diretório de instalação do WebSphere MQ.

O aplicativo cliente inicia os dois métodos de serviço ao mesmo tempo, enviando duas mensagens para a fila de amostra.

Resultados

O aplicativo de serviço recebe as mensagens da fila de amostra e fornece uma resposta para a chamada de método `HelloWorld(...)`, que o aplicativo cliente emite para o console.

Determinação de problema no canal customizado do WCF para WebSphere MQ

É possível usar o rastreio de WebSphere MQ para coletar informações detalhadas sobre o que várias partes do WebSphere MQ estão fazendo. Ao usar o Windows Communication Foundation (WCF), uma saída de rastreio separada é gerada para o rastreio de canal customizado do WCF integrado ao rastreio de infraestrutura do WCF Microsoft .

Ativar totalmente o rastreio para o canal customizado do WCF produz dois arquivos de saída:

1. O rastreio do canal customizado do WCF integrado ao rastreio de infraestrutura do WCF Microsoft .
2. O rastreio do canal customizado do WCF integrado com o XMS .NET.

Tendo duas saídas de rastreio, problemas podem ser controlados em cada interface usando as ferramentas apropriadas, por exemplo:

- Determinação de problema WCF usando o conjunto de ferramentas Microsoft adequado.
- Problemas do cliente MQI do WebSphere MQ usando o formato de rastreio XMS

Para simplificar a ativação de rastreio, as pilhas de rastreio do .NET 3 TraceSource e XMS .NET são controladas usando uma única interface conforme descrito em: “Configuração de rastreio do WCF e nomes de arquivos de rastreio” na página 630.

Hierarquia de Exceção do Canal Customizado do WCF

Os tipos de exceções lançados pelo canal customizado são consistentes com o WCF e são, geralmente, TimeoutException ou CommunicationException (ou uma subclasse de CommunicationException).

Detalhes adicionais da condição de erro, onde disponíveis, são fornecidos usando exceções vinculadas ou internas. As exceções a seguir são exemplos típicos e cada camada na arquitetura do canal contribui com uma exceção vinculada adicional, por exemplo; CommunicationsException possui uma XMSException vinculada, que possui uma MQException vinculada:

1. System.serviceModel.CommunicationsExceptions
2. IBM.XMS.XMSException
3. IBM.WMQ.MQException

As informações chave são capturadas e fornecidas na coleta de dados da CommunicationException mais alta na hierarquia. Esta captura e o fornecimento de dados evitam a necessidade de os aplicativos se vincularem a cada camada na arquitetura do canal para interrogar as exceções vinculadas e as informações adicionais que elas podem conter. Os nomes de chaves a seguir são definidos:

- IBM.XMS.WCF.ErrorCode: O código da mensagem de erro da exceção do canal customizado atual..
- IBM.XMS.ErrorCode: A mensagem de erro da primeira exceção XMS na pilha.
- IBM.WMQ.ReasonCode: O código de razão subjacente do WebSphere MQ .
- IBM.WMQ.CompletionCode: O código de conclusão subjacente do WebSphere MQ .

Configuração de rastreio do WCF e nomes de arquivos de rastreio

Quando o rastreio está totalmente ativado, ele produz dois arquivos de saída, um para diagnosticar problemas WCF e um arquivo detalhado para material de diagnóstico de rastreio interno. Para simplificar a ativação do rastreio, as pilhas de rastreio .NET 3 TraceSource e XMS .NET usam uma interface única.

Dois métodos de rastreio diferentes estão disponíveis para o canal customizado do WCF, os dois métodos de rastreio são ativados independentemente ou juntos. Cada método produz seu próprio arquivo de rastreio, portanto, quando ambos os métodos de rastreio foram ativados, dois arquivos de saída de rastreio são gerados.

Para manter a configuração e ativação o mais simples possível, a mesma interface é usada para controlar ambos os métodos de rastreio. O arquivo `app.config` deve ser editado para incluir a configuração de rastreio relevante conforme descrito na seção a seguir. Os usuários podem, então, incluir suas próprias seções equivalentes para combinar a saída com o rastreio de seus próprios aplicativos.

O rastreio do canal customizado do WCF não é ativado por padrão. É necessário, primeiro, criar um listener de rastreio e, em seguida, configurar o nível de rastreio necessário para a origem de rastreio selecionada no arquivo `app.config`.

Configurando o Canal Customizado do WCF com o Rastreio de Infraestrutura do WCF

Inclua a seção de código a seguir na seção `<system.diagnostics><sources>` no arquivo `app.config`:

```
<source name="IBM.XMS.WCF" switchValue="Verbose,ActivityTracing">
    <listeners>
        <remove name="Default"/>
        <add name="NewListener"/>
    </listeners>
</source>
```

A parte anterior do código faz o rastreio de canal usando o .NET 3 TraceSource. Todas as chamadas dos arquivos de configuração associados aos arquivos executáveis são controladas por esta parte do código.

Configurando o Canal Customizado do WCF com o Rastreio de XMS .NET

A configuração do rastreio do XMS .NET requer que você inclua uma seção de código na seção `<system.diagnostics><sources>` no arquivo `app.config` No entanto, a parte de código é incluída no elemento `<source>` extensível mostrado na seção Configurando o canal customizado do WCF com o

rastreio de infraestrutura do WCF. Portanto, embora o código de rastreio de infraestrutura do WCF deva estar presente para que o rastreio XMS .NET funcione, o rastreio de infraestrutura do WCF poderá ser desativado se não for necessário, conforme descrito na seção Ativando o rastreio do WCF .

```
<source name="IBM.XMS.WCF" switchValue="Verbose, ActivityTracing"
 xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path"
 xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced">
    <listeners>
        <remove name="Default"/>
        <add name="NewListener"/>
    </listeners>
</source>
```

Variáveis de Configuração de Rastreio do WCF

Tabela 78. Variáveis de Configuração de Rastreio do WCF

Variável	**Descrição**
nome	Especifique o nome como: `IBM.XMS.WCF`
switchValue	switchValue controla o nível de rastreio. Quando switchValue é configurado como `Off`, TraceSource da infraestrutura do WCF não é gerada. Qualquer outro valor, como `Verbose`, gera TraceSource. Para obter informações detalhadas do nível de rastreio da Microsoft, consulte a documentação do WCF ou acesse a página da web Microsoft WCF Tracing: https://msdn.microsoft.com/en-us/library/ms733025(vs.85).aspx

xmsTraceSpecification=*ComponentName=type=state*	*ComponentName* é o nome da classe que você deseja rastrear. É possível usar um caractere curinga * neste nome. Por exemplo:
	`*=all=enabled`
	especifica que você deseja rastrear todas as classes, e
	`IBM.XMS.impl.*=all=enabled`
	especifica que você requer apenas o rastreio da API.
	type pode ser qualquer um dos seguintes tipos de rastreio:
	• all
	• debug
	• evento
	• EntryExit
	state pode ser ativado ou desativado.

xmsTraceFilePath="*filename*"	Se você não especificar um xmsTraceFilePath, ou se o xmsTraceFilePath estiver presente mas contiver uma sequência vazia, o arquivo de rastreio será colocado no diretório atual. Para armazenar o arquivo de rastreio em um diretório nomeado, especifique o nome de diretório no xmsTraceFilePath, por exemplo: `xmsTraceFilePath="c:\somepath"`

Tabela 78. Variáveis de Configuração de Rastreio do WCF (continuação)	
Variável	**Descrição**
xmsTraceFileSize="*size*"	O tamanho máximo permitido do arquivo de rastreio. Quando um arquivo atinge este tamanho, ele é arquivado e renomeado. O máximo padrão é 20 KB, que é especificado como: `xmsTraceFileSize="20000000".`
xmsTraceFileNumber="*number*"	O número de arquivos de rastreio que devem ser retidos. O padrão é 4 (um arquivo ativo e três arquivos de archive). O número mínimo permitido é dois.

xmsTraceFormat="*format*"	Existem dois níveis de xmsTraceFormat: `basic` e `advanced`. O formato de rastreio padrão é básico se você não especificar um xmsTraceFormat ou se o xmsTraceFormat estiver presente, mas contiver uma sequência de caracteres vazia. Os arquivos de rastreio são produzidos neste formato se você especificar:
	`xmsTraceFormat="basic"`
	Se você requerer um rastreio que seja compatível com as ferramentas do analisador de rastreio, deverá especificar:
	`traceFormat="advanced"`

Ativando o Rastreio do WCF

Existem quatro combinações para ativar e desativar os dois métodos de rastreio diferentes. As quatro combinações requerem a edição dos valores das seções de codificação descritas nas seções anteriores.

Há também uma variável de ambiente que pode ser configurada; para obter mais informações, consulte “Ativando o Rastreio WCF com a Variável de Ambiente WCF_TRACE_ON” na página 633

Esta tabela e os valores mostrados dependem de as partes do código demonstrado anteriormente já terem sido incluídas no arquivo `app.config`.

Tabela 79. Combinações de Ativação de Rastreio de WCF.		
Tipo de Rastreio	**Valor Alterado**	**exemplo**
Rastreio XMS ativado. TraceSource de WCF ativada	O `switchValue` não é configurado como `Off`	`<source name="IBM.XMS.WCF" switchValue="Verbose, ActivityTracing" xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`

Tabela 79. Combinações de Ativação de Rastreio de WCF. (continuação)

Tipo de Rastreio	Valor Alterado	exemplo
Rastreio XMS ativado. TraceSource de WCF desativada	O `switchValue` está configurado como `Off` e uma `xmsTraceSpecification` foi fornecida	`<source name="IBM.XMS.WCF" switchValue="Off, ActivityTracing" xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`
Rastreio de XMS desativado. TraceSource de WCF ativada	Existem duas maneiras de atingir este resultado: • A variável `switchValue` não está configurada como `Off` e uma `xmsTraceSpecification` não foi incluída • A variável `switchValue` não está configurada como `Off` e o `xmsTraceSpecification` foi configurado como `disabled`	`<source name="IBM.XMS.WCF" switchValue="Verbose, ActivityTracing" xmsTraceSpecification="*=all=disabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`
Rastreio de XMS desativado. TraceSource de WCF desativada	Há três maneiras de alcançar este resultado: • Nenhum elemento `<source>` no arquivo `app.config` • A variável `switchValue` está configurada como `Off` e um `xmsTraceSpecification` não foi incluído • A variável `switchValue` está configurada como `Off` e o `xmsTraceSpecification` foi configurado como `disabled`	`<source name="IBM.XMS.WCF" switchValue="Off, ActivityTracing" xmsTraceSpecification="*=all=disabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`

Ativando o Rastreio WCF com a Variável de Ambiente WCF_TRACE_ON

Assim como os métodos anteriores descritos para ativar o rastreio WCF, o rastreio XMS .NET também pode ser ativado usando a variável de ambiente WCF_TRACE_ON.

Configurar a variável de ambiente WCF_TRACE_ON como qualquer valor não nulo é o equivalente a configurar `xmstraceSpecification` como `*=all=enabled`, por exemplo: "`set WCF_TRACE_ON=true`"

Porém, se `xmstraceSpecification` for explicitamente configurado no arquivo `app.config`, a variável de ambiente WCF_TRACE_ON será substituída.

arquivos de saída de rastreio WCF e nomes de arquivos

Os arquivos de rastreio XMS são, tradicionalmente, denominados usando o formato de nome base e ID do processo de: `xms_trace_`*`pid`*`.log`, em que *pid* é o ID do processo.

Como os arquivos de rastreio XMS ainda podem ser produzidos em paralelo com arquivos de rastreio do canal customizado do WCF, o rastreio do canal customizado do WCF integrado aos arquivos de saída de

rastreio de XMS .NET possuem o seguinte formato para evitar confusão: `wcfxms_trace_pid.log`, em que *pid* é o ID do processo.

O arquivo de saída de rastreio é criado no diretório de trabalho atual por padrão, mas este destino poderá ser redefinido, se necessário.

WCF XMS Primeira tecnologia de suporte de falha (FFST)

É possível coletar informações detalhadas sobre o que várias partes do código do WebSphere MQ estão fazendo usando o rastreio do WebSphere MQ . XMS FFST tem sua própria configuração e arquivos de saída para o canal customizado do WCF.

Os arquivos de rastreio XMS FFST são tradicionalmente nomeados usando o nome base e o formato do ID do processo de: `xmsffdcpid_date.txt`, em que *pid* é o ID do processo e *date* é a hora e a data.

Como os arquivos de rastreio XMS FFST ainda podem ser produzidos em paralelo com os arquivos XMS FFST do canal customizado do WCF, os arquivos de saída XMS FFST do canal customizado têm o seguinte formato para evitar confusão: `wcffdcpid_date.txt`, em que *pid* é o ID do processo e *date* é o horário e a data.

Este arquivo de saída de rastreio é criado no diretório de trabalho atual por padrão, mas este destino pode ser redefinido se necessário.

O canal customizado WCF com o cabeçalho de rastreio XMS .NET é semelhante ao exemplo a seguir:

```
************* Start Display XMS WCF Environment *************
  Product Name  :- value
  WCF Version   :- value
  Level    :- value
*************  End Display XMS WCF Environment  *************
```

Os arquivos de rastreio FFST são formatados da maneira padrão, sem qualquer formatação específica para o canal customizado.

Informações da Versão do WCF

As informações da versão do WCF auxiliam na determinação de problema e estão incluídas nos metadados do conjunto do canal customizado.

O canal customizado do WebSphere MQ para metadados da versão do WCF pode ser recuperado de uma das três maneiras:

- Usando o utilitário dspmqver do WebSphere MQ Para obter informações sobre como usar o dspmqver, consulte: dspmqver
- Usando o diálogo de propriedades do Windows Explorer: no Windows Explorer, clique com o botão direito em **IBM.XMS.WCF.dll** > **Propriedades** > **Versão**..
- Nas informações do cabeçalho de qualquer um dos canais FFST ou arquivos de rastreio. Para obter mais informações sobre as informações do cabeçalho FFST , consulte: “WCF XMS Primeira tecnologia de suporte de falha (FFST)” na página 634

Sugestões e Dicas do WCF

As seguintes sugestões e dicas estão em ordem não significativa e podem ser incluídas quando novas versões da documentação forem liberadas. Elas são assuntos que podem economizar tempo se elas são relevantes para o trabalho que você está fazendo.

Externalizando Exceções do Host de Serviço do WCF

Para serviços hospedados usando o host de serviço do WCF, as exceções não manipuladas lançadas pelo serviço, internos do WCF ou pilha de canais não são externalizadas por padrão. Para ser informado sobre essas exceções, um manipulador de erros deve ser registrado.

O código a seguir fornece um exemplo de como definir o comportamento do serviço do manipulador de erros que pode ser aplicado como um atributo de um serviço:

```
using System.ServiceModel.Dispatcher;
using System.Collections.ObjectModel;
....
    public class ErrorHandlerBehaviorAttribute : Attribute, IServiceBehavior, IErrorHandler
    {
        //
        // IServiceBehavior Interface
        //
        public void AddBindingParameters(ServiceDescription serviceDescription,
           ServiceHostBase serviceHostBase, Collection<ServiceEndpoint> endpoints,
           BindingParameterCollection bindingParameters)
        {
        }
        public void ApplyDispatchBehavior(ServiceDescription serviceDescription,
            ServiceHostBase serviceHostBase)
        {
            foreach (ChannelDispatcher channelDispatcher in serviceHostBase.ChannelDispatchers)
{
                channelDispatcher.ErrorHandlers.Add(this);
            }
        }
        public void Validate(ServiceDescription serviceDescription, ServiceHostBase
serviceHostBase)
        {
        }

        //
        // IErrorHandler Interface
        //
        public bool HandleError(Exception e)
        {
            // Process the exception in the required way, in this case just outputting to the
console
            Console.Out.WriteLine(e);

            // Always return false to allow any other error handlers to run
            return false;
        }
        public void ProvideFault(Exception error, MessageVersion version, ref Message fault)
        {
        }
    }
```

Manipulando Diferentes Nomes de Elemento de Resposta do SOAP

O WCF espera que o nome de um valor retornado esteja em um formato específico por padrão, mas um serviço pode não retornar um elemento com seu nome no formato esperado.

O WCF tem a convenção de esperar que o valor retornado seja nomeado no seguinte formato: *methodName*`Result` em que *methodName* é o nome da operação de serviço. Por exemplo, para um serviço chamado `getQuote`, o WCF espera que a resposta seja chamada: `getQuoteResult`.

Porém, o serviço pode retornar um elemento com um nome que não esteja em conformidade com este formato.

Ao executar a ferramenta scvutil para gerar um cliente proxy, se o WSDL especificar um nome diferente, a interface do proxy incluirá parâmetros para instruir o WCF com o nome a procurar. Por exemplo:

```
[System.ServiceModel.OperationContractAttribute(Action = "", ReplyAction = "*")]
[System.ServiceModel.XmlSerializerFormatAttribute(Style =
System.ServiceModel.OperationFormatStyle.Rpc,
                                                  Use =
System.ServiceModel.OperationFormatUse.Encoded)]
[return: System.ServiceModel.MessageParameterAttribute(Name = "getQuoteReturn")]
float getQuote(string in0);
```

Se você criar sua própria interface (por exemplo, incluindo um método pedido-resposta em uma interface de proxy existente), será necessário assegurar a inclusão dos mesmos parâmetros na interface se o serviço retornar um nome diferente. Se você não fizer isso, o problema mais comum será que uma chamada para o método de serviço sempre retornará um valor nulo; se um objeto for retornado, o método retornará nulo, mas se um valor numérico como um número inteiro for retornado, o método retornará 0.

Usando C++

WebSphere MQ fornece classes C++ equivalentes a objetos WebSphere MQ e algumas classes adicionais equivalentes aos tipos de dados da matriz. Ele fornece inúmeros recursos não disponíveis por meio do MQI.

A partir do WebSphere MQ Versão 7.0, os aprimoramentos para as interfaces de programação do WebSphere MQ não serão aplicados às classes C + +.

O WebSphere MQ C++ fornece os seguintes recursos:

- Inicialização automática de estruturas de dados WebSphere MQ .
- Conexão do gerenciador de filas e abertura de fila just-in-time.
- Fechamento de fila e desconexão do gerenciador de filas implícitos.
- Transmissão e recebimento do cabeçalho de mensagens não entregues.
- Transmissão e recebimento do cabeçalho da ponte IMS
- Transmissão e recebimento de cabeçalho de mensagem de referência.
- Recebimento de mensagem do acionador.
- Transmissão e recebimento do cabeçalho da ponte do CICS
- Recebimento e transmissão de cabeçalho do trabalho.
- Definição de canal do cliente.

Os diagramas de classes Booch a seguir mostram que todas as classes são amplamente paralelas àquelas entidades WebSphere MQ no MQI processual (por exemplo, usando C) que possuem manipulações ou estruturas de dados. Todas as classes herdam da classe ImqError (consulte ImqError C++ class), o que permite que uma condição de erro seja associada a cada objeto

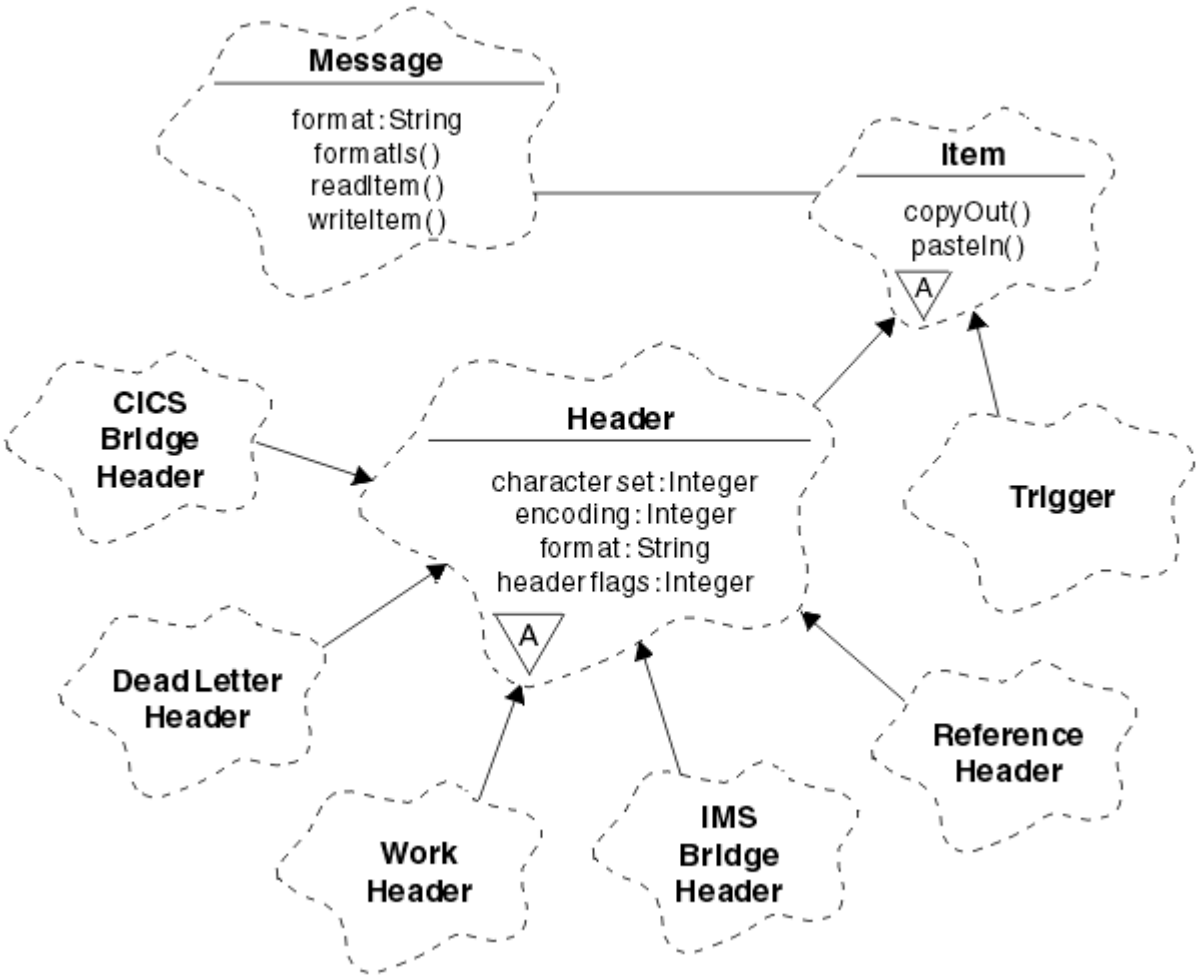


Figura 120. WebSphere MQ classes C++ (manipulação de itens)

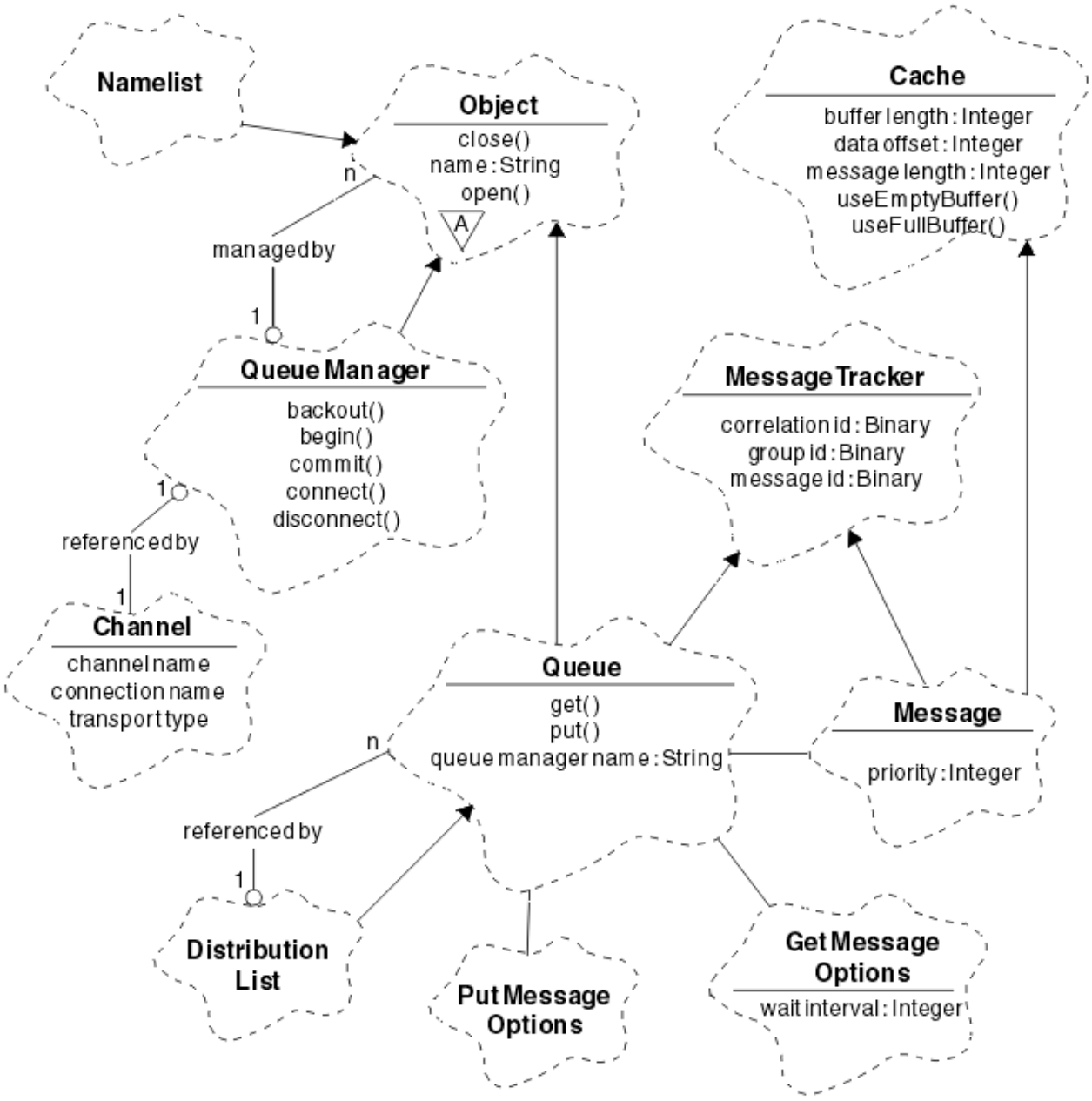


Figura 121. WebSphere MQ classes C++ (gerenciamento de filas)

Para interpretar os diagramas de classe Booch corretamente, esteja ciente das convenções a seguir:

- Os métodos e os atributos que vale a pena serem observados são mostrados abaixo do nome da *classe*.
- Um pequeno triângulo dentro de uma nuvem denota uma *classe abstrata*
- A*herança* é denotada por uma seta para a classe-pai.
- Uma linha não decorada entre as nuvens denota um *relacionamento cooperativo* entre as classes.
- Uma linha decorada com um número denota um *relacionamento referencial* entre duas classes. O número indica o número de objetos que podem participar de um determinado relacionamento a qualquer momento.

As classes e os tipos de dados a seguir são usados nas assinaturas de método C++ das classes de gerenciamento de fila (consulte Figura 121 na página 638) e as classes de manipulação de item (consulte Figura 120 na página 637):

- A classe ImqBinary (consulte ImqBinary C++ class), que encapsula matrizes de bytes como MQBYTE24.

- O tipo de dados ImqBoolean, que é definido como **typedef unsigned char ImqBoolean**.
- A classe ImqString (consulte ImqString C++ class), que encapsula matrizes de caracteres como MQCHAR64.

As entidades com estruturas de dados são incluídas nas classes de objeto apropriadas. Campos de estrutura de dados individuais (consulte Referência cruzada de C++ e MQI) são acessados com métodos.

As entidades com identificadores estão na hierarquia de classes ImqObject (consulte ImqObject C++ class) e fornecem interfaces encapsuladas para o MQI. Objetos dessas classes exibem o comportamento inteligente que pode reduzir o número de chamadas de método necessárias relativas à MQI processual. Por exemplo, é possível estabelecer e descartar as conexões do gerenciador de filas conforme necessário ou abrir uma fila com as opções a apropriadas e, em seguida, fechá-la.

A classe ImqMessage (consulte ImqMessage C++ class) encapsula a estrutura de dados MQMD e também age como um ponto de retenção para dados do usuário e *itens* (consulte “Lendo mensagens em C++” na página 648) fornecendo recursos de buffer em cache. É possível fornecer buffers de comprimento fixo para dados do usuário e usar o buffer muitas vezes. A quantia de dados presente no buffer pode variar de um uso para o próximo. Como alternativa, o sistema pode fornecer e gerenciar um buffer de comprimento flexível. O tamanho do buffer (a quantia disponível para recebimento de mensagens) e a quantia realmente usada (o número de bytes para transmissão ou o número de bytes realmente recebidos) se tornam considerações importantes.

Conceitos relacionados

Visão geral técnica

“Programas de amostra C++” na página 639
Quatro programas de amostra são fornecidos, para demonstrar a obtenção e a colocação de mensagens.

“contraprestações sobre linguagem C++” na página 643
Esta coleção de tópicos detalha os aspectos do uso da linguagem C++ e convenções que deve-se considerar ao gravar programas de aplicativos que usam um Message Queue Interface (MQI).

“Preparando dados da mensagem em C++” na página 647
Os dados da mensagem são preparados em um buffer, que pode ser fornecido pelo sistema ou aplicativo. Há vantagens para qualquer de um dos métodos. Exemplos de como usar um buffer são fornecidos.

“Decidindo qual linguagem de programação usar” na página 80
Use estas informações para descobrir sobre linguagens de programação e estruturas suportadas pelo IBM WebSphere MQe algumas considerações para usá-las.

“Desenvolvendo Aplicativos” na página 7
O IBM WebSphere MQ fornece várias maneiras nas quais é possível desenvolver aplicativos para enviar e receber mensagens necessárias para suportar seus processos de negócios. É possível também desenvolver os aplicativos para gerenciar os gerenciadores de fila e recursos relacionados.

Referências relacionadas

“Construindo programas C++ do WebSphere MQ” na página 654
A URL dos compiladores suportados é listada, juntamente com os comandos a serem usados para compilar, vincular e executar programas e amostras C++ nas plataformas WebSphere MQ .

Referência cruzada de C++ e MQI

Classes C++ do WebSphere MQ

Programas de amostra C++

Quatro programas de amostra são fornecidos, para demonstrar a obtenção e a colocação de mensagens.

Os programas de amostra são:

- HELLO WORLD (imqwrld.cpp)
- SPUT (imqsput.cpp)
- SGET (imqsget.cpp)
- DPUT (imqdput.cpp)

Os programas de amostra estão localizados nos diretórios mostrados em Tabela 80 na página 640.

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

Tabela 80. Local de programas de amostra

Ambiente	Diretório que contém a origem	Diretório que contém o construído programas
AIX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/ samp/bin/ia .
HP-UX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/ samp/bin/ah (consulte a nota “2” na página 640)
Solaris	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/as
Linux	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/
Windows	*MQ_INSTALLATION_PATH*\tools\cplus\ samples	*MQ_INSTALLATION_PATH*\tools\cplus\ samples\bin\vn (consulte a nota “3” na página 640)

Notes:

1. Os programas construídos usando o compilador ILE C++ para o IBM i estão na biblioteca QMQM. Os arquivos de inclusão estão em /QIBM/ProdData/mqm/inc.
2. Os programas construídos usando o compilador HP ANSI C++ estão localizados no diretório *MQ_INSTALLATION_PATH*/samp/bin/ah. Veja informações adicionais na publicação “Construindo programas C++ no HP-UX” na página 655.
3. Os programas construídos usando o Microsoft Visual Studio estão localizados em *MQ_INSTALLATION_PATH*\tools\cplus\samples\bin\vn. Para obter informações adicionais sobre esses compiladores, consulte “Construindo programas C++ no Windows” na página 660.

Programa de amostra HELLO WORLD (imqwrld.cpp)

Este programa de amostra C++ mostra como colocar e obter um datagrama regular (estrutura C) usando a classe ImqMessage.

Este programa mostra como colocar e obter um datagrama regular (estrutura C) usando a classe ImqMessage. Esta amostra usa algumas chamadas de método, aproveitando chamadas de método implícitas como **abrir**, **fechar** e **desconectar**.

Em todas as plataformas, exceto z/OS

Se você estiver usando uma conexão do servidor com o WebSphere MQ, siga um dos procedimentos a seguir:

- Para usar a fila padrão existente, SYSTEM.DEFAULT.LOCAL.QUEUE, execute o programa **imqwrlds** sem passar nenhum parâmetro
- Para usar uma fila temporária dinamicamente designada, execute **imqwrlds** transmitindo o nome da fila de modelo padrão, SYSTEM.DEFAULT.MODEL.QUEUE.

Se estiver usando uma conexão do cliente com o WebSphere MQ, siga um dos procedimentos a seguir:

- Configure a variável de ambiente MQSERVER (consulte MQSERVER para obter mais informações) e execute **imqwrldc** ou
- Execute **imqwrldc** passando como parâmetros **queue-name**, **queue-manager-name**e **channel-definition**, em que um **channel-definition** típico pode ser SYSTEM.DEF.SVRCONN/TCP/*hostname*(1414)

Código de Amostra

```
extern "C" {
#include <stdio.h>
}

#include <imqi.hpp> // WebSphere MQ C++

#define EXISTING_QUEUE "SYSTEM.DEFAULT.LOCAL.QUEUE"

#define BUFFER_SIZE 12

static char gpszHello[ BUFFER_SIZE ] = "Hello world" ;
int main ( int argc, char * * argv ) {
  ImqQueueManager manager ;
  int iReturnCode = 0 ;

  // Connect to the queue manager.
  if ( argc > 2 ) {
    manager.setName( argv[ 2 ] );
  }
  if ( manager.connect( ) ) {
    ImqQueue * pqueue = new ImqQueue ;
    ImqMessage * pmsg = new ImqMessage ;

    // Identify the queue which will hold the message.
    pqueue -> setConnectionReference( manager );
    if ( argc > 1 ) {
      pqueue -> setName( argv[ 1 ] );


      // The named queue can be a model queue, which will result in
      // the creation of a temporary dynamic queue, which will be
      // destroyed as soon as it is closed. Therefore we must ensure
      // that such a queue is not automatically closed and reopened.
      // We do this by setting open options which will avoid the need
      // for closure and reopening.
      pqueue -> setOpenOptions( MQOO_OUTPUT | MQOO_INPUT_SHARED |
                                MQOO_INQUIRE );
    } else {
      pqueue -> setName( EXISTING_QUEUE );

      // The existing queue is not a model queue, and will not be
      // destroyed by automatic closure and reopening. Therefore we
      // will let the open options be selected on an as-needed basis.
      // The queue will be opened implicitly with an output option
      // during the "put", and then implicitly closed and reopened
      // with the addition of an input option during the "get".
    }

    // Prepare a message containing the text "Hello world".
    pmsg -> useFullBuffer( gpszHello , BUFFER_SIZE );
    pmsg -> setFormat( MQFMT_STRING );

    // Place the message on the queue, using default put message
    // Options.
    // The queue will be automatically opened with an output option.
    if ( pqueue -> put( * pmsg ) ) {
      ImqString strQueue( pqueue -> name( ) );

      // Discover the name of the queue manager.
      ImqString strQueueManagerName( manager.name( ) );
      printf( "The queue manager name is %s.\n",
              (char *)strQueueManagerName );

      // Show the name of the queue.
      printf( "Message sent to %s.\n", (char *)strQueue );

      // Retrieve the data message just sent ("Hello world" expected)
      // from the queue, using default get message options. The queue
```

```
      // is automatically closed and reopened with an input option
      // if it is not already open with an input option. We get the
      // message just sent, rather than any other message on the
      // queue, because the "put" will have set the ID of the message
      // so, as we are using the same message object, the message ID
      // acts as in the message object, a filter which says that we
      // are interested in a message only if it has this
      // particular ID.

      if ( pqueue -> get( * pmsg ) ) {
        int iDataLength = pmsg -> dataLength( );

        // Show the text of the received message.
        printf( "Message of length %d received, ", iDataLength );

        if ( pmsg -> formatIs( MQFMT_STRING ) ) {
          char * pszText = pmsg -> bufferPointer( );

          // If the last character of data is a null, then we can
          // assume that the data can be interpreted as a text
          // string.
          if ( ! pszText[ iDataLength - 1 ] ) {
            printf( "text is \"%s\".\n", pszText );
          } else {
            printf( "no text.\n" );
          }

        } else {
          printf( "non-text message.\n" );
        }
      } else {
        printf( "ImqQueue::get failed with reason code %ld\n",
                pqueue -> reasonCode( ) );
        iReturnCode = (int)pqueue -> reasonCode( );
      }

    } else {
      printf( "ImqQueue::open/put failed with reason code %ld\n",
              pqueue -> reasonCode( ) );
      iReturnCode = (int)pqueue -> reasonCode( );
    }

    // Deletion of the queue will ensure that it is closed.
    // If the queue is dynamic then it will also be destroyed.
    delete pqueue ;
    delete pmsg ;

  } else {
    printf( "ImqQueueManager::connect failed with reason code %ld\n"
            manager.reasonCode( ) );
    iReturnCode = (int)manager.reasonCode( );
  }

  // Destruction of the queue manager ensures that it is
  // disconnected.  If the queue object were still available
  // and open (which it is not), the queue would be closed
  // prior to disconnection.

  return iReturnCode ;
}
```

Programas de amostra SPUT (imqsput.cpp) e SGET (imqsget.cpp)

Esses programas C++ colocam mensagens e recuperam as mensagens de uma fila nomeada.

Essas amostras mostram o uso das seguintes classes:

- ImqError (consulte ImqError Classe C++)
- ImqMessage (consulte ImqMessage Classe C++)
- ImqObject (veja ImqObject Classe C++)
- ImqQueue (consulte ImqQueue classe C++)
- ImqQueueManager (consulte ImqQueueManager C++ class)

Siga as instruções apropriadas para executar os programas.

Em todas as plataformas, exceto z/OS

1. Execute **imqsputs** *queue-name*.
2. Linhas de tipo de texto no console. Essas linhas são colocados como mensagens na fila especificada.
3. Insira uma linha nula para terminar a entrada.
4. Execute **imqsgets** *queue-name* para recuperar todas as linhas e exibi-las no console.

Programa de amostra DPUT (imqdput.cpp)

Este programa de amostra C++ coloca mensagens em uma lista de distribuição que consiste em duas filas.

DPUT mostra o uso da classe ImqDistributionList (consulte Classe C++ ImqDistributionList). Essa amostra não é suportada no z/OS

1. Execute **imqdputs** *queue-name-1 queue-name-2* para colocar mensagens nas duas filas nomeadas.
2. Execute **imqsgets** *queue-name-1* e **imqsgets** *queue-name-2* para recuperar as mensagens dessas filas.

contraprestações sobre linguagem C++

Esta coleção de tópicos detalha os aspectos do uso da linguagem C++ e convenções que deve-se considerar ao gravar programas de aplicativos que usam um Message Queue Interface (MQI).

Arquivos de cabeçalho C++

Os arquivos de cabeçalho são fornecidos como parte da definição do MQI, para ajudá-lo a gravar programas aplicativos WebSphere MQ na linguagem C + +.

Esses arquivos de cabeçalho são resumidos na tabela a seguir.

Tabela 81. arquivos de cabeçalho C/C++

Nome do arquivo	**Conteúdos**
IMQI.HPP	Classes C++ MQI (inclui CMQC.H e IMQTYPE.H)
IMQTYPE.H	Define o tipo de dados do **ImqBoolean**
CMQC.H	estruturas de dados MQI e constantes manifest

Para melhorar a portabilidade dos aplicativos, codifique o nome do arquivo de cabeçalho em minúsculas na diretriz do pré-processador **#include**:

```
#include <imqi.hpp> // C++ classes
```

Métodos C++ e atributos

Nomes de métodos são compostos por letras maiúsculas e minúsculas. Várias considerações se aplicam aos parâmetros e valores de retorno. Atributos são acessados usando os métodos get e set conforme apropriado.

Parâmetros dos métodos que são *const* são apenas para entrada. Os parâmetros com assinaturas incluindo um ponteiro (*) ou uma referência (&) são passados por referência. Valores de retorno que não incluem um ponteiro ou uma referência são transmitidos por valor; no caso de objetos retornados, estes serão novas entidades que se tornarão responsabilidade do responsável pela chamada.

Algumas assinaturas de método incluem itens que tomam um padrão se não forem especificados. Tais itens estão sempre no final de assinaturas e são indicados por um sinal de igual (=); o valor após o sinal de igual indica o valor padrão que se aplicará se o item for omitido.

Todos os nomes de métodos nessas classes são compostos por letras maiúsculas e minúsculas, começando por minúsculas. Cada palavra, exceto a primeira dentro de um nome de método, começa com uma letra maiúscula. Abreviações não são usadas a menos que seu significado seja amplamente compreendido. Abreviaturas usadas incluem *ID* (para identidade) e *sync* (para sincronização).

Os atributos do objeto são acessados usando métodos get e set. Um método set começa com a palavra *set*; um método get não tem prefixo. Se um atributo for *read-only*, não haverá método set.

Atributos são inicializados para estados válidos durante a construção do objeto e o estado de um objeto é sempre consistente.

Tipos de dados em C++

Todos os tipos de dados são definidos pela instrução C **typedef**.

O tipo **ImqBoolean** é definido como **unsigned char** em IMQTYPE.H e pode ter os valores TRUE e FALSE. É possível usar os objetos de classe **ImqBinary** no lugar de matrizes **MQBYTE**, e objetos de classe **ImqString** no lugar de **char ***. Muitos métodos retornam objetos em vez de ponteiros **char** ou **MQBYTE** para facilitar o gerenciamento de armazenamento. Todos os valores de retorno se tornam responsabilidade do responsável pela chamada, e, no caso de um objeto retornado, o armazenamento pode ser disposto a usar a exclusão.

Manipulação de sequências binárias em C++

Sequências de dados binários são declaradas como objetos da classe **ImqBinary**. Objetos dessa classe podem ser copiados, comparados e configurados usando os operadores C familiares. Código de exemplo é fornecido.

A amostra de código a seguir mostra operações em uma sequência binária:

```
#include <imqi.hpp> // C++ classes

ImqMessage message ;
ImqBinary id, correlationId ;
MQBYTE24 byteId ;

correlationId.set( byteId, sizeof( byteId ) ); // Set.
id = message.id( );                            // Assign.
if ( correlationId == id ) {                   // Compare.
  ...
```

Manipulação de sequências de caracteres em C++

Os dados de caracteres são geralmente retornados nos objetos de classe **ImqString** que podem ser convertidos em **char *** usando um operador de conversão. A classe ImqString contém métodos para auxiliar no processamento de sequências de caracteres.

Quando os dados de caractere são aceitos ou retornados usando métodos MQI C++, os dados de caracteres são sempre terminados em nulos e podem ter qualquer comprimento. No entanto, determinados limites são impostos pelo WebSphere MQ que podem resultar em informações truncadas. Para facilitar o gerenciamento de armazenamento, os dados de caracteres são geralmente retornados em objetos de classe **ImqString**. Esses objetos podem ser convertidos em **char ***, usando o operador de conversão fornecido e usado para fins de *leitura apenas* em muitas situações em que um **char *** é necessário.

Nota: A conversão **char *** resulta de um objeto de classe **ImqString** pode ser nula.

Embora as funções C possam ser usadas no **char ***, há métodos especiais da classe **ImqString** que são preferíveis; **operator length**() é o equivalente de **strlen** e **storage**() Indica a memória alocada para os dados de caractere

Estado inicial de objetos em C++

Todos os objetos possuem um estado inicial consistente refletido por seus atributos. Os valores iniciais são definidos nas descrições de classe.

Usando C a partir de C++

Ao usar as funções C de um programa C++, inclua cabeçalhos apropriados.

O exemplo a seguir mostra `string.h` incluído em um programa C++:

```
extern "C" {
#include <string.h>
}
```

Convenções de notação do C++

Este exemplo mostra como chamar métodos e declarar parâmetros.

Esta amostra de código usa os métodos e parâmetros**ImqBoolean ImqQueue**::**get**(**ImqMessage &** *msg*)

Declare e use os parâmetros da seguinte forma:

```
ImqQueueManager * pmanager ;    // Queue manager
ImqQueue * pqueue ;             // Message queue
ImqMessage msg ;                // Message
char szBuffer[ 100 ];          // Buffer for message data

pmanager = new ImqQueueManager ;
pqueue = new ImqQueue ;
pqueue -> setName( "myreplyq" );
pqueue -> setConnectionReference( pmanager );

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );

if ( pqueue -> get( msg ) ) {
  long lDataLength = msg.dataLength( );

  ...
}
```

Operações implícitas em C++

Várias operações podem ocorrer implicitamente, *no tempo exato*, para atender às condições de pré-requisito para a execução bem-sucedida de um método. Essas operações implícitas são conectar, abrir, reabrir, fechar e desconectar. É possível controlar o comportamento implícito de conectar e abrir usando atributos de classe.

Connect

Um objeto do Gerenciador de ImqQueueeé conectado automaticamente para qualquer método que resulte em qualquer chamada para o MQI (consulte C++ e MQI referência cruzada).

Abrir

Um objeto ImqObject é aberto automaticamente para qualquer método que resulta em uma chamada MQGET, MQINQ, MQPUT ou MQSET. Use o método **openFor** para especificar um ou mais valores de **opção aberta** relevante.

Reabrir

Um ImqObject é reaberto automaticamente para qualquer método que resulta em uma chamada MQGET, MQINQ, MQPUT ou MQSET, em que o objeto já está aberto, mas as **opções abertas** existentes não são

adequadas para permitir que a chamada MQI seja bem-sucedida. O objeto é temporariamente fechado usando um valor de **opções fechadas** temporárias de MQCO_NONE. Use o método **openFor** para incluir um relevante**opções abertas**.

Reabrir pode causar problemas em circunstâncias específicas:

- Uma fila dinâmica temporária é destruída quando ela é fechada e nunca pode ser reaberta.
- Uma fila aberta para entrada exclusiva (seja explicitamente ou por padrão) pode ser acessada por outros na janela de oportunidade durante o fechamento e a reabertura.
- A posição do cursor de pesquisa é perdida quando uma fila está fechada. Esta situação não impede o fechamento e a reabertura, mas impede o uso subsequente do cursor até que MQGMO_BROWSE_FIRST seja usado novamente.
- O contexto da última mensagem recuperada é perdido quando uma fila está fechada.

Se qualquer uma destas circunstâncias ocorrer ou puder ser prevista, evite reaberturas configurando explicitamente **opções de abertura** adequadas antes que um objeto seja aberto (seja explícita ou implicitamente).

Configurar **opções abertas** explicitamente para situações de manipulação de filas complexas resulta em melhor desempenho e evita os problemas associados ao uso da reabertura.

Fechar

Um ImqObject é fechado automaticamente em qualquer ponto em que o estado do objeto não é mais viável, por exemplo, se uma referência de conexão ImqObject for interrompida ou se um objeto ImqObject for destruído.

Desconectar

Um ImqQueueManager é desconectado automaticamente em qualquer ponto em que a conexão não é mais viável, por exemplo, se uma referência de conexão ImqObject for severa ou se um objeto ImqQueueManager for destruído.

Sequências binárias e de caracteres em C++

A classe ImqString contém o formato de dados *char ** tradicional. A classe ImqBinary contém a matriz de bytes binários. Alguns métodos que configuram os dados de caracteres podem truncar os dados.

Métodos que configuram dados de caractere (**char ***) sempre usam uma cópia dos dados, mas alguns métodos podem truncar a cópia, porque determinados limites são impostos pelo WebSphere MQ.

A classe ImqString (consulte ImqString C++ class) encapsula o **char *** tradicional e fornece suporte para:

- Comparação
- Concatenação
- Copiando
- Conversão de número inteiro para texto e de texto para número inteiro
- Extração de token (palavra)
- Conversão de maiúscula

A classe ImqBinary (consulte ImqBinary C++ class) encapsula matrizes de bytes binários de tamanho arbitrário. Em particular, ela é usada para conter os seguintes atributos:

- **token de contabilidade** (MQBYTE32)
- **identificação da conexão** (MQBYTE128)
- **ID de correlação** (MQBYTE24)
- **token de recurso** (MQBYTE8)
- **ID de grupo** (MQBYTE24)
- **ID de instância** (MQBYTE24)

- **ID de mensagem** (MQBYTE24)
- **token de mensagem** (MQBYTE16)
- **ID de instância de transação** (MQBYTE16)

Em que estes atributos pertencem a objetos das seguintes classes:

- Cabeçalho ImqCICSBridge(consulte ImqCICSBridgeClasse C++ do Cabeçalho)
- ImqGetMessageOptions (consulte ImqGetMessageOptions Classe C++)
- Cabeçalho ImqIMSBridge(consulte ImqIMSBridgeClasse C++ do Cabeçalho)
- ImqMessageTracker (veja ImqMessageTracker C++ class)
- ImqQueueManager (consulte ImqQueueManager C++ class)
- ImqReferenceHeader (veja ImqReferenceHeader C++ class)
- ImqWorkCabeçalho (consulte ImqWorkClasse C++ do cabeçalho)

A classe ImqBinary também fornece suporte para comparação e cópia.

Funções não suportadas em C++

As classes e métodos C++ do WebSphere MQ são independentes da plataforma do WebSphere MQ Eles podem, portanto, oferecer algumas funções que não sejam suportadas em certas plataformas.

Se você tentar usar uma função em uma plataforma na qual ela não é suportada, a função será detectada pelo WebSphere MQ , mas não pelas ligações de linguagem C + +. O WebSphere MQ relata o erro para seu programa, como qualquer outro erro MQI.

Sistema de mensagens em C++

Esta coleção de tópicos detalha como preparar, ler e gravar mensagens em C++.

Preparando dados da mensagem em C++

Os dados da mensagem são preparados em um buffer, que pode ser fornecido pelo sistema ou aplicativo. Há vantagens para qualquer de um dos métodos. Exemplos de como usar um buffer são fornecidos.

Ao enviar uma mensagem, os dados da mensagem são preparados pela primeira vez em um buffer gerenciado por um objeto ImqCache (consulte ImqCache Classe C++). Um buffer é associado (por herança) a cada objeto ImqMessage (consulte ImqMessage C++ class): ele pode ser fornecido pelo aplicativo (usando o método **useEmptyBuffer** ou **useFullBuffer**) ou automaticamente pelo sistema. A vantagem do aplicativo que fornece o buffer de mensagem é que nenhuma cópia de dados é necessária em muitos casos porque o aplicativo pode usar áreas de dados preparadas diretamente. A desvantagem é que o buffer fornecido é de um comprimento fixo.

O buffer pode ser reutilizado e o número de bytes transmitidos pode ser ativado de cada vez, usando o método **setMessageLength** antes da transmissão.

Quando fornecido automaticamente pelo sistema, o número de bytes disponíveis é gerenciado pelo sistema e os dados podem ser copiados para o buffer de mensagem usando, por exemplo, o método **wriite** ImqCache ou o método **writeItem** ImqMessage. O buffer de mensagem aumenta de acordo com a necessidade. Conforme o buffer aumentar, não há perda de dados gravados anteriormente. Uma mensagem grande ou multipartes pode ser gravada em partes sequenciais.

Os exemplos a seguir mostram envios da mensagem simplificada.

1. Use os dados preparados em um buffer fornecido pelo usuário

```
char szBuffer[ ] = "Hello world" ;

msg.useFullBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
```

2. Use os dados preparados em um buffer fornecido pelo usuário, em que o tamanho do buffer excede o tamanho de dados

```
char szBuffer[ 24 ] = "Hello world" ;

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.setMessageLength( 12 );
```

3. Copiar dados para um buffer fornecido pelo usuário

```
char szBuffer[ 12 ];

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

4. Copiar dados para um buffer fornecido pelo sistema

```
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

5. Copiar dados para um buffer fornecido pelo sistema usando os objetos (objetos configuram o formato da mensagem, bem como o conteúdo)

```
ImqString strText( "Hello world" );

msg.writeItem( strText );
```

Lendo mensagens em C++

Um buffer pode ser fornecido pelo aplicativo ou pelo sistema. Os dados podem ser acessados diretamente do buffer ou lidos sequencialmente. Há uma classe equivalente a cada tipo de mensagem. Código de amostra é fornecido.

Ao receber dados, o aplicativo ou o sistema pode fornecer um buffer de mensagem adequado. O mesmo buffer pode ser usado para diversas transmissões e diversos recebimentos para um objeto ImqMessage específico. Se o buffer de mensagem for fornecido automaticamente, ele aumenta para acomodar o comprimento de dados recebido. No entanto, um buffer de mensagem fornecido pelo aplicativo pode não ser grande o suficiente para conter os dados recebidos. Em seguida, truncamento ou falha pode ocorrer, dependendo das opções usadas para o recebimento de mensagens.

Dados de entrada podem ser acessados diretamente do buffer de mensagem, nesse caso, o comprimento dos dados indica a quantia total de dados recebidos. Como alternativa, dados recebidos podem ser lidos sequencialmente a partir do buffer de mensagem. Neste caso, o ponteiro de dados endereça o próximo byte de dados recebidos e o ponteiro de dados e o comprimento de dados são atualizados cada vez que os dados são lidos.

Itens são partes de uma mensagem, todos na área do usuário do buffer de mensagem, que precisam ser processados sequencialmente e separadamente. Além de dados do usuário regulares, um item pode ser um cabeçalho de mensagens não entregues ou uma mensagem do acionador. Os itens são sempre associados aos formatos de mensagem; os formatos de mensagem ***nem*** sempre são associados a itens.

Há uma classe de objeto para cada item que corresponde a um formato de mensagem reconhecível WebSphere MQ . Há uma para um cabeçalho de mensagens não entregues e uma para uma mensagem do acionador. Não há classe de objeto para dados do usuário. Ou seja, quando os formatos reconhecíveis

estiverem esgotados, o processamento do restante será deixado para o programa de aplicativo. Classes de dados do usuário podem ser escritas especializando a classe ImqItem.

O exemplo a seguir mostra o recebimento de uma mensagem que leva em consideração diversos itens em potencial que podem preceder os dados do usuário, em uma situação imaginária. Os dados do usuário não de item são definidos como tudo o que ocorre após itens que podem ser identificados. Um buffer automático (o padrão) é usado para conter uma quantia arbitrária de dados da mensagem.

```
ImqQueue queue ;
ImqMessage msg ;

if ( queue.get( msg ) ) {

  /* Process all items of data in the message buffer. */
  do while ( msg.dataLength( ) ) {
    ImqBoolean bFormatKnown = FALSE ;
    /* There remains unprocessed data in the message buffer. */

    /* Determine what kind of item is next. */

    if ( msg.formatIs( MQFMT_DEAD_LETTER_HEADER ) ) {
      ImqDeadLetterHeader header ;
      /* The next item is a dead-letter header.                */
      /* For the next statement to work and return TRUE,       */
      /* the correct class of object pointer must be supplied. */
      bFormatKnown = TRUE ;

      if ( msg.readItem( header ) ) {
        /* The dead-letter header has been extricated from the */
        /* buffer and transformed into a dead-letter object.   */
        /* The encoding and character set of the dead-letter   */
        /* object itself are MQENC_NATIVE and MQCCSI_Q_MGR.    */
        /* The encoding and character set from the dead-letter */
        /* header have been copied to the message attributes   */
        /* to reflect any remaining data in the buffer.        */

        /* Process the information in the dead-letter object.  */
        /* Note that the encoding and character set have       */
        /* already been processed.                             */
        ...
      }
      /* There might be another item after this, */
      /* or just the user data.                  */
    }
    if ( msg.formatIs( MQFMT_TRIGGER ) ) {
      ImqTrigger trigger ;
      /* The next item is a trigger message.                   */
      /* For the next statement to work and return TRUE,       */
      /* the correct class of object pointer must be supplied. */
      bFormatKnown = TRUE ;
      if ( msg.readItem( trigger ) ) {

        /* The trigger message has been extricated from the */
        /* buffer and transformed into a trigger object.    */
        /* Process the information in the trigger object. */
        ...
      }

      /* There is usually nothing after a trigger message. */
    }

    if ( msg.formatIs( FMT_USERCLASS ) ) {
      UserClass object ;
      /* The next item is an item of a user-defined class.     */
      /* For the next statement to work and return TRUE,       */
      /* the correct class of object pointer must be supplied. */
      bFormatKnown = TRUE ;

      if ( msg.readItem( object ) ) {
        /* The user-defined data has been extricated from the */
        /* buffer and transformed into a user-defined object. */

        /* Process the information in the user-defined object. */
        ...
      }

      /* Continue looking for further items. */
    }
```

```
    if ( ! bFormatKnown ) {
      /* There remains data that is not associated with a specific*/
      /* item class.                                               */
      char * pszDataPointer = msg.dataPointer( );       /* Address.*/
      int iDataLength = msg.dataLength( );              /* Length. */

      /* The encoding and character set for the remaining data are */
      /* reflected in the attributes of the message object, even   */
      /* if a dead-letter header was present.                      */
      ...

    }

  }
}
```

Neste exemplo, FMT_USERCLASS é uma constante que representa o nome do formato de 8 caracteres associada a um objeto de classe `UserClass` e definida pelo aplicativo.

`UserClass` é derivado da classe ImqItem (consulte ImqItem classe C++) e implementa os métodos **copyOut** e **pasteIn** virtuais dessa classe.

Os próximos dois exemplos mostram código da classe ImqDeadLetterHeader (consulte ImqDeadLetterHeader C++ class). O primeiro exemplo mostra a mensagem encapsulada customizada-código*writing* .

```
// Insert a dead-letter header.
// Return TRUE if successful.
ImqBoolean ImqDeadLetterHeader :: copyOut ( ImqMessage & msg ) {
  ImqBoolean bSuccess ;
  if ( msg.moreBytes( sizeof( omqdlh ) ) ) {
    ImqCache cacheData( msg ); // Preserve original message content.
    // Note original message attributes in the dead-letter header.
    setEncoding( msg.encoding( ) );
    setCharacterSet( msg.characterSet( ) );
    setFormat( msg.format( ) );

    // Set the message attributes to reflect the dead-letter header.
    msg.setEncoding( MQENC_NATIVE );
    msg.setCharacterSet( MQCCSI_Q_MGR );
    msg.setFormat( MQFMT_DEAD_LETTER_HEADER );
    // Replace the existing data with the dead-letter header.
    msg.clearMessage( );
    if ( msg.write( sizeof( omqdlh ), (char *) & omqdlh ) ) {
      // Append the original message data.
      bSuccess = msg.write( cacheData.messageLength( ),
                            cacheData.bufferPointer( ) );
    } else {
      bSuccess = FALSE ;
    }
  } else {
    bSuccess = FALSE ;
  }
  // Reflect and cache error in this object.
  if ( ! bSuccess ) {
    setReasonCode( msg.reasonCode( ) );
    setCompletionCode( msg.completionCode( ) );
  }

  return bSuccess ;
}
```

O segundo exemplo mostra o código de*leitura* da mensagem encapsulada customizada

```
// Read a dead-letter header.
// Return TRUE if successful.
ImqBoolean ImqDeadLetterHeader :: pasteIn ( ImqMessage & msg ) {
  ImqBoolean bSuccess = FALSE ;

  // First check that the eye-catcher is correct.
  // This is also our guarantee that the "character set" is correct.
  if ( ImqItem::structureIdIs( MQDLH_STRUC_ID, msg ) ) {
    // Next check that the "encoding" is correct, as the MQDLH
```

```
    // contains numeric data.
    if ( msg.encoding( ) == MQENC_NATIVE ) {

      // Finally check that the "format" is correct.
      if ( msg.formatIs( MQFMT_DEAD_LETTER_HEADER ) ) {
        char * pszBuffer = (char *) & omqdlh ;
        // Transfer the MQDLH from the message and move pointer on.
        if ( bSuccess = msg.read( sizeof( omdlh ), pszBuffer ) ) {
          // Update the encoding, character set and format of the
          // message to reflect the remaining data.
          msg.setEncoding( encoding( ) );
          msg.setCharacterSet( characterSet( ) );
          msg.setFormat( format( ) );
        } else {

          // Reflect the cache error in this object.
          setReasonCode( msg.reasonCode( ) );
          setCompletionCode( msg.completionCode( ) );
        }
      } else {
        setReasonCode( MQRC_INCONSISTENT_FORMAT );
        setCompletionCode( MQCC_FAILED );
      }
    } else {
      setReasonCode( MQRC_ENCODING_ERROR );
      setCompletionCode( MQCC_FAILED );
    {
  } else {
    setReasonCode( MQRC_STRUC_ID_ERROR );
    setCompletionCode( MQCC_FAILED );
  }

  return bSuccess ;
}
```

Com um buffer automático, o armazenamento em buffer é *volátil*. Ou seja, os dados do buffer podem ser mantidos em um local físico diferente após cada chamada do método **get**. Portanto, toda vez que o buffer de dados é referido, use os métodos **bufferPointer** ou **dataPointer** para acessar dados da mensagem.

Você pode desejar que um programa separe uma área fixa para receber dados da mensagem. Nesse caso, chame o método **useEmptyBuffer** antes de usar o método **get**.

Usar uma área fixa, não automática, limita as mensagens a um tamanho máximo, portanto, é importante considerar a opção MQGMO_ACCEPT_TRUNCATED_MSG do objeto ImqGetMessageOptions. Se essa opção não for especificada (o padrão), o código de razão MQRC_TRUNCATED_MSG_FAILED pode ser esperado. Se essa opção for especificada, o código de razão MQRC_TRUNCATED_MSG_ACCEPTED pode ser esperado, dependendo do design do aplicativo.

O próximo exemplo mostra como uma área fixa de armazenamento pode ser usada para receber mensagens:

```
char * pszBuffer = new char[ 100 ];

msg.useEmptyBuffer( pszBuffer, 100 );
gmo.setOptions( MQGMO_ACCEPT_TRUNCATED_MSG );
queue.get( msg, gmo );

delete [ ] pszBuffer ;
```

Nesse fragmento de código, o buffer pode sempre ser direcionado diretamente, com *pszBuffer*, em vez de usar o método **bufferPointer**. No entanto, é melhor usar o método **dataPointer** para acesso de propósito geral. O aplicativo (não o objeto de classe ImqCache) deve descartar um buffer definido pelo usuário (nonautomatic).

Atenção: especificar um ponteiro nulo e comprimento zero com **useEmptyBuffer** não denomina um buffer de comprimento fixo com comprimento zero como pode ser esperado. Essa combinação é interpretada como uma solicitação para ignorar qualquer buffer anterior definido pelo usuário e, em vez disso, reverter para o uso de um buffer automático.

Gravando uma mensagem na fila de mensagens não entregues em C++

Código do programa de exemplo para gravar uma mensagem na fila de mensagens não entregues.

Um caso típico de uma mensagem multipartes é um que contém um cabeçalho de mensagens não entregues. Os dados de uma mensagem que não podem ser processados são anexados ao cabeçalho da fila de devoluções.

```
ImqQueueManager mgr ;        // The queue manager.
ImqQueue queueIn ;           // Incoming message queue.
ImqQueue queueDead ;         // Dead-letter message queue.
ImqMessage msg ;             // Incoming and outgoing message.
ImqDeadLetterHeader header ; // Dead-letter header information.

// Retrieve the message to be rerouted.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Set up the dead-letter header information.
header.setDestinationQueueManagerName( mgr.name( ) );
header.setDestinationQueueName( queueIn.name( ) );
header.setPutApplicationName( /* ? */ );
header.setPutApplicationType( /* ? */ );
header.setPutDate( /* TODAY */ );
header.setPutTime( /* NOW */ );
header.setDeadLetterReasonCode( FB_APPL_ERROR_1234 );

// Insert the dead-letter header information. This will vary
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );

// Send the message to the dead-letter queue.
queueDead.setConnectionReference( mgr );
queueDead.setName( mgr.deadLetterQueueName( ) );
queueDead.put( msg );
```

Gravando uma mensagem na ponte IMS em C++

Código do programa de exemplo para gravar uma mensagem na ponte do IMS

Mensagens enviadas para a ponte WebSphere MQ-IMS podem usar um cabeçalho especial. O cabeçalho da ponte IMS é prefixado para os dados da mensagem regular

```
ImqQueueManager    mgr;         // The queue manager.
ImqQueue           queueBridge; // IMS bridge message queue.
ImqMessage         msg;         // Outgoing message.
ImqIMSBridgeHeader header;      // IMS bridge header.

// Set up the message.
//
// Here we are constructing a message with format
// MQFMT_IMS_VAR_STRING, and appropriate data.
//
msg.write( 2,       /* ? */ );         // Total message length.
msg.write( 2,       /* ? */ );         // IMS flags.
msg.write( 7,       /* ? */ );         // Transaction code.
msg.write( /* ? */, /* ? */ );         // String data.
msg.setFormat( MQFMT_IMS_VAR_STRING ); // The format attribute.

// Set up the IMS bridge header information.
//
// The reply-to-format is often specified.
// Other attributes can be specified, but all have default values.
//
header.setReplyToFormat( /* ? */ );

// Insert the IMS bridge header into the message.
//
// This will:
// 1) Insert the header into the message buffer, before the existing
//    data.
// 2) Copy attributes out of the message descriptor into the header,
```

```
//    for example the IMS bridge header format attribute will now
//    be set to MQFMT_IMS_VAR_STRING.
// 3) Set up the message attributes to describe the header, in
//    particular setting the message format to MQFMT_IMS.
//
msg.writeItem( header );

// Send the message to the IMS bridge queue.
//
queueBridge.setConnectionReference( mgr );
queueBridge.setName( /* ? */ );
queueBridge.put( msg );
```

Gravando uma mensagem na ponte CICS em C++

Código do programa de exemplo para gravar uma mensagem na ponte do CICS

Mensagens enviadas para o WebSphere MQ para z/OS usando a ponte CICS requerem um cabeçalho especial. O cabeçalho da ponte CICS é prefixado aos dados da mensagem regular

```
ImqQueueManager mgr ;         // The queue manager.
ImqQueue queueIn ;            // Incoming message queue.
ImqQueue queueBridge ;        // CICS bridge message queue.
ImqMessage msg ;              // Incoming and outgoing message.
ImqCicsBridgeHeader header ;  // CICS bridge header information.

// Retrieve the message to be forwarded.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Set up the CICS bridge header information.
// The reply-to format is often specified.
// Other attributes can be specified, but all have default values.
header.setReplyToFormat( /* ? */ );

// Insert the CICS bridge header information.  This will vary
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );

// Send the message to the CICS bridge queue.
queueBridge.setConnectionReference( mgr );
queueBridge.setName( /* ? */ );
queueBridge.put( msg );
```

Gravando uma mensagem com um cabeçalho de trabalho em C++

Código do programa de exemplo para gravar uma mensagem destinada a uma fila gerenciada pelo z/OS Workload Manager.

Mensagens enviadas ao WebSphere MQ para z/OS, que são destinadas a uma fila gerenciada pelo z/OS Workload Manager, requerem um cabeçalho especial. O cabeçalho do trabalho tem como prefixo dados da mensagem regular.

```
ImqQueueManager mgr ;             // The queue manager.
ImqQueue queueIn ;                // Incoming message queue.
ImqQueue queueWLM ;               // WLM managed queue.
ImqMessage msg ;                  // Incoming and outgoing message.
ImqWorkHeader header ;            // Work header information

// Retrieve the message to be forwarded.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Insert the Work header information.  This will vary
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );
```

```
// Send the message to the WLM managed queue.
queueWLM.setConnectionReference( mgr );
queueWLM.setName( /* ? */ );
queueWLM.put( msg );
```

Construindo programas C++ do WebSphere MQ

A URL dos compiladores suportados é listada, juntamente com os comandos a serem usados para compilar, vincular e executar programas e amostras C++ nas plataformas WebSphere MQ .

Os compiladores para cada plataforma e versão suportados do WebSphere MQ estão listados na página de requisitos do sistema do WebSphere MQ em Requisitos do Sistema para IBM WebSphere MQ

O comando necessário para compilar e vincular seu programa C++ do WebSphere MQ depende de sua instalação e requisitos. Os exemplos a seguir mostram comandos típicos de compilação e link para alguns dos compiladores usando a instalação padrão do WebSphere MQ em várias plataformas.

Construindo programas C++ no AIX

Construa programas C++ do WebSphere MQ no AIX usando o compilador XL C Enterprise Edition ..

Client

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

Aplicativo não encadeado de 32 bits

```
xlC -o imqsputc_32 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqc23ia -limqb23ia -lmqic
```

Aplicativo encadeado de 32 bits

```
xlC_r -o imqsputc_32_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqc23ia_r -limqb23ia_r -lmqic_r
```

Aplicativo não encadeado de 64 bits

```
xlC -q64 -o imqsputc_64 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqc23ia -limqb23ia -lmqic
```

Aplicativo encadeado de 64 bits

```
xlC_r -q64 -o imqsputc_64_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqc23ia_r -limqb23ia_r -lmqic_r
```

Servidor

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

Aplicativo não encadeado de 32 bits

```
xlC -o imqsput_32 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqs23ia -limqb23ia -lmqm
```

Aplicativo encadeado de 32 bits

```
xlC_r -o imqsput_32_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -limqs23ia_r -limqb23ia_r -lmqm_r
```

Aplicativo não encadeado de 64 bits

```
xlC -q64 -o imqsput_64 imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqs23ia -limqb23ia -lmqm
```

Aplicativo encadeado de 64 bits

```
xlC_r -q64 -o imqsput_64_r imqsput.cpp -qchars=signed -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -limqs23ia_r -limqb23ia_r -lmqm_r
```

Construindo programas C++ no HP-UX

Construa programas WebSphere MQ C++ no HP-UX usando os compiladores aC+ + ou aCC .

No HP-UX Itanium, o WebSphere MQ suporta apenas o tempo de execução Padrão Use o compilador aCC.

- libimqi23bh.sl fornece as classes C++ WebSphere MQ para o tempo de execução Padrão.
- Para compatibilidade com liberações anteriores, um link simbólico é fornecido de libimqi23ah.sl para libimqi23bh.sl.

IA64 (IPF)

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

Cliente: IA64 (IPF)

Aplicativo não encadeado de 32 bits

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh -lmqic
```

Aplicativo encadeado de 32 bits

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh_r -lmqic_r -lpthread
```

Aplicativo não encadeado de 64 bits

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsputc_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh -lmqic
```

Aplicativo encadeado de 64 bits

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsputc_64_r imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh_r
-lmqic_r
-lpthread
```

Servidor: IA64 (IPF)

Aplicativo não encadeado de 32 bits

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh -lmqm
```

Aplicativo encadeado de 32 bits

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh_r -lmqm_r -lpthread
```

Aplicativo não encadeado de 64 bits

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsput_64 imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh -lmqm
```

Aplicativo encadeado de 64 bits

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsput_64_r imqsput.cpp
-IMQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh_r
-lmqm_r
-lpthread
```

Construindo programas C++ no Linux

Construa programas C++ do WebSphere MQ no Linux usando o compilador GNU g + +.

System p

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

Cliente: System p

Aplicativo não encadeado de 32 bits

```
g++ -m32 -o imqsputc_32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl
-limqb23gl -lmqic
```

Aplicativo encadeado de 32 bits

```
g++ -m32 -o imqsputc_r32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r
-limqb23gl_r -lmqic_r
```

Aplicativo não encadeado de 64 bits

```
g++ -m64 -o imqsputc_64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Aplicativo encadeado de 64 bits

```
g++ -m64 -o imqsputc_r64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r
```

Servidor: Sistema p

Aplicativo não encadeado de 32 bits

```
g++ -m32 -o imqsput_32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl
-limqb23gl -lmqm
```

Aplicativo encadeado de 32 bits

```
g++ -m32 -o imqsput_r32 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
```

```
-limqs23gl_r
-limqb23gl_r -lmqm_r
```

Aplicativo não encadeado de 64 bits

```
g++ -m64 -o imqsput_64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl -limqb23gl -lmqm
```

Aplicativo encadeado de 64 bits

```
g++ -m64 -o imqsput_r64 imqsput.cpp -fsigned-char -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl_r -limqb23gl_r -lmqm_r
```

System z

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

Cliente: System z

Aplicativo não encadeado de 32 bits

```
g++ -m31 -fsigned-char -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl -limqb23gl -lmqic
```

Aplicativo encadeado de 32 bits

```
g++ -m31 -fsigned-char -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r -limqb23gl_r -lmqic_r
-lpthread
```

Aplicativo não encadeado de 64 bits

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Aplicativo encadeado de 64 bits

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64  -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Servidor: System z

Aplicativo não encadeado de 32 bits

```
g++ -m31 -fsigned-char -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib  -Wl,-rpath=/usr/lib
-limqs23gl -limqb23gl -lmqm
```

Aplicativo encadeado de 32 bits

```
g++ -m31 -fsigned-char -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

Aplicativo não encadeado de 64 bits

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl -limqb23gl -lmqm
```

Aplicativo encadeado de 64 bits

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

Sistema x (32 bits)

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

Cliente: System x (32 bits)

Aplicativo não encadeado de 32 bits

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,
-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqc23gl -limqb23gl -lmqic
```

Aplicativo encadeado de 32 bits

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqc23gl_r -limqb23gl_r
-lmqic_r -lpthread
```

Aplicativo não encadeado de 64 bits

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqc23gl -limqb23gl
-lmqic
```

Aplicativo encadeado de 64 bits

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqc23gl_r -limqb23gl_r
-lmqic_r -lpthread
```

Servidor: System x (32 bits)

Aplicativo não encadeado de 32 bits

```
g++ -m32 -fsigned-char -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqs23gl -limqb23gl -lmqm
```

Aplicativo encadeado de 32 bits

```
g++ -m32 -fsigned-char -o imqsput_32_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATHlib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -LMQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqs23gl_r -limqb23gl_r
-lmqm_r -lpthread
```

Aplicativo não encadeado de 64 bits

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqs23gl -limqb23gl -lmqm
```

Aplicativo encadeado de 64 bits

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -LMQ_INSTALLATION_PATH/
lib64
```

```
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqs23gl_r -limqb23gl_r
-lmqm_r -lpthread
```

Construindo programas C++ no Solaris

Construa programas C++ do WebSphere MQ no Solaris usando o compilador Sun ONE.

SPARC

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

Cliente: SPARC

Aplicativo de 32 bits

```
CC -xarch=v8plus -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic -lsocket -lnsl -ldl
```

Aplicativo de 64 bits

```
CC -xarch=v9 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Servidor: SPARC

Aplicativo de 32 bits

```
CC -xarch=v8plus -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as
-lmqm -lsocket -lnsl -ldl
```

Aplicativo de 64 bits

```
CC -xarch=v9 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as
-lmqm -lsocket -lnsl -ldl
```

x86-64

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

Cliente: x86-64

Aplicativo de 32 bits

```
CC -xarch=386 -mt -o imqsputc_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic -lsocket -lnsl -ldl
```

Aplicativo de 64 bits

```
CC -xarch=amd64 -mt -o imqsputc_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Servidor: x86-64

Aplicativo de 32 bits

```
CC -xarch=386 -mt -o imqsput_32 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib -RMQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as
-lmqm -lsocket -lnsl -ldl
```

Aplicativo de 64 bits

```
CC -xarch=amd64 -mt -o imqsput_64 imqsput.cpp -IMQ_INSTALLATION_PATH/inc
-LMQ_INSTALLATION_PATH/lib64 -RMQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as
-lmqm -lsocket -lnsl -ldl
```

Construindo programas C++ no Windows

Construa programas C++ do WebSphere MQ no Windows usando o compilador C++ do Microsoft Visual Studio.

Arquivos de biblioteca (.lib) e arquivos dll para uso com aplicativos de 32 bits são instalados em *MQ_INSTALLATION_PATH*/Tools/Lib, arquivos para uso com aplicativos de 64 bits são instalados em *MQ_INSTALLATION_PATH*/Tools/Lib64.*MQ_INSTALLATION_PATH* Representa o diretório de alto nível no qual o WebSphere MQ está instalado

Client

```
cl -MD imqsput.cpp /Feimqsputc.exe imqb23vn.lib imqc23vn.lib
```

Servidor

```
 cl -MD imqsput.cpp /Feimqsput.exe imqb23vn.lib imqs23vn.lib
```

Usando classes do WebSphere MQ para Java

As classes do WebSphere MQ para Java permitem usar o WebSphere MQ em um ambiente Java. Um aplicativo Java pode usar classes WebSphere MQ para Java ou classes WebSphere MQ para JMS para acessar recursos do WebSphere MQ .

As classes WebSphere MQ para Java permitem que um aplicativo Java:

- Conecte-se ao WebSphere MQ como um cliente WebSphere MQ
- Conecte-se diretamente a um gerenciador de filas do WebSphere MQ

As classes WebSphere MQ para Java encapsulam a Message Queue Interface (MQI), a API WebSphere MQ nativa.

As classes do WebSphere MQ para Java usam um modelo de objeto semelhante às interfaces C++ e .NET para WebSphere MQ.

Por que devo usar as classes WebSphere MQ para Java?

Se os pontos a seguir forem significativos em sua instalação, considere usar as classes do WebSphere MQ para Java:

- As classes WebSphere MQ para Java encapsulam a Message Queue Interface (MQI), a API WebSphere MQ nativa.
  - Se você estiver familiarizado com o uso do MQI em linguagens processuais, será possível transferir esse conhecimento para o ambiente Java

  - É possível explorar o intervalo completo de recursos do WebSphere MQ, além daqueles disponíveis por meio do JMS
- As classes do WebSphere MQ para Java usam um modelo de objeto semelhante às interfaces C++ e .NET para WebSphere MQ. Se você estiver familiarizado com essas interfaces, será possível transferir esse conhecimento para o ambiente Java

Nota: A reconexão automática do cliente não é suportada pelas classes do WebSphere MQ para Java

Introdução às classes do WebSphere MQ para Java

Esta coleção de tópicos fornece uma visão geral das classes do WebSphere MQ para Java e seus usos

O que são classes do WebSphere MQ para Java?

As classes do WebSphere MQ para Java permitem usar o WebSphere MQ em um ambiente Java.

As classes WebSphere MQ para Java permitem que um aplicativo Java:

- Conecte-se ao WebSphere MQ como um cliente WebSphere MQ
- Conecte-se diretamente a um gerenciador de filas do WebSphere MQ

As classes WebSphere MQ para Java encapsulam a Message Queue Interface (MQI), a API WebSphere MQ nativa.

As classes do WebSphere MQ para Java usam um modelo de objeto semelhante às interfaces C++ e .NET para WebSphere MQ.

Por que devo usar as classes WebSphere MQ para Java?

Um aplicativo Java pode usar classes WebSphere MQ para Java ou classes WebSphere MQ para JMS para acessar recursos do WebSphere MQ . Há várias vantagens em usar classes do WebSphere MQ para Java.

Se os pontos a seguir forem significativos em sua instalação, considere usar classes do Websphere MQ para Java:

- As classes WebSphere MQ para Java encapsulam a Message Queue Interface (MQI), a API WebSphere MQ nativa.
  - Se você estiver familiarizado com o uso do MQI em linguagens processuais, será possível transferir esse conhecimento para o ambiente Java
  - É possível explorar o intervalo completo de recursos do WebSphere MQ, além daqueles disponíveis por meio do JMS
- As classes do WebSphere MQ para Java usam um modelo de objeto semelhante às interfaces C++ e .NET para WebSphere MQ. Se você estiver familiarizado com essas interfaces, será possível transferir esse conhecimento para o ambiente Java

Opções de Conexão para Classes WebSphere MQ para Java

As classes WebSphere MQ para Java podem se conectar no modo cliente ou de ligações.

As opções programáveis permitem que as classes do WebSphere MQ para Java se conectem ao WebSphere MQ de uma das seguintes maneiras:

- Como um cliente MQI do WebSphere MQ usando Protocolo de Controle de Transmissões /Internet Protocol (TCP/IP)
- No modo de ligações, conectando-se diretamente ao WebSphere MQ usando a Java Native Interface (JNI)

Os clientes não podem ser executados no z/OS, mas os clientes em outras plataformas podem se conectar a um gerenciador de fila do WebSphere MQ para z/OS se o Client Attach Facility estiver instalado

As seções a seguir descrevem as opções de conexão do modo de cliente e do modo de ligações em mais detalhes

Conexão do cliente

Para conectar-se a um gerenciador de filas no modo cliente, um aplicativo WebSphere MQ pode ser executado no mesmo sistema no qual o gerenciador de filas está em execução ou em um sistema diferente. Em cada caso, as classes WebSphere MQ para Java se conectam ao gerenciador de filas sobre TCP/IP.

Uma classe WebSphere MQ para aplicativo Java pode se conectar a qualquer gerenciador de filas suportado usando o modo cliente.

Para obter mais informações sobre como gravar aplicativos para usar conexões no modo cliente, consulte "WebSphere MQ classes para modos de conexão Java" na página 676.

Conexão de ligações

Quando usado no modo de ligações, as classes do WebSphere MQ para Java usam a Java Native Interface (JNI) para chamar diretamente na API do gerenciador de filas existente, em vez de se comunicarem por meio de uma rede. Na maioria dos ambientes, a conexão no modo de ligações fornece melhor desempenho para as classes WebSphere MQ para aplicativos Java do que a conexão no modo de cliente, evitando o custo da comunicação TCP/IP.

Os aplicativos que usam as classes do WebSphere MQ para Java para conectar no modo de ligações devem ser executados no mesmo sistema que o gerenciador de filas ao qual eles estão se conectando.

O Java Runtime Environment, que está sendo usado para executar as classes WebSphere MQ para aplicativo Java, deve ser configurado para carregar as classes WebSphere MQ para bibliotecas Java; consulte As classes WebSphere MQ para bibliotecas Java para obter informações adicionais.

Para obter mais informações sobre como gravar aplicativos para usar conexões do modo de ligações, consulte "WebSphere MQ classes para modos de conexão Java" na página 676.

Pré-requisitos para classes do WebSphere MQ para Java

Para usar classes do WebSphere MQ para Java, você precisa de outros produtos de software.

Para obter as informações mais recentes sobre os pré-requisitos para classes do WebSphere MQ para Java, consulte o arquivo LEIA-ME WebSphere MQ .

Para desenvolver classes WebSphere MQ para aplicativos Java, é necessário um Java Development Kit (JDK). Detalhes dos JDKs suportados com seu sistema operacional podem ser localizados na página de requisitos do sistema do WebSphere MQ em Requisitos do Sistema para IBM WebSphere MQ.

Para executar classes WebSphere MQ para aplicativos Java, você precisa dos seguintes componentes de software:

- Um gerenciador de filas do WebSphere MQ para aplicativos que se conectam a um gerenciador de filas
- Um Java Runtime Environment (JRE), para cada sistema no qual você executa aplicativos. Um JRE adequado é fornecido com o WebSphere MQ.

Se você precisar de conexões SSL para usar módulos criptográficos que tenham sido certificados pelo FIPS 140-2, precisará do provedor Java JSSE FIPS IBM (IBMJSSEFIPS). Cada JDK e JRE do IBM na Versão 1.4.2 ou posterior contém IBMJSSEFIPS.

É possível usar os endereços do Internet Protocol Versão 6 (IPv6) em suas classes do WebSphere MQ para aplicativos Java se IPv6 for suportado pela Java virtual machine (JVM) e a implementação TCP/IP em seu sistema operacional.

Instalação e Configuração de Classes WebSphere MQ para Java

Esta seção descreve os diretórios e arquivos que são criados ao instalar classes do WebSphere MQ para Java e informa como configurar as classes do WebSphere MQ para Java após a instalação.

O que é instalado para classes do WebSphere MQ para Java

A versão mais recente das classes do WebSphere MQ para Java é instalada com o WebSphere MQ Talvez seja necessário substituir opções de instalação padrão para assegurar que isso seja feito.

Para obter informações adicionais sobre a instalação do WebSphere MQ , consulte:

Instalando um Servidor do WebSphere MQ
Instalando um IBM WebSphere MQ cliente

WebSphere MQ classes para Java estão contidas nos arquivos Java archive (JAR), com.ibm.mq.jare com.ibm.mq.jmqi.jar.

Suporte para cabeçalhos de mensagem padrão, como Programmable Command Format (PCF), está contido no arquivo JAR com.ibm.mq.headers.jar.

Suporte para Programmable Command Format (PCF) está contido no arquivo JAR com.ibm.mq.pcf.jar.

Instalando e Atualizando as Classes WebSphere MQ para Arquivos JAR Java

A única maneira suportada de obter as classes WebSphere MQ para arquivos JAR Java em um sistema é instalar o produto WebSphere MQ ou o cliente MQI WebSphere MQ SupportPacou usar uma ferramenta de gerenciamento de software como Apache Maven, para obter mais informações, consulte “IBM WebSphere MQ classes for Java e ferramentas de gerenciamento de software” na página 671.

Não mova ou copie as classes WebSphere MQ para arquivos JAR Java de outras máquinas, a menos que esteja usando uma ferramenta de gerenciamento de software.

- Os fix packs não podem ser aplicados a uma "instalação" em que os arquivos JAR foram copiados de outra máquina e tornam muito mais difícil assegurar que todos os arquivos JAR sejam mantidos em etapas entre si e estejam em níveis compatíveis.
- Copiar as classes WebSphere MQ para arquivos JAR JMS entre máquinas também pode resultar em várias cópias dos arquivos que residem na mesma máquina, o que pode causar problemas de manutenção do código e problemas de depuração.

Não inclua as classes do WebSphere MQ para arquivos JAR Java nos archives do aplicativo

- As atualizações para as classes do WebSphere MQ para Java não podem ser aplicadas usando um Fix Pack do WebSphere MQ
- Não é possível que o Suporte IBM determine facilmente a versão das classes do WebSphere MQ para Java que estão sendo usadas pelo aplicativo
- Podem surgir problemas se vários aplicativos em execução dentro do mesmo Java Runtime Environment incluírem diferentes versões das classes do WebSphere MQ para Java, já que várias versões das classes do WebSphere MQ para Java são carregadas no Java Runtime Environment ao mesmo tempo
- Se um aplicativo usar o transporte BINDINGS para se conectar a um gerenciador de filas, quaisquer upgrades principais para o gerenciador de filas também exigirão que o aplicativo seja atualizado para incluir o nível correspondente das classes do WebSphere MQ para Java

  Por exemplo, se um gerenciador de filas for atualizado para o nível do WebSphere MQ Versão 7.1 , então quaisquer aplicativos que se conectem ao gerenciador de filas usando o transporte BINDINGS também precisarão ser atualizados para incluir as classes do WebSphere MQ Versão 7.1 para Java

A biblioteca Java a seguir é distribuída com as classes WebSphere MQ para Java:

- connector.jar (Versão 1.0)

O aplicativo de amostra chamado Postcard está no arquivo JAR com.ibm.mq.postcard.jar.

A ferramenta Javadoc foi utilizada para gerar as páginas HTML contendo as especificações das classes WebSphere MQ para Java e WebSphere MQ para APIs JMS As páginas HTML estão no subdiretório doc das classes WebSphere MQ para o diretório de instalação JMS. Nos sistemas UNIX, Linuxe Windows , o subdiretório doc contém as páginas HTML individuais.

Quando a instalação for concluída, arquivos e amostras serão instalados nos locais mostrados em “Diretórios de Instalação para Classes WebSphere MQ para Java” na página 664.

Após a instalação, em qualquer plataforma diferente do Windows, deve-se atualizar suas variáveis de ambiente, conforme descrito em “Variáveis de Ambiente Relevantes para Classes do WebSphere MQ para Java” na página 664

Diretórios de Instalação para Classes WebSphere MQ para Java

As classes WebSphere MQ para arquivos Java são instaladas em locais diferentes de acordo com a plataforma.

A Tabela 82 na página 664 mostra onde as classes WebSphere MQ para arquivos Java estão instaladas

Tabela 82. WebSphere MQ classes para diretórios de instalação Java

Plataforma	**Diretório**
AIX	*MQ_INSTALLATION_PATH*/java/lib
HP-UX, Linuxe Solaris	*MQ_INSTALLATION_PATH*/java/lib
Windows	*MQ_INSTALLATION_PATH*\java\lib
O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado.	

Alguns aplicativos de amostra, como o Installation Verification Programs (IVP), são fornecidos com o WebSphere MQ. Tabela 83 na página 664 mostra onde os aplicativos de amostra estão instalados. As classes WebSphere MQ para amostras Java estão em um subdiretório chamado `wmqjava`. As amostras PCF estão em um subdiretório chamado `pcf`.

Tabela 83. Diretórios de amostras

Plataforma	**Diretório**
AIX	*MQ_INSTALLATION_PATH*/samp/wmqjava/
HP-UX, Linuxe Solaris	*MQ_INSTALLATION_PATH*/samp/wmqjava/
Windows	*MQ_INSTALLATION_PATH*\tools\wmqjava\
O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado.	

Variáveis de Ambiente Relevantes para Classes do WebSphere MQ para Java

Se você desejar executar as classes WebSphere MQ para aplicativos Java, seus caminhos de classe deverão incluir as classes WebSphere MQ para Java e os diretórios de amostras

Para que as classes WebSphere MQ para aplicativos Java sejam executadas, seu caminho de classe deve incluir as classes apropriadas do WebSphere MQ para o diretório Java Para executar os aplicativos de amostra, o caminho de classe também deve incluir os diretórios de amostras apropriados. Essas informações podem ser fornecidas no comando de chamada Java ou na variável de ambiente CLASSPATH

A Tabela 84 na página 665 mostra a configuração CLASSPATH apropriada a ser usada em cada plataforma para executar classes WebSphere MQ para aplicativos Java, incluindo os aplicativos de amostra.

Tabela 84. Configuração de CLASSPATH para executar classes WebSphere MQ para aplicativos Java, incluindo as classes WebSphere MQ para aplicativos de amostra Java	
Plataforma	**Configuração de CLASSPATH**
AIX	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:
HP-UX, Linuxe Solaris	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:
Windows	CLASSPATH=*MQ_INSTALLATION_PATH*\Java\lib\com.ibm.mq.jar; *MQ_INSTALLATION_PATH*\tools\wmqjava\samples;
O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado.	

Se você compilar usando a opção -Xlint, poderá ver uma mensagem que avise que o com.ibm.mq.ese.jar não está presente. É possível ignorar o aviso. Esse arquivo estará presente apenas se você tiver instalado o IBM WebSphere MQ Advanced Message Security

Os scripts fornecidos com as classes do WebSphere MQ para Java usam as seguintes variáveis de ambiente:

MQ_JAVA_DATA_PATH
Essa variável de ambiente especifica o diretório para log e saída de rastreio.

MQ_JAVA_INSTALL_PATH
Esta variável de ambiente especifica o diretório no qual as classes WebSphere MQ para Java são instaladas, conforme mostrado nas classes WebSphere MQ para diretórios de instalação Java.

MQ_JAVA_LIB_PATH
Esta variável de ambiente especifica o diretório no qual as classes WebSphere MQ para bibliotecas Java são armazenadas, conforme mostrado em O local das classes WebSphere MQ para bibliotecas Java para cada plataforma. Alguns scripts fornecidos com classes WebSphere MQ para Java, como IVTRun, usam essa variável de ambiente.

No Windows, todas as variáveis de ambiente são configuradas automaticamente durante a instalação Em qualquer outra plataforma, deve-se configurá-las sozinho.Em um sistema UNIX , é possível usar o script **setjmsenv** (se estiver usando uma JVM de 32 bits) ou **setjmsenv64** (se estiver usando uma JVM de 64 bits) para configurar as variáveis de ambiente. No AIX, HP-UX Linuxe Solaris, esses scripts estão no diretório *MQ_INSTALLATION_PATH*/java/bin .

As classes IBM WebSphere MQ para bibliotecas Java

O local das classes IBM WebSphere MQ para bibliotecas Java varia de acordo com a plataforma.. Especifique este local ao iniciar um aplicativo.

Para especificar o local das bibliotecas Java Native Interface (JNI), inicie seu aplicativo usando um comando **java** com o seguinte formato:

```
java -Djava.library.path=library_path application_name
```

em que *library_path* é o caminho para as classes do WebSphere MQ para bibliotecas Java, que incluem as bibliotecas JNI Tabela 85 na página 666 mostra o local das classes WebSphere MQ para bibliotecas Java para cada plataforma.

Tabela 85. O local das classes WebSphere MQ para bibliotecas Java para cada plataforma	
Plataforma	**Diretório que contém as classes do WebSphere MQ para bibliotecas Java.**
AIX	*MQ_INSTALLATION_PATH*/java/lib (bibliotecas de 32 bits) *MQ_INSTALLATION_PATH*/java/lib64 (bibliotecas de 64 bits)
HP-UX Linux (POWER, x86-64 e zSeries s390x plataformas) Solaris (plataformasx86-64 e SPARC)	*MQ_INSTALLATION_PATH*/java/lib (bibliotecas de 32 bits) *MQ_INSTALLATION_PATH*/java/lib64 (bibliotecas de 64 bits)
Linux (plataformax86)	*MQ_INSTALLATION_PATH*/java/lib
Windows	*MQ_INSTALLATION_PATH*\java\lib (bibliotecas de 32 bits) *MQ_INSTALLATION_PATH*\java\lib64 (bibliotecas de 64 bits)
O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado.	

Nota:

1. No AIX, HP-UX, Linux (plataforma Power) ou Solaris, use as bibliotecas de 32 bits ou as bibliotecas de 64 bits. Use as bibliotecas de 64 bits apenas se estiver executando seu aplicativo em uma Java virtual machine (JVM) de 64 bits em uma plataforma de 64 bits. Caso contrário, use as bibliotecas de 32 bits.
2. No Windows, é possível usar a variável de ambiente PATH para especificar o local das classes WebSphere MQ para bibliotecas Java em vez de especificar seu local no comando **java** .
3. Para usar classes WebSphere MQ para Java no modo de ligações em IBM i, assegure-se de que a biblioteca QMQMJAVA esteja em sua lista de bibliotecas.

Tarefas relacionadas

Usando classes do WebSphere MQ para Java

Suporte para OSGi em IBM WebSphere MQ classes for Java

OSGi fornece uma estrutura que suporta a implementação de aplicativos como pacotes configuráveis. Um pacote configurável OSGi é fornecido como parte do IBM WebSphere MQ classes for Java ..

O OSGi fornece uma estrutura Java de propósito geral, segura e gerenciada, que suporta a implementação de aplicativos fornecidos na forma de pacotes configuráveis. Os dispositivos compatíveis com o OSGi poderão fazer download e instalar pacotes configuráveis, e removê-los quando não forem mais necessários. A estrutura gerencia a instalação e a atualização de pacotes configuráveis de um modo dinâmico e escalável.

O IBM WebSphere MQ classes for Java.. inclui o seguinte pacote configurável OSGi.

com.ibm.mq.osgi.java_< version number> .jar
Os arquivos JAR para permitir que os aplicativos usem o IBM WebSphere MQ classes for Java.

em que < version number> é o número da versão do WebSphere MQ que foi instalado..

O pacote configurável é instalado no subdiretório `java/lib/OSGi` da instalação do IBM WebSphere MQ ou na pasta `java\lib\OSGi` no Windows.

Nove outros pacotes configuráveis também são instalados no subdiretório `java/lib/OSGi` da instalação do IBM WebSphere MQ ou na pasta `java\lib\OSGi` no Windows. Esses pacotes configuráveis são parte do IBM WebSphere MQ classes for JMS e não devem ser carregados em um ambiente de tempo de execução do OSGi que tenha o pacote configurável IBM WebSphere MQ classes for Java carregado. Se o pacote configurável OSGi IBM WebSphere MQ classes for Java for carregado em um ambiente de tempo de execução OSGi que também tenha os pacotes configuráveis IBM WebSphere MQ classes for JMS carregados, erros como:

```
java.lang.ClassCastException: com.ibm.mq.MQException incompatible with com.ibm.mq.MQException
```

ocorrerá quando os aplicativos usando o pacote configurável do IBM WebSphere MQ classes for Java ou os pacotes configuráveis do IBM WebSphere MQ classes for JMS forem executados.

O pacote configurável do OSGi para o IBM WebSphere MQ classes for Java foi gravado na especificação da liberação 4 do OSGi; ele não funciona em um ambiente de liberação 3 do OSGi.

Deve-se configurar seu caminho de sistema ou caminho da biblioteca corretamente para que o ambiente de tempo de execução do OSGi possa localizar quaisquer arquivos da DLL ou bibliotecas compartilhadas requeridas.

Se você usar o pacote configurável do OSGi para o IBM WebSphere MQ classes for Java, as classes de saída do canal gravadas em Java não serão suportadas devido a um problema inerente em classes de carregamento em um ambiente múltiplo de carregador de classes como OSGi. Um pacote configurável do usuário pode estar ciente do pacote configurável IBM WebSphere MQ classes for Java, mas o pacote configurável IBM WebSphere MQ classes for Java não está ciente de qualquer pacote configurável do usuário. Como resultado, o carregador de classes usado em um pacote configurável do IBM WebSphere MQ classes for Java não pode carregar uma classe de saída de canal que está em um pacote configurável do usuário...

Para obter mais informações sobre o OSGi, consulte o website do OSGi Alliance.

O arquivo de configuração IBM WebSphere MQ classes for Java

Um arquivo de configuração IBM WebSphere MQ classes for Java especifica propriedades que são utilizadas para configurar o IBM WebSphere MQ classes for Java.

O formato de um arquivo de configuração do IBM WebSphere MQ classes for Java é o de um arquivo de propriedades padrão do Java.

V 7.5.0.9 A partir do IBM WebSphere MQ Version 7.5.0, Fix Pack 9, um arquivo de configuração de amostra chamado `mqjava.config` é fornecido no subdiretório `bin` do diretório de instalação do IBM WebSphere MQ classes for Java . Este arquivo documenta todas as propriedades compatíveis e seus valores padrão.

Nota: O arquivo de configuração de amostra é sobrescrito quando a instalação do IBM WebSphere MQ é submetida a upgrade para um futuro Fix Pack. Portanto, é recomendável que você faça uma cópia do arquivo de configuração de amostra para uso com seus aplicativos.

É possível escolher o nome e o local de um arquivo de configuração do IBM WebSphere MQ classes for Java. Ao iniciar o seu aplicativo, use um comando **java** com o seguinte formato:

```
java -Dcom.ibm.msg.client.config.location=config_file_url application_name
```

No comando, *config_file_url* é um Localizador Uniforme de Recursos (URL) que especifica o nome e o local do arquivo de configuração do IBM WebSphere MQ classes for Java. As URLs dos tipos a seguir são suportadas: `http`, `file`, `ftp` e `jar`.

O exemplo a seguir mostra um comando **java**:

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/mqjava.config MyAppClass
```

Este comando identifica o arquivo de configuração do IBM WebSphere MQ classes for Java como o arquivo `D:\mydir\mqjava.config` no local do Windows do sistema.

Um arquivo de configuração do IBM WebSphere MQ classes for Java pode ser usado com qualquer um dos transportes suportados entre um aplicativo e um gerenciador de filas ou broker.

Substituindo propriedades especificadas em um arquivo de configuração do IBM WebSphere MQ classes for Java

Um arquivo de configuração do IBM WebSphere MQ MQI client também pode especificar propriedades que são usadas para configurar o IBM WebSphere MQ classes for Java. No entanto, as propriedades que são especificadas em um arquivo de configuração do IBM WebSphere MQ MQI client se aplicam somente quando um aplicativo se conecta a um gerenciador de filas no modo cliente.

Se necessário, será possível substituir qualquer atributo em um arquivo de configuração do IBM WebSphere MQ MQI client especificando-o como uma propriedade em um arquivo de configuração do IBM WebSphere MQ classes for Java. Para substituir um atributo em um arquivo de configuração do IBM WebSphere MQ MQI client, use uma entrada com o formato a seguir no arquivo de configuração do IBM WebSphere MQ classes for Java:

```
com.ibm.mq.cfg.stanza.propName=propValue
```

As variáveis na entrada possuem os seguintes significados:

stanza
O nome da sub-rotina no arquivo de configuração do IBM WebSphere MQ MQI client que contém o atributo.

propName
O nome do atributo, conforme especificado no arquivo de configuração do IBM WebSphere MQ MQI client.

propValue
O valor da propriedade que substitui o valor do atributo que é especificado no arquivo de configuração do IBM WebSphere MQ MQI client.

Como alternativa, é possível substituir um atributo em um arquivo de configuração do IBM WebSphere MQ MQI client, especificando a propriedade como uma propriedade de sistema no comando **java**. Use o formato anterior para especificar a propriedade como uma propriedade de sistema.

Somente os atributos a seguir em um arquivo de configuração do IBM WebSphere MQ MQI client são relevantes para IBM WebSphere MQ classes for Java. Se você especificar ou substituir outros atributos, isso não terá efeito. Especificamente, observe que `ChannelDefinitionFile` e `ChannelDefinitionDirectory` na sub-rotina CHANNELS do arquivo de configuração do cliente não são usados. Consulte “Usando uma tabela de definição de canal do cliente com o IBM WebSphere MQ classes for Java” na página 680 para obter detalhes de como usar o CCDT com o IBM WebSphere MQ classes for Java.

Tabela 86. Qual sub-rotina do arquivo de configuração do cliente contém qual atributo

Sub-rotina	**Atribuir**
Sub-rotina ClientExitPath do arquivo de configuração do cliente	Caminho Padrão das Saídas
Sub-rotina ClientExitPath do arquivo de configuração do cliente	ExitsDefaultPath64
Sub-rotina ClientExitPath do arquivo de configuração do cliente	JavaExitsClasspath
Sub-rotina MessageBuffer do arquivo de configuração do cliente	MaximumSize
Sub-rotina MessageBuffer do arquivo de configuração do cliente	PurgeTime

Tabela 86. Qual sub-rotina do arquivo de configuração do cliente contém qual atributo (continuação)	
Sub-rotina	**Atribuir**
Sub-rotina MessageBuffer do arquivo de configuração do cliente	UpdatePercentage
Sub-rotina TCP do Arquivo de Configuração do Cliente	ClntRcvBufSize
Sub-rotina TCP do Arquivo de Configuração do Cliente	ClntSndBufSize
Sub-rotina TCP do Arquivo de Configuração do Cliente	Connect_Timeout
Sub-rotina TCP do Arquivo de Configuração do Cliente	KeepAlive

Para obter mais informações sobre a configuração do IBM WebSphere MQ MQI client, veja Configurando um cliente usando um arquivo de configuração.

Tarefas relacionadas

Rastreando aplicativos IBM WebSphere MQ classes for Java

Sub-rotina de rastreio do ambiente Java Standard

É possível usar a sub-rotina Configurações de rastreio do ambiente padrão do Java para configurar o recurso de rastreio do IBM WebSphere MQ classes for Java .

com.ibm.msg.client.commonservices.trace.outputName = `traceOutputName`

`traceOutputName` é o nome do diretório e do arquivo para o qual a saída de rastreio é enviada

O nome padrão do arquivo de rastreio depende da versão do IBM WebSphere MQ classes for Java que está sendo usada por um aplicativo:

- Para IBM WebSphere MQ classes for Java para Version 7.5.0, Fix Pack 8 ou anterior, *traceOutputName* é padronizado para um arquivo denominado `mqjms_%PID%.trc` no diretório atualmente em funcionamento.
- V 7.5.0.9 Em IBM WebSphere MQ classes for Java de Version 7.5.0, Fix Pack 9, *traceOutputName* é padronizado para um arquivo denominado `mqjava_%PID%.trc` no diretório atualmente em funcionamento.

em que *%PID%* é o ID do processo atual. Se um ID de processo estiver indisponível, um número aleatório será gerado e prefixado com a letra `f`. Para incluir o ID do processo em um nome de arquivo especificado, use a sequência `%PID%` .

Se você especificar um diretório alternativo, ele deve existir e você deve ter permissão de gravação para esse diretório. Se você não tiver permissão de gravação, a saída de rastreio será gravada para `System.err`.

com.ibm.msg.client.commonservices.trace.include = `includeList`

`includeList` é uma lista de pacotes e classes rastreados ou os valores especiais `ALL` ou `NONE`.

Separe os nomes de pacote ou classe com um ponto e vírgula (`;`). O **`includeList`** é padronizado como `ALL`e rastreia todos os pacotes e classes no IBM WebSphere MQ classes for Java

Nota: É possível incluir um pacote, mas depois excluir os subpacotes desse pacote. Por exemplo, se você incluir o pacote `a.b` e excluir o pacote `a.b.x`, o rastreio inclui tudo no `a.b.y` e `a.b.z`, mas não `a.b.x` ou `a.b.x.1`.

com.ibm.msg.client.commonservices.trace.exclude = `excludeList`

`excludeList` é uma lista de pacotes e classes não rastreados ou os valores especiais `ALL` ou `NONE`.

Separe os nomes de pacote ou classe com um ponto e vírgula (;). O **excludeList** é padronizado como NONEe, portanto, não exclui nenhum pacote e classe no IBM WebSphere MQ classes for Java de ser rastreado

Nota: É possível excluir um pacote, mas depois incluir os subpacotes desse pacote. Por exemplo, se você excluir o pacote a.b e incluir o pacote a.b.x, o rastreio incluirá tudo em a.b.x e a.b.x.1, mas não a.b.y ou a.b.z.

Qualquer pacote ou classe que estiver especificado, no mesmo nível, como ambos incluídos e excluídos, será incluído.

com.ibm.msg.client.commonservices.trace.maxBytes = maxArrayBytes
maxArrayBytes é o número máximo de bytes que são rastreados de quaisquer matrizes de bytes.

Se **maxArrayBytes** for configurado para um número inteiro positivo, ele limitará o número de bytes em uma matriz de bytes que são gravados no arquivo de rastreio. Ele trunca a matriz de bytes após gravar *maxArrayBytes* para fora. Configurar **maxArrayBytes** reduz o tamanho do arquivo de rastreio resultante e reduz o efeito do rastreio no desempenho do aplicativo.

Um valor 0 para esta propriedade significa que nenhum conteúdo de qualquer matriz de bytes é enviado para o arquivo de rastreio.

O valor padrão é -1, o que remove qualquer limite no número de bytes em uma matriz de bytes que são enviados para o arquivo de rastreio.

com.ibm.msg.client.commonservices.trace.limit = maxTraceBytes
maxTraceBytes é o número máximo de bytes gravados em um arquivo de saída de rastreio.

maxTraceBytes funciona com **traceCycles**. Se o número de bytes de rastreio gravado estiver perto do limite, o arquivo será fechado e um novo arquivo de saída de rastreio será iniciado.

Um valor 0 significa que um arquivo de saída de rastreio tem o comprimento zero. O valor padrão é -1, o que significa que a quantidade de dados a serem gravados em um arquivo de saída de rastreio é ilimitada.

com.ibm.msg.client.commonservices.trace.count = traceCycles
traceCycles é o número de arquivos de saída de rastreio para percorrer.

Se o arquivo de saída de rastreio atual atingir o limite especificado por **maxTraceBytes**, o arquivo será fechado. A saída de rastreio adicional é gravada para o próximo arquivo de saída de rastreio na sequência. Cada arquivo de saída de rastreio é distinguido por um sufixo numérico anexado ao nome do arquivo. O arquivo de saída de rastreio atual ou mais recente possui o sufixo .trc.0, o próximo arquivo de saída de rastreio mais recente termina com .trc.1, e assim por diante Os arquivos de rastreio seguem o mesmo padrão de numeração até o limite.

O valor padrão de **traceCycles** é 1. Se **traceCycles** for 1, quando o arquivo de saída de rastreio atual atingir seu tamanho máximo, o arquivo será fechado e excluído. Um novo arquivo de saída de rastreio com o mesmo nome é iniciado. Portanto, existe só um arquivo de saída de rastreio por vez.

com.ibm.msg.client.commonservices.trace.parameter = traceParameters
traceParameters controla se os parâmetros de método e os valores de retorno são incluídos no rastreamento

traceParameters assume como padrão TRUE. Se **traceParameters** for configurado como FALSE, apenas as assinaturas de método serão rastreados

com.ibm.msg.client.commonservices.trace.compress = compressedTrace
Configure **compressedTrace** como TRUE para compactar a saída de rastreio

O valor padrão de **compressedTrace** é FALSE.

Se **compressedTrace** for configurado como TRUE, a saída de rastreio será compactado O nome do arquivo de saída de rastreio padrão tem a extensão .trz. Se a compactação estiver configurada como FALSE, o valor padrão, o arquivo terá a extensão .trc para indicar que está descompactado. No entanto, se o nome do arquivo para a saída de rastreio for especificado em **traceOutputName** , esse nome será usado e nenhum sufixo será aplicado ao arquivo.

A saída de rastreio compactada é menor do que a descompactada. Como há menos E/S, isso pode ser gravado mais rápido do que o rastreio descompactado. O rastreio compactado tem menos efeito no desempenho do IBM WebSphere MQ classes for Java do que o rastreio descompactado

Se **maxTraceBytes** e **traceCycles** forem configurados, diversos arquivos de rastreamento compactado serão criados no lugar de vários arquivos simples

Se o IBM WebSphere MQ classes for Java terminar de uma maneira não controlada, um arquivo de rastreio compactado poderá não ser válido Por essa razão, a compactação de rastreio deve ser usada apenas quando o IBM WebSphere MQ classes for Java for fechado de uma maneira controlada Utilize a compactação de rastreio apenas se os problemas que estão sendo investigados não fizerem com que a JVM em si pare inesperadamente Não use a compactação de rastreio quando estiver diagnosticando problemas que podem resultar em encerramentos do `System.Halt()` ou finalizações anormais e não controladas da JVM

com.ibm.msg.client.commonservices.trace.level = traceLevel

traceLevel especifica um nível de filtragem para o rastreio.. Os níveis de rastreio definidos são os seguintes:

Tabela 87. O que é rastreado para cada nível de rastreio

Value	O que é rastreado
0	O rastreio está desativado
1	Exceções
3	Exceções Avisos
6	Exceções Avisos Pontos de rastreio informativos
8	Exceções Avisos Pontos de rastreio informativos Entrada e Saída do Método
9	Exceções Avisos Pontos de rastreio informativos Entrada e Saída do Método Os dados que são enviados entre o IBM WebSphere MQ classes for Java e um gerenciador de filas.

Nota: Sempre use o valor 9 , a menos que orientado de outra forma pelo Suporte IBM .

IBM WebSphere MQ classes for Java e ferramentas de gerenciamento de software

Ferramentas de gerenciamento de software como Apache Maven podem ser usadas com o IBM WebSphere MQ classes for Java.

Muitas das grandes organizações de desenvolvimento usam essas ferramentas para gerenciar centralmente os repositórios de bibliotecas de terceiros.

Os IBM WebSphere MQ classes for Java são compostos por um número de arquivos JAR. Quando você está desenvolvendo aplicativos de linguagem Java usando essa API, uma instalação de um IBM WebSphere MQ Servidor, IBM WebSphere MQ Cliente ou IBM WebSphere MQ Cliente SupportPac é necessária na máquina na qual o aplicativo está sendo desenvolvido

Se desejar usar uma ferramenta de gerenciamento de software e incluir os arquivos JAR que formam o IBM WebSphere MQ classes for Java para um repositório gerenciado centralmente, os seguintes pontos deverão ser observados:

- Um repositório ou contêiner deve ser disponibilizado somente para os desenvolvedores em sua organização. Qualquer distribuição fora da organização não é permitida.
- O repositório precisa conter um conjunto completo e consistente de arquivos JAR a partir de uma liberação única ou fix pack do IBM WebSphere MQ.
- Você é responsável por atualizar o repositório com qualquer manutenção fornecida pelo Suporte do IBM

Para o IBM WebSphere MQ Version 7.5, os seguintes arquivos JAR precisam ser instalados no repositório:

- com.ibm.mq.commonservices.jar
- com.ibm.mq.jar
- com.ibm.mq.jmqi.jar
- com.ibm.mq.pcf.jar
- com.ibm.mq.headers.jar
- connector.jar

Configuração de pós-instalação para os aplicativos IBM WebSphere MQ

Após instalar o IBM WebSphere MQ, será possível configurar sua instalação para que seus próprios aplicativos sejam executados.

Lembre-se de verificar o arquivo LEIA-ME do IBM WebSphere MQ para obter informações posteriores ou mais específicas para seu ambiente.

Antes de tentar executar um aplicativo IBM WebSphere MQ classes para Java no modo de ligações, certifique-se de ter configurado IBM WebSphere MQ conforme descrito em Configurando .

Configurando seu gerenciador de filas para aceitar conexões do cliente a partir das classes WebSphere MQ para Java

Para configurar seu gerenciador de filas para aceitar pedidos de conexão que chegam de clientes, definir e permitir a utilização de um canal de conexão do servidor e iniciar um programa listener.

Consulte “Preparando e executando os programas de amostra” na página 111 para obter detalhes.

Executando classes do WebSphere MQ para aplicativos Java no Java Security Manager

As classes WebSphere MQ para Java podem ser executadas com o Java Security Manager ativado. Para executar aplicativos com êxito com o Security Manager ativado, deve-se configurar a Java virtual machine (JVM) com um arquivo de definição de política adequado.

A maneira mais simples de fazer isso é alterar o arquivo de políticas fornecido com seu JRE Na maioria dos sistemas, esse arquivo é armazenado no caminho `lib/security/java.policy`, relativo ao seu diretório JRE É possível editar os arquivos de política usando seu editor preferido ou o programa `policytool` fornecido com seu JRE.

Você deve fornecer autoridade para o arquivo com.ibm.mq.jmqi.jar para que ele possa:

- Criar soquetes (no modo cliente)
- Carregara biblioteca nativa (no modo de ligações)
- Ler várias propriedades a partir do ambiente

A propriedade de sistema os.name deve estar disponível para as classes do WebSphere MQ para Java ao executar no Java Security Manager.

Aqui está um exemplo de uma entrada de arquivo de políticas que permite que as classes do WebSphere MQ para Java sejam executadas com sucesso no gerenciador de segurança padrão:

```
grant codeBase "file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.jmqi.jar" {
  //Required
  permission java.util.PropertyPermission "user.name","read";
  permission java.util.PropertyPermission "os.name","read";
  //Required if mqclient.ini/mqs.ini configuration files are used
  permission java.io.FilePermission "/var/mqm/mqclient.ini","read";
  permission java.io.FilePermission "/var/mqm/mqs.ini","read";
  //For the client transport type.
  permission java.net.SocketPermission "*","connect";
  //For the bindings transport type.
  permission java.lang.RuntimePermission "loadLibrary.*";
  //For applications that use CCDT tables (access to the CCDT
AMQCLCHL.TAB)
  permission java.io.FilePermission
"/var/mqm/qmgrs/QMGR/@ipcc/AMQCLCHL.TAB","read";
  //For applications that use User Exits
  permission java.io.FilePermission "/var/mqm/exits/*","read";
  permission java.lang.RuntimePermission "createClassLoader";
  //Required for the z/OS platform
  permission java.util.PropertyPermission
"com.ibm.vm.bitmode","read";
};
grant codeBase
"file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.commonservices.jar" {
  permission java.util.PropertyPermission "user.dir","read";
  permission java.util.PropertyPermission "line.separator","read";
  //tracing permissions
  permission java.util.PropertyPermission "com.ibm.mq.commonservices", "read";
  permission java.util.logging.LoggingPermission "control";
  //For access to the trace properties file.
  permission java.io.FilePermission "/tmp/trace.properties", "read";
  //For access to the trace output files.
  permission java.io.FilePermission "/tmp/*", "read,write";
};
```

Notes:

- O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado.
- Este exemplo de um arquivo de políticas permite que as classes do WebSphere MQ para Java funcionem corretamente sob o gerenciador de segurança, mas ainda pode ser necessário ativar seu próprio código para executar corretamente antes que seus aplicativos funcionem.
- Para permitir que as classes do WebSphere MQ para Java acessem os arquivos Java archive (JAR) de um aplicativo, inclua a permissão a seguir na primeira instrução `grant` :

  ```
  permission java.io.FilePermission "/path_to_your_app/-", "read";
  ```

- Para usar essas instruções `grant` em seu arquivo de configuração de política, pode ser necessário modificar os nomes de caminho dependendo de onde você instalou as classes do WebSphere MQ para Java e onde você armazena seus aplicativos.
- O código de amostra fornecido com classes WebSphere MQ para Java não foi especificamente ativado para uso com o gerenciador de segurança; no entanto, os testes IVT são executados com esse arquivo de políticas e o gerenciador de segurança padrão em vigor.

Verificando as classes IBM WebSphere MQ para a instalação do Java

Um programa de verificação de instalação, MQIVP, é fornecido com as classes IBM WebSphere MQ para Java É possível usar esse programa para testar todos os modos de conexão de classes IBM WebSphere MQ para Java

O programa solicita diversas opções e outros dados para determinar qual modo de conexão você deseja verificar. Use o procedimento a seguir para verificar sua instalação:

1. Se for executar o programa no modo cliente, configure seu gerenciador de filas conforme descrito em “Preparando e executando os programas de amostra” na página 111. A fila a ser usada é SYSTEM.DEFAULT.LOCAL.QUEUE.
2. Se for executar o programa no modo cliente, consulte também “Usando classes do WebSphere MQ para Java” na página 660.

   Execute as etapas restantes deste procedimento no sistema no qual irá executar o programa.
3. Certifique-se de que você tenha atualizado sua variável de ambiente CLASSPATH de acordo com as instruções em “Variáveis de Ambiente Relevantes para Classes do WebSphere MQ para Java” na página 664.
4. Mude o Diretório para `MQ_INSTALLATION_PATH/mqm/VRM/java/samples/wmqjava`, em que MQ_INSTALLATION_PATH é o caminho para sua IBM WebSphere MQ instalação e `VRM` é a versão, a liberação e o número de modificação do produto. Em seguida, no prompt de comandos, insira:

   ```
   java -Djava.library.path=library_path MQIVP
   ```

   em que *library_path* é o caminho para as classes IBM WebSphere MQ para bibliotecas Java (consulte As classes WebSphere MQ para bibliotecas Java).

   No prompt marcado (1):

   - Para usar uma conexão TCP/IP, insira um nome do servidor host IBM WebSphere MQ .
   - Para usar conexão nativa (modo de ligações), deixe o campo em branco (não insira um nome).

   O programa tenta:

   - 1. Conectar-se ao gerenciador de filas
   - 2. Abra a fila SYSTEM.DEFAULT.LOCAL.QUEUE, coloque uma mensagem na fila, obtenha uma mensagem da fila e, em seguida, feche a fila
   - 3. Desconecte do gerenciador de filas
   - 4. Retorne uma mensagem se as operações forem bem-sucedidas

Aqui está um exemplo de prompts e respostas que você pode ver. Os prompts reais e suas respostas dependem da sua rede do IBM WebSphere MQ.

```
Please enter the IP address of the MQ server              : ipaddress(1)
Please enter the port to connect to                       : (1414)(2)
Please enter the server connection channel name            : channelname(2)
Please enter the queue manager name                       : qmname
Success: Connected to queue manager.
Success: Opened SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Put a message to SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Got a message from SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Closed SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Disconnected from queue manager


Tests complete -
SUCCESS: This MQ Transport is functioning correctly.
Press Enter to continue ...
```

Nota:

1. Se você escolher conexão do servidor, não verá os prompts marcados como (2).

Resolvendo problemas do IBM WebSphere MQ

Inicialmente, execute o programa de verificação da instalação. Também pode ser necessário usar o recurso de rastreio.

Se um programa não for concluído com êxito, execute o programa de verificação da instalação e siga o conselho fornecido nas mensagens de diagnóstico. Este programa está descrito em “Verificando as classes IBM WebSphere MQ para a instalação do Java” na página 673.

Se os problemas continuarem e for necessário contatar a equipe de serviço da IBM, poderá ser solicitado que você ative o recurso de rastreio. Faça isso conforme mostrado no exemplo a seguir.

Para rastrear o programa MQIVP:

- Crie um arquivo de propriedades *com.ibm.mq.commonservices* (consulte Usando com.ibm.mq.commonservices)
- Insira o seguinte comando:

```
java -Dcom.ibm.mq.commonservices=commonservices_properties_file java
-Djava.library.path=library_path MQIVP -trace
```

em que:

- *commonservices_properties_file* é o caminho (incluindo o nome do arquivo) para o arquivo de propriedades do *com.ibm.mq.commonservices*.
- *library_path* é o caminho para as classes WebSphere MQ para bibliotecas Java (consulte As classes WebSphere MQ para bibliotecas Java).

Para obter mais informações sobre como usar o rastreio, consulte Rastreando aplicativos IBM WebSphere MQ classes for Java.

Introdução para Programadores

Esta coleção dos tópicos contém informações gerais para programadores..

Para obter informações mais detalhadas sobre como gravar programas, consulte “Gravando classes do WebSphere MQ para aplicativos Java” na página 675.

As classes do WebSphere MQ para interface Java

A interface de programação de aplicativos do WebSphere MQ processual usa verbos, que atuam em objetos A interface de programação Java usa objetos, nos quais você atua chamando métodos.

A interface de programação de aplicativos processual do WebSphere MQ é construída em torno de verbos como estes:

```
MQBACK, MQBEGIN, MQCLOSE, MQCONN, MQDISC,
MQGET, MQINQ, MQOPEN, MQPUT, MQSET, MQSUB
```

Todos estes verbos utilizam, como um parâmetro, um identificador para o objeto do WebSphere MQ no qual eles devem operar. Como Java é orientado a objetos, a interface de programação Java faz essa rodada. Seu programa consiste em um conjunto de objetos do WebSphere MQ, sob o qual você atua chamando métodos nestes objetos.

Ao usar a interface processual, desconecte-se de um gerenciador de filas usando a chamada MQDISC(Hconn, CompCode, Reason), em que *Hconn* é um identificador para o gerenciador de filas.

Na interface Java, o gerenciador de fila é representado por um objeto de classe MQQueueManager.. Desconecte-se do gerenciador de filas chamando o método disconnect() nessa classe.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.disconnect();
```

Gravando classes do WebSphere MQ para aplicativos Java

Esta coleção de tópicos fornece informações para ajudar a gravar aplicativos Java para interagir com sistemas WebSphere MQ .

Para usar as classes do WebSphere MQ para Java para acessar filas do WebSphere MQ , você grava aplicativos Java que contêm chamadas que colocam mensagens e obtêm mensagens de filas do WebSphere MQ . Para obter detalhes de classes individuais, consulte WebSphere MQ classes para Java..

Nota: A reconexão automática do cliente não é suportada pelas classes do WebSphere MQ para Java

WebSphere MQ classes para modos de conexão Java

A maneira como você programa para classes WebSphere MQ para Java tem algumas dependências nos modos de conexão que deseja usar.

Se você usar conexões do cliente, há uma série de diferenças do IBM WebSphere MQ MQI client, mas é conceitualmente semelhante. Se usar o modo de ligações, será possível usar ligações de atalho e poderá emitir o comando MQBEGIN. Você especifica qual modo usar configurando variáveis na classe MQEnvironment.

WebSphere MQ classes para conexões do cliente Java

Quando as classes WebSphere MQ para Java são usadas como um cliente, é como o IBM WebSphere MQ MQI client, mas tem várias diferenças.

Se você estiver programando para *WebSphere MQ classes para Java* para uso como um cliente, esteja ciente das diferenças a seguir:

- Ele suporta apenas TCP/IP.
- Ele não lê nenhuma variável de ambiente WebSphere MQ na inicialização.
- Informações que seriam armazenadas em uma definição de canal e nas variáveis de ambiente podem ser armazenadas em uma classe chamada Environment. Como alternativa, estas informações podem ser transmitidos como parâmetros quando a conexão for feita.
- Condições de erro e exceção são gravadas em um log especificado na classe `MQException`. O destino do erro padrão é o console Java
- Apenas os atributos a seguir em um arquivo de configuração do cliente WebSphere MQ são relevantes para as classes do WebSphere MQ para Java. Se você especificar outros atributos, são ineficazes.

Sub-rotina	**Atribuir**
Sub-rotina ClientExitPath do arquivo de configuração do cliente	Caminho Padrão das Saídas
Sub-rotina ClientExitPath do arquivo de configuração do cliente	ExitsDefaultPath64
Sub-rotina ClientExitPath do arquivo de configuração do cliente	JavaExitsClasspath
Sub-rotina MessageBuffer do arquivo de configuração do cliente	MaximumSize
Sub-rotina MessageBuffer do arquivo de configuração do cliente	PurgeTime
Sub-rotina MessageBuffer do arquivo de configuração do cliente	UpdatePercentage
Sub-rotina TCP do Arquivo de Configuração do Cliente	ClntRcvBufSize
Sub-rotina TCP do Arquivo de Configuração do Cliente	ClntSndBufSize
Sub-rotina TCP do Arquivo de Configuração do Cliente	Connect_Timeout

Sub-rotina	Atribuir
Sub-rotina TCP do Arquivo de Configuração do Cliente	KeepAlive

- Se estiver se conectando a um gerenciador de filas que requer que dados de caractere sejam convertidos, o cliente Java V7 agora será capaz de fazer a conversão se o gerenciador de filas não puder fazer isso. A JVM do cliente deve suportar a conversão entre o CCSID do cliente e que do gerenciador de filas.
- A reconexão do cliente automática não é suportada pelas classes do WebSphere MQ para Java.

Quando usado no modo cliente, as classes *WebSphere MQ para Java* não suportam a chamada MQBEGIN

Consulte “Opções de Conexão para Classes WebSphere MQ para Java” na página 661 para obter informações adicionais sobre os ambientes suportados

WebSphere MQ classes para o modo de ligações Java

O modo de ligações das classes WebSphere MQ para Java difere do modo cliente de três maneiras principais.

Quando usado no modo de ligações, as classes do WebSphere MQ para Java usam a Java Native Interface (JNI) para chamar diretamente na API do gerenciador de filas existente, em vez de se comunicarem por meio de uma rede

Por padrão, os aplicativos que usam as classes WebSphere MQ para Java no modo de ligações se conectam a um gerenciador de filas usando *ConnectOption*, MQCNO_STANDARD_BINDINGS.

As classes WebSphere MQ para Java suportam o seguinte *ConnectOptions*:

- MQCNO_FASTPATH_BINDING
- MQCNO_STANDARD_BINDING
- MQCNO_SHARED_BINDING
- MQCNO_ISOLATED_BINDING

Para obter informações adicionais sobre *ConnectOptions*, consulte “Conectando-se a um gerenciador de filas usando a chamada MQCONNX” na página 212.

O modo de ligações suporta a chamada MQBEGIN para iniciar unidades globais de trabalho que são coordenadas pelo gerenciador de filas em todas as plataformas, exceto WebSphere MQ para IBM i e WebSphere MQ para z/OS.

A maioria dos parâmetros fornecidos pela classe `MQEnvironment` não são relevantes para o modo de ligações e são ignoradas.

Consulte “Opções de Conexão para Classes WebSphere MQ para Java” na página 661 para obter informações adicionais sobre os ambientes suportados

Definindo quais classes do WebSphere MQ para conexão Java usar

O tipo de conexão a ser usado é determinado pela configuração de variáveis na classe MQEnvironment.

Duas variáveis são usadas:

MQEnvironment.properties
O tipo de conexão é determinado pelo valor associado com o nome da chave `CMQC.TRANSPORT_PROPERTY`. Os valores possíveis são os seguintes:

CMQC.TRANSPORT_MQSERIES_BINDINGS
Conectar no modo de ligações

CMQC.TRANSPORT_MQSERIES_CLIENT
Conectar no modo cliente

CMQC.TRANSPORT_MQSERIES
O modo de conexão é determinado pelo valor da propriedade *hostname*

MQEnvironment.hostname
Configure o valor dessa variável como segue:

- Para conexões do cliente, configure o valor dessa variável para o nome do host do servidor IBM WebSphere MQ ao qual deseja conectar-se
- Para o modo de ligações não configure essa variável ou configure-a como nulo

Operações nos Gerenciadores de Filas

Esta coleção de tópicos descreve como se conectar e desconectar de um gerenciador de filas usando classes WebSphere MQ para Java.

Configurando o ambiente do WebSphere MQ para classes do WebSphere MQ para Java

Para que um aplicativo estabeleça conexão com um gerenciador de filas no modo cliente, o aplicativo deve especificar o nome do canal, o nome do host e o número da porta.

Nota: As informações neste tópico são relevantes apenas se o seu aplicativo se conectar a um gerenciador de filas no modo cliente. Ele *não* é relevante se o aplicativo se conectar no modo de ligações. Consulte: "Modos de Conexão para Classes do WebSphere MQ para JMS" na página 783

É possível especificar o nome do canal, o nome do host e o número da porta de uma das duas maneiras a seguir: como campos na classe MQEnvironment ou como propriedades do objeto MQQueueManager.

Se você configurar campos na classe MQEnvironment, eles se aplicarão ao seu aplicativo inteiro, exceto onde são substituídos por uma tabela hash de propriedades. Para especificar o nome do canal e o nome do host em MQEnvironment, use o código a seguir:

```
MQEnvironment.hostname = "host.domain.com";
MQEnvironment.channel  = "java.client.channel";
```

Isto é equivalente à configuração de uma variável de ambiente **MQSERVER**:

```
"java.client.channel/TCP/host.domain.com".
```

Por padrão, os clientes Java tentam se conectar a um listener WebSphere MQ na porta 1414. Para especificar uma porta diferente, use o código a seguir:

```
MQEnvironment.port = nnnn;
```

em que nnnn é o número da porta requerido

Se você passar propriedades para um objeto do gerenciador de filas em sua criação, elas se aplicarão apenas a esse gerenciador de filas. Crie entradas em um objeto Hashtable com chaves de **hostname**, **channel** e, opcionalmente, **port**, e com valores apropriados. Para usar a porta padrão, 1414, é possível omitir a entrada **port**. Crie o objeto MQQueueManager usando um construtor que aceita a tabela hash de propriedades.

Identificando uma conexão com o gerenciador de filas configurando um nome de aplicativo

Um aplicativo pode configurar um nome que identifica sua conexão com o gerenciador de filas. Esse nome do aplicativo é mostrado pelo comando **DISPLAY CONN MQSC/PCF** (em que o campo é chamado **APPLTAG**) ou na exibição WebSphere MQ Explorer **Conexões de Aplicativos** (em que o campo é chamado **App name**).

Os nomes de aplicativos são limitados a 28 caracteres, e nomes mais longos são truncados para ajuste. Se um nome de aplicativo não for especificado, um padrão será fornecido. O nome padrão tem como base a classe de chamada (principal), mas se esta informação não estiver disponível, o texto `WebSphere MQ Client for Java` é usado.

Se o nome da classe de chamada for usado, ele será ajustado removendo os nomes de pacotes iniciais, se necessário. Por exemplo, se a classe de chamada for `com.example.MainApp`, o nome completo será usado, mas se a classe de chamada for `com.example.dictionaryAndThesaurus.multilingual.mainApp`, o nome `multilingual.mainApp` será usado, pois ele é a combinação mais longa do nome da classe e o nome do pacote mais à direita que cabe no comprimento disponível.

Se o próprio nome da classe tiver mais de 28 caracteres de comprimento, ele será truncado para ajuste. Por exemplo, `com.example.mainApplicationForSecondTestCase` se torna `mainApplicationForSecondTest`.

Nota: Os gerenciadores de filas em execução nas plataformas z/OS não suportam a configuração de nomes de aplicativos

Para configurar um nome de aplicativo na classe MQEnvironment, inclua o nome na tabela hash MQEnvironment.properties, com uma chave de **`MQConstants.APPNAME_PROPERTY`**, usando o código a seguir:

```
MQEnvironment.properties.put(MQConstants.APPNAME_PROPERTY, "my_application_name");
```

Para configurar um nome de aplicativo na tabela hash de propriedades que é passada para o construtor MQQueueManager, inclua o nome na tabela hash de propriedades com uma chave de **`MQConstants.APPNAME_PROPERTY`**.

Substituindo propriedades especificadas em um arquivo de configuração do cliente WebSphere MQ

Um arquivo de configuração do cliente WebSphere MQ também pode especificar propriedades que são usadas para configurar classes WebSphere MQ para Java. No entanto, as propriedades especificadas em um arquivo de configuração do cliente MQI do WebSphere MQ se aplicam apenas quando um aplicativo se conecta a um gerenciador de filas no modo cliente.

Se necessário, é possível substituir qualquer atributo em um arquivo de configuração do WebSphere MQ de qualquer uma das maneiras a seguir. As opções são mostradas em ordem de precedência.

- Configure uma propriedade de sistema Java para a propriedade de configuração
- Configure a propriedade no mapa MQEnvironment.properties.
- Em Java5 e liberações posteriores, configure uma variável de ambiente do sistema.

Somente os atributos a seguir em um arquivo de configuração do cliente WebSphere MQ são relevantes para as classes do WebSphere MQ para Java Se você especificar ou substituir outros atributos, isso não terá efeito.

Sub-rotina	**Atribuir**
Sub-rotina ClientExitPath do arquivo de configuração do cliente	Caminho Padrão das Saídas
Sub-rotina ClientExitPath do arquivo de configuração do cliente	ExitsDefaultPath64
Sub-rotina ClientExitPath do arquivo de configuração do cliente	JavaExitsClasspath
Sub-rotina MessageBuffer do arquivo de configuração do cliente	MaximumSize
Sub-rotina MessageBuffer do arquivo de configuração do cliente	PurgeTime
Sub-rotina MessageBuffer do arquivo de configuração do cliente	UpdatePercentage

Sub-rotina	Atribuir
Sub-rotina TCP do Arquivo de Configuração do Cliente	ClntRcvBufSize
Sub-rotina TCP do Arquivo de Configuração do Cliente	ClntSndBufSize
Sub-rotina TCP do Arquivo de Configuração do Cliente	Connect_Timeout
Sub-rotina TCP do Arquivo de Configuração do Cliente	KeepAlive

Conectando-se a um gerenciador de filas no WebSphere MQ classes para Java

Conecte-se a um gerenciador de filas criando uma nova instância da classe MQQueueManager. Desconecte-se de um gerenciador de filas chamando o método disconnect().

Agora você está pronto para conectar-se a um gerenciador de filas criando uma nova instância da classe MQQueueManager:

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Para desconectar-se de um gerenciador de filas, chame o método disconnect() no gerenciador de filas:

```
queueManager.disconnect();
```

Se você chamar o método disconnect, todas as filas e processos abertos que você acessou usando esse gerenciador de filas serão encerrados. Entretanto, é uma boa prática de programação fechar estes recursos explicitamente quando você terminar de usá-los. Para fazer isso, use o método close() nos objetos relevantes.

O commit() e métodos de recuperação() em um gerenciador de filas são equivalentes às chamadas MQCMIT e MQBACK que são usadas com a interface processual.

Usando uma tabela de definição de canal do cliente com o IBM WebSphere MQ classes for Java

Uma classe IBM WebSphere MQ classes for Java aplicativo cliente pode usar definições de canal de conexão do cliente armazenadas em uma tabela de definição de canal do cliente (CCDT).

Como uma alternativa para criar uma definição de canal de conexão do cliente configurando determinados campos e propriedades de ambiente na classe `MQEnvironment` ou passando-os para um `MQQueueManager` em uma tabela hash de propriedades, um aplicativo cliente IBM WebSphere MQ classes for Java pode usar definições de canal de conexão do cliente que são armazenadas em uma tabela de definição de canal do cliente. Essas definições são criadas pelos comandos do IBM WebSphere MQ Script (MQSC) ou os comandos do IBM WebSphere MQ Programmable Command Format (PCF) ou usando o IBM WebSphere MQ Explorer.

Quando o aplicativo cria um objeto `MQQueueManager` , as classes IBM WebSphere MQ classes for Java procuram na tabela de definição de canal do cliente uma definição de canal de conexão do cliente adequada e usam a definição de canal para iniciar um canal MQI.. Para obter mais informações sobre tabelas de definição de canal do cliente e como construir um, consulte Client Channel Definition Table.

Para usar uma tabela de definição de canal do cliente, um aplicativo deve primeiramente criar um objeto de URL. O objeto de URL contém um localizador uniforme de recursos (URL) que identifica o nome e o local do arquivo contendo a tabela de definição de canal do cliente e especifica como o arquivo pode ser acessado.

Por exemplo, se o arquivo `ccdt1.tab` contiver uma tabela de definição de canal do cliente e for armazenado no mesmo sistema no qual o aplicativo está em execução, o aplicativo poderá criar um objeto de URL da seguinte maneira:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
```

Como outro exemplo, suponha que o arquivo `ccdt2.tab` contenha uma tabela de definição de canal do cliente e esteja armazenada em um sistema diferente daquele no qual o aplicativo está em execução. Se o arquivo puder ser acessado usando o protocolo FTP, o aplicativo poderá criar um objeto de URL da seguinte maneira:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
```

Depois que o aplicativo criou um objeto de URL, o aplicativo poderá criar um objeto `MQQueueManager` usando um dos construtores que obtém um objeto de URL como um parâmetro. Aqui está um exemplo:

```
MQQueueManager mars = new MQQueueManager("MARS", chanTab2);
```

Essa instrução faz com que as classes IBM WebSphere MQ classes for Java acessem a tabela de definição de canal do cliente identificada pelo objeto de URL chanTab2, procure a tabela para uma definição de canal de conexão do cliente adequada e, em seguida, use a definição de canal para iniciar um canal MQI para o gerenciador de filas chamado MARS.

Observe os seguintes pontos que se aplicarão, se um aplicativo usar uma tabela de definição de canal do cliente:

- Quando o aplicativo criar um objeto `MQQueueManager` usando um construtor que usa um objeto de URL como um parâmetro, nenhum nome de canal deverá ser configurado na classe `MQEnvironment`, como um campo ou como uma propriedade de ambiente. Se um nome de canal for configurado, as classes IBM WebSphere MQ classes for Java lançam um `MQException`. A propriedade do campo ou ambiente especificando o nome do canal será considerada para ser configurada, se seu valor for algo diferente de nulo, uma sequência vazia ou uma sequência que contém todos os caracteres em branco.
- O parâmetro **queueManagerName** no construtor `MQQueueManager` pode ter um dos seguintes valores:
  - O nome de um gerenciador de filas
  - Um asterisco (*) seguido pelo nome de um grupo de gerenciadores de filas
  - Um asterisco (*)
  - Nulo, uma sequência vazia ou uma sequência que contém todos os caracteres em branco

  Estes são os mesmos valores que podem ser usados para o parâmetro **QMgrName** em uma chamada MQCONN emitida por um aplicativo cliente que está usando o Message Queue Interface (MQI). For more information about the meaning of these values, see “Visão geral da Message Queue Interface” na página 198.

  Se seu aplicativo usar conjunto de conexões, consulte “Controlando o conjunto de conexões padrão em classes do WebSphere MQ para Java” na página 702.
- Quando o cliente IBM WebSphere MQ classes for Java localiza uma definição de canal de conexão do cliente adequada na tabela de definição de canal do cliente, ele usa apenas as informações extraídas dessa definição de canal para iniciar um canal MQI. Quaisquer campos ou propriedades de ambiente relacionados ao canal que o aplicativo possa ser configurado na classe `MQEnvironment` são ignorados.

  Em particular, observe os seguintes pontos se você estiver usando Secure Sockets Layer (SSL):
  - Um canal MQI usa SSL somente se a definição de canal extraída da tabela de definições de canais do cliente especificar o nome de um CipherSpec suportado pelas classes IBM WebSphere MQ classes for Java .
  - Uma tabela de definição de canal do cliente também contém informações sobre o local dos servidores Lightweight Directory Access Protocol (LDAP) que mantém as listas de revogação de

certificado (CRLs). As classes IBM WebSphere MQ classes for Java usam apenas essas informações para acessar os servidores LDAP que contêm CRLs

– Uma tabela de definição de canal de cliente também pode conter o local de um respondente do Online Certificate Status Protocol (OCSP). As classes IBM WebSphere MQ classes for Java não podem usar as informações do OCSP em um arquivo de tabela de definições de canal do cliente.. No entanto, você pode configurar o OCSP conforme descrito na seção Usando Online Certificate Protocol.

Para obter mais informações sobre como usar o SSL com uma tabela de definição de canal do cliente, consulte Especificando que um canal de MQI usa o SSL.

Observe também os pontos a seguir se estiver usando saídas de canal:

– Um canal MQI usa as saídas do canal e dados do usuário associados especificado pela definição de canal extraída da tabela de definição de canal do cliente de preferência para saídas de canal e dados especificados usando outros métodos.

– Uma definição de canal extraída de uma tabela de definições de canal do cliente pode especificar saídas de canal gravadas em Java, C ou C + +. Para obter mais informações sobre como gravar uma saída de canal em Java, consulte “Criando uma saída de canal em classes WebSphere MQ para Java” na página 695. Para obter mais informações sobre como gravar uma saída de canal em outros idiomas, consulte “Usando saídas de canal não gravadas em Java com classes WebSphere MQ para Java” na página 699.

A especificação de um intervalo de portas para conexões do cliente IBM WebSphere MQ classes for Java

É possível especificar uma porta ou um intervalo de portas, que um aplicativo pode ser ligado em uma de duas maneiras.

Quando um aplicativo IBM WebSphere MQ classes for Java tenta se conectar a um gerenciador de filas do IBM WebSphere MQ no modo cliente, um firewall pode permitir apenas as conexões que se originam de portas ou intervalo de portas especificados. Nesta situação, é possível especificar uma porta ou um intervalo de portas ao qual o aplicativo pode ser ligado. É possível especificar a(s) porta(s) das seguintes maneiras:

- É possível configurar o campo localAddressSetting na classe MQEnvironment. Aqui está um exemplo:

```
MQEnvironment.localAddressSetting = "192.0.2.0(2000,3000)";
```

- É possível configurar o CMQC.LOCAL_ADDRESS_PROPERTY da propriedade do ambiente. Aqui está um exemplo:

```
(MQEnvironment.properties).put(CMQC.LOCAL_ADDRESS_PROPERTY,
                               "192.0.2.0(2000,3000)");
```

- Quando for possível construir o objeto MQQueueManager, será possível transmitir uma hashtable de propriedades que contém um LOCAL_ADDRESS_PROPERTY com o valor "192.0.2.0(2000,3000)"

Em cada um desses exemplos, quando o aplicativo posteriormente se conectar a um gerenciador de filas, o aplicativo será ligado a um endereço IP local e ao número da porta no intervalo de 192.0.2.0(2000) para 192.0.2.0(3000).

Em um sistema com mais de uma interface de rede, também é possível usar o campo localAddressSetting ou o CMQC.LOCAL_ADDRESS_PROPERTY da propriedade do ambiente para especificar qual interface de rede deve ser usada para uma conexão.

Os erros de conexão poderão ocorrer se você restringir o intervalo de portas. Se ocorrer um erro, uma MQException será lançada contendo o código de razão do IBM WebSphere MQ MQRC_Q_MGR_NOT_AVAILABLE e a seguinte mensagem:

```
Socket connection attempt refused due to LOCAL_ADDRESS_PROPERTY restrictions
```

Um erro pode ocorrer se todas as portas no intervalo especificado estiverem em uso ou se o endereço IP especificado, um nome do host ou número de porta não for válido (um número de porta negativo, por exemplo).

Acessando filas, tópicos e processos no WebSphere MQ classes para Java

Para acessar filas, tópicos e processos, use métodos da classe MQQueueManager. O MQOD (estrutura do descritor de objeto) é reduzido nos parâmetros destes métodos.

Filas

Para abrir uma fila, é possível usar o método accessQueue da classe MQQueueManager. Por exemplo, em um gerenciador de filas chamado queueManager, use o seguinte código:

```
MQQueue queue = queueManager.accessQueue("qName",CMQC.MQOO_OUTPUT);
```

O método accessQueue retorna um novo objeto da classe MQQueue.

Quando tiver concluído o uso da fila, use o método close() para fechá-la, como no exemplo a seguir:

```

queue.close();
```

Também é possível criar uma fila usando o construtor MQQueue. Os parâmetros são exatamente os mesmos que para o método accessQueue, com a adição de um parâmetro do gerenciador de filas. Por exemplo:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            CMQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserID");
```

É possível especificar várias opções ao criar filas. Para obter detalhes sobre isso, consulte Class.com.ibm.mq.MQQueue. Construir um objeto de fila desta maneira permite que você grave suas próprias subclasses de MQQueue.

tópicos

De forma semelhante, é possível abrir um tópico usando o método da classe accessTopic da classe MQQueueManager. Por exemplo, em um gerenciador de filas chamado queueManager, use o seguinte código para criar um assinante e publicador:

```
MQTopic subscriber =
      queueManager.accessTopic("TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_SUBSCRIPTION, CMQC.MQSO_CREATE);
```

```
MQTopic publisher =
      queueManager.accessTopic("TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_PUBLICATION, CMQC.MQOO_OUTPUT);
```

Quando tiver concluído o uso do tópico, use o método close() para fechá-lo.

Também é possível criar um tópico usando o construtor MQTopic. Os parâmetros são exatamente os mesmos que para o método accessTopic, com a adição de um parâmetro do gerenciador de filas. Por exemplo:

```
MQTopic subscriber = new
      MQTopic(queueManager,"TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_SUBSCRIPTION, CMQC.MQSO_CREATE);
```

É possível especificar várias opções ao criar tópicos. Para obter detalhes, consulte Classe com.ibm.mq.MQTopic . Construir um objeto de tópico desta maneira permite que você grave suas próprias subclasses de MQTopic.

Um tópico deve ser aberto para publicação ou para assinatura. A classe MQQueueManager tem oito métodos accessTopic e a classe Tópico possui oito construtores. Em cada caso, quatro desses têm um parâmetro **destination** e quatro têm um parâmetro **subscriptionName** (incluindo dois que possuem ambos). Eles podem ser usados somente para abrir o tópico para assinaturas. Os dois métodos restantes possuem um parâmetro **openAs**, e o tópico pode ser aberto para qualquer publicação ou assinatura dependendo do valor do parâmetro **openAs**.

Para criar um tópico como assinante durável, use um método accessTopic da classe MQQueueManager ou um construtor MQTopic que aceite um nome de assinatura e, em qualquer caso, defina a opção CMQC.MQSO_DURABLE.

Processos

Para acessar um processo, use o método accessProcess do MQQueueManager. Por exemplo, em um gerenciador de filas chamado queueManager, use o seguinte código para criar um objeto MQProcess:

```
MQProcess process =
      queueManager.accessProcess("PROCESSNAME",
      CMQC.MQOO_FAIL_IF_QUIESCING);
```

Para acessar um processo, use o método accessProcess do MQQueueManager.

O método accessProcess retorna um novo objeto da classe MQProcess.

Quando tiver concluído o uso do objeto do processo, use o método close() para fechá-lo, como no exemplo a seguir:

```
process.close();
```

Também é possível criar um processo usando o construtor MQProcess. Os parâmetros são exatamente os mesmos que para o método accessProcess, com a adição de um parâmetro do gerenciador de filas. Por exemplo:

```
 MQProcess process =
      new MQProcess(queueManager,"PROCESSNAME",
      CMQC.MQOO_FAIL_IF_QUIESCING);
```

Construir um objeto do processo desta maneira permite que você grave suas próprias subclasses de MQProcess.

Manipulando mensagens em classes WebSphere MQ para Java

As mensagens são representadas pela classe MQMessage. Você colocar e obter mensagens utilizando métodos da classe MQDestination, que possui subclasses de MQQueue e MQTopic.

Coloque mensagens nas filas ou tópicos usando o método put() da classe MQDestination. Você recebe mensagens das filas ou tópicos usando o método get() da classe MQDestination. Diferente da interface processual, em que MQPUT e MQGET colocam e obtêm matrizes de bytes, a linguagem de programação Java coloca e obtém instâncias da classe MQMessage. A classe MQMessage encapsula o buffer de dados que contém os dados da mensagem reais, junto com todos os parâmetros MQMD (descritor de mensagens) e propriedades de mensagem que descrevem essa mensagem.

Para criar uma nova mensagem, crie uma nova instância da classe MQMessage, e utilizar os métodos writeXXX para colocar dados no buffer de mensagem.

Quando a nova instância de mensagem for criada, todos os parâmetros MQMD serão configurados automaticamente para seus valores padrão, conforme definido em Valores iniciais e declarações de idioma para MQMD . O método put() de MQDestination também usa uma instância da classe

MQPutMessageOptions como parâmetro. Esta classe representa a estrutura MQPMO. O exemplo a seguir cria uma mensagem e a coloca em uma fila:

```
// Build a new message containing my age followed by my name
MQMessage myMessage = new MQMessage();
myMessage.writeInt(25);

String name = "Charlie Jordan";
myMessage.writeInt(name.length());
myMessage.writeBytes(name);

// Use the default put message options...
MQPutMessageOptions pmo = new MQPutMessageOptions();

// put the message!
queue.put(myMessage,pmo);
```

O método get() de MQDestination retorna uma nova instância de MQMessage, que representa a mensagem recém-obtida da fila. Ele também utiliza uma instância da classe MQGetMessageOptions como um parâmetro. Esta classe representa a estrutura de MQGMO.

Você não precisa especificar um tamanho de mensagem máximo, porque o método get() ajusta automaticamente o tamanho de seu buffer interno para encaixar a mensagem recebida. Use os métodos readXXX da classe MQMessage para acessar os dados na mensagem retornada.

O exemplo a seguir mostra como obter uma mensagem de uma fila:

```
// Get a message from the queue
MQMessage theMessage    = new MQMessage();
MQGetMessageOptions gmo = new MQGetMessageOptions();
queue.get(theMessage,gmo);  // has default values

// Extract the message data
int age = theMessage.readInt();
int strLen = theMessage.readInt();
byte[] strData = new byte[strLen];
theMessage.readFully(strData,0,strLen);
String name = new String(strData,0);
```

É possível alterar o formato numérico que os métodos de leitura e gravação usam configurando a variável de membro *encoding*.

É possível alterar o conjunto de caracteres para usar para ler e gravar sequências configurando a variável de membro *characterSet*.

Consulte “Classe MQMessage” na página 1082 para obter mais informações.

Nota: O método writeUTF() de MQMessage codifica automaticamente o comprimento da sequência, bem como os bytes Unicode que ela contém. Quando sua mensagem será lida por outro programa Java (usando readUTF()), esta é a maneira mais simples de enviar informações de sequência.

Melhorando o desempenho de mensagens não persistentes em classes WebSphere MQ para Java

Para melhorar o desempenho ao procurar as mensagens ou ao consumir as mensagens não persistentes de um aplicativo cliente, será possível usar a *leitura antecipada*. Os aplicativos clientes que usam MQGET ou consumo assíncrono se beneficiarão das melhorias de desempenho ao procurar mensagens ou ao consumir mensagens não persistentes.

Para obter informações gerais sobre o recurso de leitura antecipada, consulte o tópico relacionado.

Nas classes do WebSphere MQ para Java, você usa o CMQC.MQSO_READ_AHEAD e CMQC.MQSO_NO_READ_AHEAD de um objeto MQQueue ou MQTopic para determinar se os consumidores de mensagens e os navegadores de filas têm permissão para usar leitura antecipada nesse objeto.

Colocando mensagens de forma assíncrona usando classes do WebSphere MQ para Java

Para colocar uma mensagem forma assíncrona, configure MQPMO_ASYNC_RESPONSE.

Coloque as mensagens nas filas ou tópicos usando o método put() da classe MQDestination. Para colocar uma mensagem de forma assíncrona, ou seja, permitindo que a operação seja concluída sem aguardar por uma resposta do gerenciador de filas, é possível configurar MQPMO_ASYNC_RESPONSE no campo de opções de MQPutMessageOptions. Para determinar o sucesso ou falha de colocações assíncronas, use a chamada MQQueueManager.getAsyncStatus.

Publicar / assinar em classes WebSphere MQ para Java

Nas classes WebSphere MQ para Java, o tópico é representado pela classe MQTopic e você publica nele usando os métodos MQTopic.put().

Para obter informações gerais sobre a publicação / assinatura do WebSphere MQ , consulte Introdução ao sistema de mensagens de publicação / assinatura do WebSphere MQ

Manipulando cabeçalhos da mensagem do WebSphere MQ com classes do WebSphere MQ para Java

Classes Java são fornecidas representando diferentes tipos de cabeçalho da mensagem. Duas classes auxiliares também são fornecidas.

Objetos de cabeçalho são descritos pela interface MQHeader, que fornece métodos de propósitos gerais para acessar campos de cabeçalho e para ler e gravar conteúdo de mensagem. Cada tipo de cabeçalho tem sua própria classe que implementa a interface MQHeader e inclui métodos getter e setter para campos individuais. Por exemplo, o tipo de cabeçalho MQRFH2 é representado pela classe MQRFH2; o tipo de cabeçalho MQDLH pela classe MQDLH e assim por diante. As classes de cabeçalho executam qualquer conversão de dados necessária automaticamente e podem ler ou gravar dados em qualquer codificação numérica ou conjunto de caracteres (CCSID) especificado.

Duas classes auxiliares, MQHeaderIterator e MQHeaderList, ajudam na leitura e decodificação (análise) do conteúdo de cabeçalho em mensagens:

- A classe MQHeaderIterator funciona como um java.util.Iterator. Enquanto houver mais cabeçalhos na mensagem, o método next() retorna true e o método nextHeader() ou next() retorna o próximo objeto de cabeçalho.
- A MQHeaderList funciona como um java.util.List. Como o MQHeaderIterator, ele analisa o conteúdo do cabeçalho, mas também permite procurar cabeçalhos específicos, incluir novos cabeçalhos, remover cabeçalhos existentes, atualizar campos de cabeçalho e, em seguida, gravar o conteúdo do cabeçalho novamente em uma mensagem. Como alternativa, é possível criar uma MQHeaderList vazia e, em seguida, preenchê-la com instâncias de cabeçalho e gravá-la em uma mensagem uma vez ou repetidamente.

As classes MQHeaderIterator e MQHeaderList usam as informações no MQHeaderRegistry para saber quais classes de cabeçalho WebSphere MQ estão associadas a tipos e formatos de mensagens específicos. O MQHeaderRegistry é configurado com conhecimento de todos os formatos e tipos de cabeçalho WebSphere MQ atuais e suas classes de implementação e também é possível registrar seus próprios tipos de cabeçalho.

O suporte é fornecido para os cabeçalhos a seguir comumente usados do Websphere MQ

- MQRFH - Regras e formatação de cabeçalho
- MQRFH2 -Como MQRFH, usado para transmitir mensagens para e de um message broker pertencente ao WebSphere Message Broker. Também usado para conter as propriedades de mensagem
- MQCIH-Ponte CICS
- MQDLH - Cabeçalho de mensagens não entregues

- MQIIH-cabeçalho de informações do IMS
- MQRMH - cabeçalho de mensagem de referência
- MQSAPH - Cabeçalho SAP
- MQWIH - Cabeçalho de informações de trabalho
- MQXQH - Cabeçalho da fila de transmissão
- MQDH - Cabeçalho de distribuição
- MQEPH - Cabeçalho PCF contido

Também é possível definir classes que representam seus próprios cabeçalhos.

Para usar um MQHeaderIterator para obter um cabeçalho RFH2, configure MQGMO_PROPERTIES_FORCE_MQRFH2 em GetMessageOptions ou configure a propriedade da fila PROPCTL para FORCE.

Imprimindo todos os cabeçalhos em uma mensagem usando classes WebSphere MQ para Java

Neste exemplo, uma instância de MQHeaderIterator analisa os cabeçalhos em uma MQMessage que foi recebida de uma fila. O objetos MQHeader retornados do método nextHeader() exibem sua estrutura e conteúdo quando seu método toString é chamado.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeader;
import com.ibm.mq.headers.MQHeaderIterator;
...
MQMessage message = ... // Message received from a queue.
MQHeaderIterator it = new MQHeaderIterator (message);

while (it.hasNext ())
{
    MQHeader header = it.nextHeader ();

    System.out.println ("Header type " + header.type () + ": " + header);
}
```

Ignorando os cabeçalhos em uma mensagem usando classes do WebSphere MQ para Java

Neste exemplo, o método skipHeaders () de MQHeaderIterator posiciona o cursor de leitura da mensagem imediatamente após o último cabeçalho.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderIterator;
...
MQMessage message = ... // Message received from a queue.
MQHeaderIterator it = new MQHeaderIterator (message);

it.skipHeaders ();
```

Localizando o código de razão em uma mensagem de devoluções usando classes WebSphere MQ para Java

Neste exemplo, o método de leitura preenche o objeto MQDLH pela leitura da mensagem. Após a operação de leitura, o cursor de leitura da mensagem é posicionado imediatamente após o conteúdo do cabeçalho MQDLH.

As mensagens na fila de mensagens não entregues do gerenciador de filas são prefixadas com um cabeçalho de devoluções (MQDLH). Para decidir como manipular essas mensagens, por exemplo, para determinar se deseja tentar novamente ou descartá-las, um aplicativo de manipulação de devoluções deve ver o código de razão contido no MQDLH.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
...
MQMessage message = ... // Message received from the dead-letter queue.
```

```
MQDLH dlh = new MQDLH ();

dlh.read (message);

System.out.println ("Reason: " + dlh.getReason ());
```

Todas as classes de cabeçalho também fornecem um construtor de conveniência para inicializá-las diretamente a partir da mensagem em uma única etapa. Portanto, o código neste exemplo poderia ser simplificado da seguinte forma:

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
...
MQMessage message = ... // Message received from the dead-letter queue.
MQDLH dlh = new MQDLH (message);

System.out.println ("Reason: " + dlh.getReason ());
```

Lendo e removendo o MQDLH de uma mensagem de devoluções usando classes WebSphere MQ para Java

Neste exemplo, MQDLH é usado para remover o cabeçalho de uma mensagem não entregue.

Um aplicativo de manipulação de mensagens não entregues irá normalmente reenviar mensagens que foram rejeitadas se o código de razão indicar um erro transitório. Antes de reenviar a mensagem, ele deve remover o cabeçalho MQDLH.

Este exemplo executa as etapas a seguir (consulte os comentários no código de exemplo):

1. O MQHeaderList lê a mensagem inteira e cada cabeçalho encontrado na mensagem se torna um item na lista.
2. Mensagens não entregues contêm um MQDLH como seu primeiro cabeçalho, portanto, ele pode ser localizado no primeiro item da lista do cabeçalho. O MQDLH já foi preenchido a partir da mensagem quando o MQHeaderList foi construído, portanto, não há necessidade de chamar seu método de leitura.
3. O código de razão é extraído usando o método getReason() fornecido pela classe MQDLH.
4. O código de razão foi inspecionado e indica que é apropriado para reenviar a mensagem. O MQDLH é removido usando o método MQHeaderList remove().
5. O MQHeaderList grava seu conteúdo restante em um novo objeto de mensagem. A nova mensagem agora contém tudo o que estiver na mensagem original, exceto o MQDLH e pode ser gravada em uma fila. O argumento **true** para o construtor e para o método de gravação indica que o corpo da mensagem deve ser mantido no MQHeaderList e gravado novamente.
6. O campo de formato no descritor de mensagens da nova mensagem agora contém o valor que estava anteriormente no campo de formato do MQDLH. Os dados da mensagem correspondem à codificação numérica e ao CCSID configurado no descritor de mensagens.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
import com.ibm.mq.headers.MQHeaderList;
...
MQMessage message = ... // Message received from the dead-letter queue.
MQHeaderList list = new MQHeaderList (message, true); // Step 1.
MQDLH dlh = (MQDLH) list.get (0); // Step 2.
int reason = dlh.getReason (); // Step 3.
...
list.remove (dlh); // Step 4.

MQMessage newMessage = new MQMessage ();

list.write (newMessage, true); // Step 5.
newMessage.format = list.getFormat (); // Step 6.
```

Imprimindo o conteúdo de uma mensagem usando classes WebSphere MQ para Java

Este exemplo usa MQHeaderList para imprimir o conteúdo de uma mensagem, incluindo seus cabeçalhos.

A saída contém uma visualização de todos os conteúdos do cabeçalho, bem como do corpo da mensagem. A classe MQHeaderList decodifica todos os cabeçalhos de uma vez, enquanto que MQHeaderIterator passa por eles um por vez sob controle do aplicativo. Você pode usar essa técnica para fornecer uma ferramenta de depuração simples ao gravar aplicativos MQ.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderList;
...
MQMessage message = ... // Message received from a queue.

System.out.println (new MQHeaderList (message, true));
```

Este exemplo também imprime os campos do descritor de mensagens, usando a classe MQMD. O método copyFrom() da classe com.ibm.mq.headers.MQMD preenche o objeto de cabeçalho a partir dos campos do descritor de mensagens da MQMessage em vez de lendo o corpo da mensagem.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQMD;
import com.ibm.mq.headers.MQHeaderList;
...
MQMessage message = ...
MQMD md = new MQMD ();
...
md.copyFrom (message);
System.out.println (md + "\n" + new MQHeaderList (message, true));
```

Localizando um tipo específico de cabeçalho em uma mensagem usando classes WebSphere MQ para Java

Este exemplo usa o método indexOf(String) de MQHeaderList para localizar um cabeçalho MQRFH2 em uma mensagem, se uma estiver presente.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderList;
import com.ibm.mq.headers.MQRFH2;
...
MQMessage message = ...
MQHeaderList list = new MQHeaderList (message);
int index = list.indexOf ("MQRFH2");

if (index >= 0)
{
    MQRFH2 rfh = (MQRFH2) list.get (index);
    ...
}
```

Analisando um cabeçalho MQRFH2 usando classes WebSphere MQ para Java

Este exemplo mostra como acessar um valor de campo conhecido em uma pasta denominada, usando a classe MQRFH2.

A classe MQRFH2 fornece inúmeras maneiras para acessar não apenas os campos na parte fixa da estrutura, mas também o conteúdo da pasta codificada por XML que é transportado dentro do campo NameValueData. Este exemplo mostra como acessar um valor de campo conhecido em uma pasta nomeada-nesta instância, o campo Rto na pasta jms, que representa o nome da fila de respostas em uma mensagem JMS do MQ ..

```
MQRFH2 rfh = ...

String value = rfh.getStringFieldValue ("jms", "Rto");
```

Para descobrir o conteúdo de um MQRFH2 (em vez de solicitar campos específicos diretamente), é possível usar o método getFolders para retornar uma lista de MQRFH2.Element, que representa a estrutura de uma pasta que poderia conter campos e outras pastas. A definição de um campo ou pasta para nulo a remove do MQRFH2. Ao manipular o conteúdo da pasta NameValueData desta maneira, o campo StrucLength é automaticamente atualizado de acordo.

Lendo e gravando fluxos de bytes diferentes de objetos MQMessage usando classes WebSphere MQ para Java

Esses exemplos usam as classes de cabeçalho para analisar e manipular o conteúdo do cabeçalho do WebSphere MQ quando a fonte de dados não é um objeto MQMessage.

É possível usar as classes de cabeçalho para analisar e manipular o conteúdo do cabeçalho do WebSphere MQ , mesmo quando a origem de dados é algo diferente de um objeto MQMessage A interface MQHeader implementada por cada classe de cabeçalho fornece os métodos `int read (java.io.DataInput message, int encoding, int characterSet)` e `int write (java.io.DataOutput message, int encoding, int characterSet)`.. A classe com.ibm.mq.MQMessage implementa as interfaces java.io.DataInput e java.io.DataOutput. Isso significa que é possível usar os dois métodos MQHeader para ler e gravar conteúdo de MQMessage, substituindo a codificação e o CCSID especificados no descritor de mensagens. Isso é útil para mensagens que contêm uma cadeia de cabeçalhos em diferentes codificações.

Também é possível obter objetos DataInput e DataOutput de outros fluxos de dados, por exemplo, fluxos de arquivo ou de soquete, ou matrizes de bytes transportadas em mensagens JMS As classes java.io.DataInputStream implementam DataInput e as classes java.io.DataOutputStream implementam DataOutput. Este exemplo lê o conteúdo do cabeçalho WebSphere MQ a partir de uma matriz de bytes:

```
import java.io.*;
import com.ibm.mq.headers.*;
...
byte [] bytes = ...
DataInput in = new DataInputStream (new ByteArrayInputStream (bytes));
MQHeaderIterator it = new MQHeaderIterator (in, CMQC.MQENC_NATIVE,
  CMQC.MQCCSI_DEFAULT);
```

A linha iniciada por MQHeaderIterator poderia ser substituída por

```
MQDLH dlh = new MQDLH (in, CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
// or any other header type
```

Este exemplo grava em uma matriz de bytes usando um DataOutputStream:

```
MQHeader header = ... // Could be any header type
ByteArrayOutputStream out = new ByteArrayOutputStream ();

header.write (new DataOutputStream (out), CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
byte [] bytes = out.toByteArray ();
```

Ao trabalhar com fluxos dessa maneira, tome cuidado para usar os valores corretos para a codificação e os argumentos de characterSet. Ao ler cabeçalhos, especifique a codificação e o CCSID com o qual o conteúdo de bytes foi gravado originalmente. Ao gravar cabeçalhos, especifique a codificação e o CCSID que deseja produzir. A conversão de dados é executada automaticamente pelas classes de cabeçalho.

Criando classes para novos tipos de cabeçalho usando classes WebSphere MQ para Java

É possível criar classes Java para tipos de cabeçalho não fornecidos com classes WebSphere MQ para Java.

Para incluir uma classe Java que representa um novo tipo de cabeçalho que pode ser usado da mesma maneira que qualquer classe de cabeçalho fornecida com as classes WebSphere MQ para Java, crie uma classe que implementa a interface MQHeader. A abordagem mais simples é estender a classe com.ibm.mq.headers.impl.Header. Este exemplo produz uma classe totalmente funcional que representa a estrutura do cabeçalho MQTM. Não é necessário incluir métodos getter e setter individuais para cada campo, mas é uma conveniência útil para usuários da classe de cabeçalho. Os métodos getValue e setValue genéricos que usam uma sequência para o nome do campo funcionarão para todos os campos definidos no tipo de cabeçalho. Os métodos read, write e size herdados permitirão que instâncias do novo tipo de cabeçalho sejam lidas e gravadas e calcularão o tamanho do cabeçalho corretamente com base em sua definição de campo. A definição de tipo é criada apenas uma vez, no entanto, muitas instâncias dessa classe de cabeçalho são criadas. Para disponibilizar a nova definição de cabeçalho para decodificar o uso das classes MQHeaderIterator ou MQHeaderList, você deveria registrá-la usando o

MQHeaderRegistry. Observe que a classe de cabeçalho MQTM já é fornecida neste pacote e registrado no registro padrão.

```
import com.ibm.mq.headers.impl.Header;
import com.ibm.mq.headers.impl.HeaderField;
import com.ibm.mq.headers.CMQC;

public class MQTM extends Header {
    final static HeaderType TYPE = new HeaderType ("MQTM");
    final static HeaderField StrucId = TYPE.addMQChar ("StrucId", CMQC.MQTM_STRUC_ID);
    final static HeaderField Version = TYPE.addMQLong ("Version", CMQC.MQTM_VERSION_1);
    final static HeaderField QName = TYPE.addMQChar ("QName", CMQC.MQ_Q_NAME_LENGTH);
    final static HeaderField ProcessName = TYPE.addMQChar ("ProcessName",
      CMQC.MQ_PROCESS_NAME_LENGTH);
    final static HeaderField TriggerData = TYPE.addMQChar ("TriggerData",
      CMQC.MQ_TRIGGER_DATA_LENGTH);
    final static HeaderField ApplType = TYPE.addMQLong ("ApplType");
    final static HeaderField ApplId = TYPE.addMQChar ("ApplId", 256);
    final static HeaderField EnvData = TYPE.addMQChar ("EnvData", 128);
    final static HeaderField UserData = TYPE.addMQChar ("UserData", 128);

    protected MQTM (HeaderType type){
        super (type);
    }
    public String getStrucId () {
        return getStringValue (StrucId);
    }
    public int getVersion () {
        return getIntValue (Version);
    }
    public String getQName () {
        return getStringValue (QName);
    }
    public void setQName (String value) {
        setStringValue (QName, value);
    }
    // ...Add convenience getters and setters for remaining fields in the same way.
}
```

Manipulando mensagens PCF com classes WebSphere MQ para Java

As classes Java são fornecidas para criar e analisar mensagens estruturadas por PCF e para facilitar o envio de solicitações PCF e a coleta de respostas PCF.

Classes PCFMessage e MQCFGR representam matrizes de estruturas de parâmetros PCF. Elas fornecem métodos de conveniência para inclusão e recuperação de parâmetros PCF.

Estruturas de parâmetros PCF são representadas pelas classes MQCFH, MQCFIN, MQCFIN64, MQCFST, MQCFBS, MQCFIL, MQCFIL64 MQCFSL e MQCFGR. Elas compartilham interfaces operacionais básicas:

- Métodos para ler e gravar conteúdo da mensagem: read (), write () e size ()
- Métodos para manipular parâmetros: getValue (), setValue (), getParameter () e outros
- O método enumerador .nextParameter (), que analisa o conteúdo de PCF em um MQMessage

O parâmetro de filtro PCF é usado em comandos inquire para fornecer uma função de filtro. Ele está contido nas classes a seguir:

- MQCFIF - filtro de número inteiro
- MQCFSF - filtro de sequência
- MQCFBF - filtro de byte

Duas classes de agente, PCFAgent e PCFMessageAgent, são fornecidas para gerenciar a conexão com um Gerenciador de filas, a fila do servidor de comandos e uma fila de resposta associada. PCFMessageAgent estende PCFAgent e deve ser normalmente usado preferencialmente. A classe PCFMessageAgent converte os MQMessages recebidos e os transmite de volta ao responsável pela chamada como uma matriz PCFMessage. PCFAgent retorna uma matriz de MQMessages, que você precisa para analisar antes de usar.

Manipulando propriedades de mensagem em classes WebSphere MQ para Java

As chamadas de função para processar identificadores de mensagens não têm equivalente nas classes WebSphere MQ para Java. Para configurar, retornar ou excluir propriedades de identificadores de mensagens, use métodos da classe MQMessage.

Para obter informações gerais sobre propriedades de mensagem, consulte “Nomes de propriedades” na página 19.

No WebSphere MQ classes para acesso Java a mensagens é por meio da classe MQMessage. Portanto, os identificadores de mensagens não são fornecidos no ambiente Java e não há equivalente às chamadas de função do WebSphere MQ MQCRTMH, MQDLTMH, MQMHBUF e MQBUFMH

Para configurar as propriedades de identificadores de mensagens na interface processual, use a chamada MQSETMP. Nas classes WebSphere MQ para Java, use o método apropriado da classe MQMessage:

- setBooleanProperty
- setByteProperty
- setBytesProperty
- setShortProperty
- setIntProperty
- setInt2Property
- setInt4Property
- setInt8Property
- setLongProperty
- setFloatProperty
- setDoubleProperty
- setStringProperty
- setObjectProperty

Estes são, às vezes, referidos coletivamente como os métodos *set*property*.

Para retornar o valor de propriedades de identificador de mensagens na interface processual, use a chamada MQINQMP. Nas classes WebSphere MQ para Java, use o método apropriado da classe MQMessage:

- getBooleanProperty
- getByteProperty
- getBytesProperty
- getShortProperty
- getIntProperty
- getInt2Property
- getInt4Property
- getInt8Property
- getLongProperty
- getFloatProperty
- getDoubleProperty
- getStringProperty
- getObjectProperty

Estes são, às vezes, referidos coletivamente como os métodos *get*property*.

Para excluir o valor de propriedades de identificadores de mensagens na interface processual, use a chamada MQDLTMP. Nas classes do WebSphere MQ para Java, use o método deleteProperty da classe MQMessage

Manipulando erros no WebSphere MQ classes para Java

Manipular erros originados das classes WebSphere MQ para Java usando blocos Java `try` e `catch` .

Os métodos na interface Java não retornam um código de conclusão e um código de razão. Em vez disso, eles lançam uma exceção sempre que o código de conclusão e código de razão resultantes de um chamada do WebSphere MQ não são ambos zero. Isto simplifica a lógica do programa de forma que não seja necessário verificar os códigos de retorno após cada chamada para o WebSphere MQ. É possível decidir em quais pontos em seu programa você deseja lidar com a possibilidade de falha. Nestes pontos, é possível cercar seu código com blocos `try` e `catch`, como no exemplo a seguir:

```
try {
    myQueue.put(messageA,putMessageOptionsA);
    myQueue.put(messageB,putMessageOptionsB);
}
catch (MQException ex) {
    // This block of code is only executed if one of
    // the two put methods gave rise to a non-zero
    // completion code or reason code.
    System.out.println("An error occurred during the put operation:" +
                           "CC = " + ex.completionCode +
                           "RC = " + ex.reasonCode);
    System.out.println("Cause exception:" + ex.getCause() );
}
```

Os códigos de razão de chamadas WebSphere MQ relatados de volta em exceções Java para z/OS são documentados em Códigos de razão para z/OS e Códigos de razão para todas as outras plataformas.

As exceções que são lançadas enquanto um WebSphere MQ classes para aplicativo Java está em execução também são gravadas no log. No entanto, um aplicativo pode chamar o método MQException.logExclude() para evitar que as exceções associadas a um código de razão específico sejam registradas. Talvez você deseja fazer isso em situações em que espera que muitas exceções associadas a um código de razão específico sejam lançadas e não deseje que o log fique cheio com essas exceções. Por exemplo, se seu aplicativo tenta obter uma mensagem de uma fila toda vez que a iteração em um loop acontece e, para a maioria destas tentativas, você espera que nenhuma mensagem adequada esteja na fila, você pode desejar impedir que as exceções associadas ao código de razão MQRC_NO_MSG_AVAILABLE sejam registradas. Se um aplicativo evitou anteriormente que exceções associadas a um código de razão específico fossem registradas, ele pode permitir que essas exceções sejam registradas novamente, chamando o método MQException.logInclude().

Às vezes, o código de razão não transmite todos os detalhes associados ao erro. Para cada exceção que é lançada, um aplicativo deve verificar a exceção vinculada. A exceção vinculada sozinha pode ter outra exceção vinculada e assim as exceções vinculadas formam uma cadeia levando de volta ao problema subjacente original. Uma exceção vinculada é implementada usando o mecanismo de exceção em cadeia da classe java.lang.Throwable e um aplicativo obtém uma exceção vinculada chamando o método Throwable.getCause(). Em uma exceção que é uma instância de MQException, MQException.getCause() recupera a instância subjacente do com.ibm.mq.jmqi.JmqiException e getCause, a partir desta exceção, recupera a java.lang.Exception subjacente que causou o erro.

Por padrão, a classe MQException transmite automaticamente as exceções para System.err, que geralmente é direcionado para o console Se desejar parar exceções que aparecem no console, inclua uma linha em seu aplicativo para configurar MQException.log= null.

Obtendo e configurando valores de atributos em classes do WebSphere MQ para Java

Os métodos getXXX() e setXXX() são fornecidos para muitos atributos comuns. Outros podem ser acessados usando os métodos inquire() e set() genéricos.

Para muitos dos atributos comuns, as classes MQManagedObject, MQDestination, MQQueue, MQTopic, MQProcess e MQQueueManager contêm os métodos getXXX() e setXXX(). Esses métodos permitem que você obtenha e configure seus valores de atributos. Observe que para MQDestination, MQQueue e MQTopic, os métodos funcionam apenas se você especificar a consulta apropriada e configurar sinalizadores quando abrir o objeto.

Para atributos menos comuns, as classes MQQueueManager, MQDestination, MQQueue, MQTopic e MQProcess herdam de uma classe chamada MQManagedObject. Esta classe define as interfaces inquire() e set().

Quando você cria um novo objeto do gerenciador de filas usando o operador *new*, ele é aberto automaticamente para consulta. Quando você usa o método accessProcess() para acessar um objeto do processo, esse objeto é automaticamente aberto para consulta. Quando você usa o método accessQueue() para acessar um objeto da fila, esse objeto *não* é automaticamente aberto para as operações de consulta ou configuração. Isso pode ocorrer porque incluir essas opções automaticamente pode causar problemas com alguns tipos de filas remotas. Para usar os métodos inquire, set, getXXX e setXXX em uma fila, deve-se especificar os sinalizadores de consulta e configuração apropriados no parâmetro openOptions do método accessQueue(). O mesmo é verdadeiro para os objetos de destino e de tópico.

Os métodos inquire e set utilizam três parâmetros:

- matriz de seletores
- matriz intAttrs
- matriz charAttrs

Não são necessários os parâmetros SelectorCount, IntAttrCount e CharAttrLength localizados em MQINQ, porque o comprimento de uma matriz em Java é sempre conhecido. O exemplo a seguir mostra como fazer uma consulta em uma fila:

```
// inquire on a queue
final static int MQIA_DEF_PRIORITY = 6;
final static int MQCA_Q_DESC = 2013;
final static int MQ_Q_DESC_LENGTH = 64;

int[] selectors  = new int[2];
int[] intAttrs   = new int[1];
byte[] charAttrs = new byte[MQ_Q_DESC_LENGTH]

selectors[0] = MQIA_DEF_PRIORITY;
selectors[1] = MQCA_Q_DESC;

queue.inquire(selectors,intAttrs,charAttrs);

System.out.println("Default Priority = " + intAttrs[0]);
System.out.println("Description : " + new String(charAttrs,0));
```

Programas multiencadeados em Java

O Java Runtime Environment é inerentemente multiencadeado. WebSphere MQ classes para Java permite que um objeto do gerenciador de fila seja compartilhado por vários encadeamentos, mas assegura que todo o acesso ao gerenciador de fila de destino seja sincronizado.

Programas multiencadeados são difíceis de evitar em Java. Considere um programa simples que se conecta a um gerenciador de filas e abre uma fila na inicialização. O programa exibe um único botão na tela. Quando um usuário clica nesse botão, o programa busca uma mensagem da fila.

O Java Runtime Environment é inerentemente multiencadeado. Portanto, a inicialização do seu aplicativo ocorre em um encadeamento e o código que é executado em resposta ao pressionamento do botão é executado em um encadeamento separado (o encadeamento da interface com o usuário).

Com o cliente MQI do WebSphere MQ baseado em C, isso causaria um problema, pois há limitações para o compartilhamento de manipulações por vários encadeamentos WebSphere MQ classes para Java relaxa essa restrição, permitindo que um objeto do gerenciador de filas (e sua fila associada, tópico e objetos do processo) seja compartilhado por vários encadeamentos..

A implementação de classes WebSphere MQ para Java assegura que, para uma determinada conexão (instância do objetoMQQueueManager), todo o acesso ao gerenciador de filas WebSphere MQ de destino seja sincronizado. Um encadeamento que deseja emitir uma chamada a um gerenciador de filas é bloqueado até que todas as outras chamadas em andamento para essa conexão sejam concluídas. Se você requerer acesso simultâneo ao mesmo gerenciador de filas a partir de múltiplos encadeamentos em seu programa, crie um novo objeto MQQueueManager para cada encadeamento que requer acesso simultâneo. (Isto é equivalente a emitir uma chamada MQCONN separada para cada encadeamento.)

Nota: As instâncias da classe `com.ibm.mq.MQGetMessageOptions` não devem ser compartilhadas entre encadeamentos que estão solicitando mensagens simultaneamente. Instâncias dessa classe são atualizadas com dados durante a solicitação MQGET correspondente, que pode resultar em consequências inesperadas quando vários encadeamentos são operacionais simultaneamente na mesma instância do objeto.

Usando saídas de canal nas classes do WebSphere MQ para Java

Uma visão geral de como usar saídas de canal em um aplicativo usando as classes WebSphere MQ para Java.

Os tópicos a seguir descrevem como gravar uma saída de canal em Java, como designá-la e como transmitir dados para ela Eles, em seguida, descrevem como usar saídas de canal gravadas em C e como usar uma sequência de saídas do canal.

Seu aplicativo deve ter a permissão de segurança correta para carregar a classe de saída do canal.

Criando uma saída de canal em classes WebSphere MQ para Java

É possível fornecer suas próprias saídas de canal definindo uma classe Java que implementa uma interface apropriada.

Para implementar uma saída, defina uma nova classe Java que implementa a interface apropriada. Três interfaces de saída são definidas no pacote com.ibm.mq.exits:

- WMQSendExit
- WMQReceiveExit
- WMQSecurityExit

Nota: Saídas do canal são suportadas apenas para conexões do cliente; elas não são suportadas para conexões de ligações. Não é possível usar uma saída de canal Java fora das classes WebSphere MQ para Java, por exemplo, se você estiver usando um aplicativo cliente gravado em C.

Qualquer criptografia SSL definida para uma conexão é executada *após* as saídas de envio e de segurança terem sido chamadas. Da mesma forma, a decriptografia é executada *antes* das saídas de recebimento e de segurança serem chamadas.

A amostra a seguir define uma classe que implementa todas as três interfaces:

```
public class MyMQExits implements
WMQSendExit, WMQReceiveExit, WMQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method comes from the send exit interface
  public ByteBuffer channelSendExit(
MQCXP channelExitParms,
                                    MQCD channelDefinition,
                                    ByteBuffer agentBuffer)
  {
    // Fill in the body of the send exit here
  }
    // This method comes from the receive exit interface
  public ByteBuffer channelReceiveExit(
MQCXP channelExitParms,
                                       MQCD channelDefinition,
                                       ByteBuffer agentBuffer)
  {
    // Fill in the body of the receive exit here
  }
    // This method comes from the security exit interface
```

```
  public ByteBuffer channelSecurityExit(
MQCXP channelExitParms,
                                      MQCD channelDefinition,
                                      ByteBuffer agentBuffer)
  {
    // Fill in the body of the security exit here
  }
}
```

Cada saída transmitiu um objeto MQCXP e um objeto MQCD. Estes objetos representam as estruturas MQCXP e MQCD definidas na interface processual.

Toda classe de saída gravada deve ter um construtor. Este pode ser o construtor padrão ou um que obtenha um argumento de sequência. Se ele obtiver uma sequência, os dados do usuário serão transmitidos para a classe de saída quando ela for criada. Se a classe de saída contiver um construtor padrão e um construtor de argumento único, o construtor de argumento único terá prioridade.

Para as saídas de envio e de segurança, seu código de saída deve retornar os dados que deseja enviar para o servidor. Para uma saída de recebimento, seu código de saída deve retornar os dados modificados que você deseja que WebSphere MQ interprete.

O corpo de saída mais simples possível é:

```
{ return agentBuffer; }
```

Não feche o gerenciador de filas de dentro de uma saída do canal.

Usando classes de saída do canal existentes

Nas versões do WebSphere MQ anteriores ao 7.0, você implementaria essas saídas usando as interfaces MQSendExit, MQReceiveExit, e MQSecurityExit, como no exemplo a seguir: Este método permanece válido, mas o novo método é preferido para funcionalidade e desempenho aprimorados.

```
public class MyMQExits implements MQSendExit, MQReceiveExit, MQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method comes from the send exit
  public byte[] sendExit(MQChannelExit channelExitParms,
                         MQChannelDefinition channelDefParms,
                         byte agentBuffer[])
  {
    // Fill in the body of the send exit here
  }
    // This method comes from the receive exit
  public byte[] receiveExit(MQChannelExit channelExitParms,
                            MQChannelDefinition channelDefParms,
                            byte agentBuffer[])
  {
    // Fill in the body of the receive exit here
  }
    // This method comes from the security exit
  public byte[] securityExit(MQChannelExit channelExitParms,
                             MQChannelDefinition channelDefParms,
                             byte agentBuffer[])
  {
    // Fill in the body of the security exit here
  }
}
```

Designando uma saída do canal no IBM WebSphere MQ classes for Java

É possível designar uma saída do canal usando o IBM WebSphere MQ classes for Java.

Não há nenhum equivalente direto para o canal IBM WebSphere MQ no IBM WebSphere MQ classes for Java. As saídas de canal são designadas a um MQQueueManager. Por exemplo, tendo definido uma classe que implementa a interface WMQSecurityExit, um aplicativo pode usar a saída de segurança em uma de quatro maneiras:

- Ao designar uma instância da classe para o campo MQEnvironment.channelSecurityExit antes de criar um objeto MQQueueManager
- Configurar o campo MQEnvironment.channelSecurityExit como uma sequência que representa a classe de saída de segurança antes de criar um objeto MQQueueManager
- Ao criar um par de chave/valor na hashtable de propriedades transmitida para MQQueueManager com uma chave de CMQC.SECURITY_EXIT_PROPERTY
- Usando uma Tabela de Definição de Canal do Cliente (CCDT)

Qualquer saída designada configurando o campo MQEnvironment.channelSecurityExit para uma sequência, criando um par de chave/valor na hashtable de propriedades ou usando um CCDT, deve ser gravada com um construtor padrão. Uma saída designada como uma instância de uma classe não precisa de um construtor padrão, dependendo do aplicativo.

Um aplicativo pode usar uma saída de envio ou de recebimento de forma semelhante. Por exemplo, o fragmento de código a seguir mostra como usar a segurança, saídas de envio e recebimento que são implementados na classe MyMQExits, que foi definida anteriormente, usando MQEnvironment:

```
MyMQExits myexits = new MyMQExits();
MQEnvironment.channelSecurityExit = myexits;
MQEnvironment.channelSendExit = myexits;
MQEnvironment.channelReceiveExit = myexits;
:
MQQueueManager jupiter = new MQQueueManager("JUPITER");
```

Se mais de um método for usado para designar uma saída do canal, a ordem de precedência será a seguinte:

1. Se a URL de uma CCDT for transmitida para o MQQueueManager, o conteúdo da CCDT determina as saídas do canal a serem usadas e quaisquer definições de saída em MQEnvironment ou hashtable de propriedades são ignoradas.
2. Se nenhuma URL CCDT for transmitida, as definições de saída de MQEnvironment e hashtable serão mescladas
   - Se o mesmo tipo de saída for definido em ambos MQEnvironment e a hashtable, a definição na hashtable será usada.
   - Se tipos de saída antigos e novos equivalentes forem especificados (por exemplo, o campo sendExit , que pode ser usado somente para o tipo de saída usado em versões de IBM WebSphere MQ anteriores à Versão 7.0e o campo de saída channelSend, que pode ser usado para qualquer saída de envio), a nova saída (channelSendExit) será usada em vez da saída antiga.

Se você tiver declarado uma saída do canal como uma sequência, deverá ativar o IBM WebSphere MQ para localizar o programa de saída do canal. É possível fazer isso de várias maneiras, dependendo do ambiente no qual o aplicativo está em execução e sobre como os programas de saída de canal são compactados.

- Para um aplicativo que está em execução em um servidor de aplicativos, deve-se armazenar os arquivos no diretório mostrado em Tabela 88 na página 698 ou compactados em arquivos JAR referenciados por **exitClasspath**.
- Para um aplicativo que não está em execução em um servidor de aplicativos, as regras a seguir se aplicam:
  - Se as classes de saída do canal forem compactadas em arquivos JAR separados, esses arquivos JAR deverão ser incluídos no **exitClasspath**.
  - Se as classes de saída do canal não forem compactadas em arquivos JAR, os arquivos de classe podem ser armazenados no diretório mostrado em Tabela 88 na página 698 ou em qualquer diretório no caminho de classe do sistema da JVM ou **exitClasspath**.

A propriedades **exitClasspath** pode ser especificada de quatro maneiras. Em ordem de prioridade, essas maneiras são as seguintes:

1. A propriedade do sistema com.ibm.mq.exitClasspath (definida na linha de comandos usando a opção -D)

2. A sub-rotina exitPath do arquivo `mqclient.ini`
3. Uma entrada de hashtable com a chave CMQC.EXIT_CLASSPATH_PROPERTY
4. A variável MQEnvironment **exitClasspath**

Separe vários caminhos usando o caractere java.io.File.pathSeparator.

Tabela 88. O diretório para programas de saída de canal

Plataforma	**Diretório**
AIX, HP-UX, Linuxe Solaris	`/var/mqm/exits` (programas de saída de canal de 32 bits) `/var/mqm/exits64` (programas de saída de canal de 64 bits)
Windows	*install_data_dir* \exits
Nota: *install_data_dir* é o diretório que você escolheu para os arquivos de dados do IBM WebSphere MQ durante a instalação. O diretório padrão é `C:\Program Files\IBM\WebSphere MQ`.	

Passando dados para saídas do canal no WebSphere MQ classes para Java

É possível passar saídas de canal e retornar dados de saídas de canal para seu aplicativo.

O parâmetro agentBuffer

Para uma saída de envio, o parâmetro *agentBuffer* contém os dados que estão prestes a serem enviados. Para uma saída de recebimento ou uma saída de segurança, o parâmetro *agentBuffer* contém os dados que acabam de ser recebidos. Você não precisa de um parâmetro de comprimento, porque a expressão agentBuffer.limit() indica o comprimento da matriz.

Para as saídas de envio e de segurança, seu código de saída deve retornar os dados que deseja enviar para o servidor. Para uma saída de recebimento, seu código de saída deve retornar os dados modificados que você deseja que WebSphere MQ interprete.

O corpo de saída mais simples possível é:

```
{ return agentBuffer; }
```

Saídas de canal são chamadas com um buffer que tem uma matriz auxiliar. Para melhor desempenho, a saída deve retornar um buffer com uma matriz auxiliar.

Dados do usuário

Se um aplicativo se conecta a um gerenciador de filas, configurando channelSecurityExit, channelSendExit ou channelReceiveExit, 32 bytes de dados do usuário podem ser passados para a classe de saída do canal apropriada quando chamada, usando os campos channelSecurityExitUserData, channelSendExitUserData ou channelReceiveExitUserData. Esses dados do usuário estão disponíveis para a classe de saída do canal, mas são atualizados sempre que a saída for chamada. Quaisquer mudanças feitas nos dados do usuário na saída do canal serão, portanto, perdidas. Se deseja fazer mudanças persistentes nos dados em uma saída de canal, use exitUserArea de MQCXP. Os dados nesse campo são mantidos entre as chamadas da saída.

Se o aplicativo configurar securityExit, sendExit ou receiveExit, nenhum dado do usuário poderá ser passado para essas classes de saída do canal.

Se um aplicativo usar uma tabela de definição de canal de cliente (CCDT) para se conectar a um gerenciador de filas, quaisquer dados do usuário especificados em uma definição de canal de conexão de cliente serão passados para classes de saída do canal quando forem chamadas. Para obter mais informações sobre como usar uma tabela de definição de canal do cliente, consulte “Usando uma tabela de definição de canal do cliente com o IBM WebSphere MQ classes for Java” na página 680.

Usando saídas de canal não gravadas em Java com classes WebSphere MQ para Java

Como usar programas de saída de canal gravados em C de um aplicativo Java.

No WebSphere MQ Versão 7.0, é possível especificar o nome de um programa de saída de canal gravado em C como uma Sequência transmitida para os campos de Saída channelSecurity, Saída channelSendou Saída channelReceiveno objeto MQEnvironment ou propriedades Hashtable. No entanto, não é possível usar uma saída de canal gravada em Java em um aplicativo gravado em outra linguagem..

Especifique o nome do programa de saída no formato `library(function)` e assegure que o local do programa de saída seja incluído na variável de ambiente de caminho.

Para obter informações sobre como escrever uma saída do canal em C, consulte "Programas de Saída de Canal para Canais de Mensagens" na página 402.

Usando as classes de saída externas

Nas versões do WebSphere MQ anteriores à Versão 7.0, três classes foram fornecidas para permitir que você use saídas de canal gravadas em idiomas diferentes de Java:

- MQExternalSecurityExit, que implementa a interface MQSecurityExit
- MQExternalSendExit, que implementa a interface MQSendExit
- MQExternalReceiveExit, que implementa a interface MQReceiveExit

O uso dessas classes permanece válido, mas o novo método é preferencial.

Para usar uma saída de segurança que não é gravada em Java, um aplicativo primeiro teve que criar um objeto de Saída MQExternalSecurity O aplicativo especificou, como parâmetros no construtor MQExternalSecurityExit, o nome da biblioteca que contém a saída de segurança, o nome do ponto de entrada para a saída de segurança e os dados do usuário a serem passados para a saída de segurança quando foi chamado. Programas de saída do canal que não são gravados em Java foram armazenados no diretório mostrado em Tabela 88 na página 698.

Usando uma sequência de saídas de envio ou recebimento de canal nas classes do WebSphere MQ para Java

Um aplicativo WebSphere MQ classes para Java pode usar uma seqüência de saídas de envio ou recebimento de canal que são executadas em sucessão.

Para usar uma sequência de saídas de envio, um aplicativo pode criar um List ou String contendo as saídas de envio. Se um List for usado, cada elemento de List poderá ser qualquer um dos seguintes:

- Uma instância de uma classe definida pelo usuário que implementa a interface WMQSendExit
- Uma instância de uma classe definida pelo usuário que implementa a interface MQSendExit (para uma saída de envio gravada em Java)
- Uma instância da classe de saída MQExternalSend(para uma saída de envio não gravada em Java)
- Uma instância da classe MQSendExitChain
- Uma instância da classe String

Um List não pode conter outro List.

O aplicativo pode usar uma sequência de saídas de recebimento de maneira semelhante.

Se uma Sequência for usada, ela deverá consistir em uma ou mais definições de saída separadas por vírgula, cada uma das quais pode ser o nome de uma classe Java ou um programa C no formato `library(function)`.

O aplicativo então designará o objeto List ou String ao campo MQEnvironment.channelSendExit antes de criar um objeto MQQueueManager.

O contexto de informações transmitidas às saídas está unicamente no domínio das saídas. Por exemplo, se uma saída Java e uma saída C forem encadeadas, a presença da saída Java não terá efeito na saída C.

Usando as classes de cadeia de saída

Em versões do WebSphere MQ anteriores à Versão 7.0, duas classes foram fornecidas para permitir sequências de saídas:

- MQSendExitChain, que implementa a interface MQSendExit
- MQReceiveExitChain, que implementa a interface MQReceiveExit

O uso dessas classes permanece válido, mas o novo método é preferencial. O uso das Classes do WebSphere MQ para interfaces Java significa que seu aplicativo ainda tem uma dependência de `com.ibm.mq.jar` Se o novo conjunto de interfaces no pacote `com.ibm.mq.exits` for usado, não haverá dependência de `com.ibm.mq.jar`

Para usar uma sequência de saídas de envio, um aplicativo criou uma lista de objetos, em que cada objeto era um dos seguintes:

- Uma instância de uma classe definida pelo usuário que implementa a interface MQSendExit (para uma saída de envio gravada em Java)
- Uma instância da classe de saída MQExternalSend(para uma saída de envio não gravada em Java)
- Uma instância da classe MQSendExitChain

O aplicativo criou um objeto MQSendExitChain transmitindo esta lista de objetos como um parâmetro no construtor. O aplicativo teria então designado o objeto MQSendExitChain para o campo MQEnvironment.sendExit antes de criar um objeto MQQueueManager.

Compactação de canal no WebSphere MQ classes para Java

A compactação de dados que fluem em um canal pode melhorar o desempenho do canal e reduzir o tráfego de rede.IBM WebSphere MQ classes for Java Use a função de compactação construída em IBM WebSphere MQ.

Usando a função fornecida com o IBM WebSphere MQ, é possível compactar os dados que fluem em canais de mensagens e canais MQI e, em qualquer tipo de canal, é possível compactar dados de cabeçalho e dados de mensagens independentemente um do outro. Por padrão, nenhum dados é compactado em um canal. Para obter uma descrição completa da compactação de canal, incluindo como ela é implementada no IBM WebSphere MQ, consulte Compactação de dados (COMPMSG) e Compactação de cabeçalho (COMPHDR).

Um aplicativo IBM WebSphere MQ classes for Java especifica as técnicas que podem ser usadas para compactar dados de cabeçalho ou mensagem em uma conexão do cliente criando um objeto java.util.Collection . Cada técnica de compactação é um objeto Integer na coleta e a ordem na qual o aplicativo inclui as técnicas de compactação para a coleta é a ordem na qual as técnicas de compactação serão negociadas com o gerenciador de filas quando a conexão do cliente for iniciada. O aplicativo pode então designar a coleta para o campo hdrCompList, para os dados do cabeçalho ou o msgCompList, para dados da mensagem, na classe MQEnvironment. Quando o aplicativo estiver pronto, ele poderá iniciar a conexão do cliente criando um objeto MQQueueManager.

Os seguintes fragmentos de código ilustram a abordagem descrita. O primeiro fragmento de código mostra como implementar a compactação de dados de cabeçalho:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(CMQXC.MQCOMPRESS_SYSTEM));
:
MQEnvironment.hdrCompList = headerComp;
:
MQQueueManager qMgr = new MQQueueManager(QM);
```

O segundo fragmento de código mostra como implementar a compactação dos dados da mensagem:

```
Collection msgComp = new Vector();
msgComp.add(new Integer(CMQXC.MQCOMPRESS_RLE));
msgComp.add(new Integer(CMQXC.MQCOMPRESS_ZLIBHIGH));
:
```

```
MQEnvironment.msgCompList = msgComp;
:
MQQueueManager qMgr = new MQQueueManager(QM);
```

No segundo exemplo, as técnicas de compactação serão negociadas na ordem RLE, em seguida, ZLIBHIGH, quando a conexão do cliente for iniciada. A técnica de compactação selecionada não pode ser mudada durante o tempo de vida do objeto MQQueueManager.

As técnicas de compactação para os dados de cabeçalho e de mensagens que são suportadas pelo gerenciador de filas e o cliente em uma conexão do cliente são transmitidas para uma saída de canal como coletas nos campos hdrCompList e msgCompList de um objeto MQChannelDefinition. As técnicas reais que estão sendo atualmente usadas para compactar os dados de cabeçalho e mensagens em uma conexão do cliente são transmitidas para uma saída de canal nos campos CurHdrCompression e CurMsgCompression de um objeto MQChannelExit.

Se a compactação for usada em uma conexão do cliente, os dados serão compactados antes de quaisquer saídas de envio do canal serem processadas e extraídas após quaisquer saídas de recebimento de canal serem processadas. Os dados transmitidos para enviar e receber saídas está, portanto, em um estado compactado.

Para obter mais informações sobre como especificar técnicas de compactação e sobre quais técnicas de compactação estão disponíveis, consulte Classe com.ibm.mq.MQEnvironment e Interface com.ibm.mq.MQC .

Compartilhando uma conexão TCP/IP em IBM WebSphere MQ classes for Java

Várias instâncias de um canal MQI podem ser feitas para compartilhar uma única conexão TCP/IP.

No IBM WebSphere MQ classes for Java, usa-se a variável MQEnvironment.sharingConversations para controlar o número de conversas que podem compartilhar uma única conexão TCP/IP.

O atributo SHARECNV é uma abordagem de melhor esforço para o compartilhamento de conexão. Portanto, quando um valor SHARECNV maior que 0 é usado com o IBM WebSphere MQ classes for Java, não é garantido que um novo pedido de conexão irá sempre compartilhar uma conexão já estabelecida.

Conjunto de conexões em classes WebSphere MQ para Java

As classes WebSphere MQ para Java permitem que conexões sobressalentes sejam agrupadas para reutilização.

As classes do WebSphere MQ para Java fornecem suporte adicional para aplicativos que lidam com várias conexões com gerenciadores de fila do WebSphere MQ . Quando uma conexão não for mais necessária, em vez de destruí-la, ela poderá ser agrupada e reutilizada posteriormente. Isso pode fornecer um aprimoramento de desempenho substancial para aplicativos e middleware conectados de forma serial aos gerenciadores de filas arbitrários.

O WebSphere MQ fornece um conjunto de conexões padrão Os aplicativos podem ativar ou desativar esse conjunto de conexões registrando e removendo o registro de tokens através da classe MQEnvironment. Se o conjunto estiver ativo quando as classes WebSphere MQ para Java construírem um objeto MQQueueManager , ele procurará esse conjunto padrão e reutilizará qualquer conexão adequada. Quando uma chamada MQQueueManager.disconnect() ocorrer, a conexão subjacente será retornada ao conjunto.

Como alternativa, os aplicativos podem construir um conjunto de conexões MQSimpleConnectionManager para um uso específico. Em seguida, o aplicativo pode especificar esse conjunto durante a construção de um objeto MQQueueManager ou transmitir esse conjunto para MQEnvironment para usar como o conjunto de conexões padrão.

Para evitar que conexões usem muitos recursos, é possível limitar o número total de conexões que um objeto MQSimpleConnectionManager pode manipular e o tamanho do conjunto de conexões. A configuração de limites será útil se houver demandas conflitantes para conexões dentro de uma JVM.

Por padrão, o método getMaxConnections() retorna o valor de zero, o que significa que não há limite para o número de conexões que o objeto MQSimpleConnectionManager pode manipular. É possível configurar

um limite usando o método setMaxConnections(). Se você configurar um limite e o limite for atingido, uma solicitação para uma conexão adicional poderá fazer com que uma MQException seja lançada, com um código de razão de MQRC_MAX_CONNS_LIMIT_REACHED.

Controlando o conjunto de conexões padrão em classes do WebSphere MQ para Java

Este exemplo mostra como usar o conjunto de conexões padrão.

Considere o aplicativo de exemplo a seguir, MQApp1:

```
import com.ibm.mq.*;
public class MQApp1
{
      public static void main(String[] args) throws MQException
      {
         for (int i=0; i<args.length; i++) {
            MQQueueManager qmgr=new MQQueueManager(args[i]);
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
         }
      }
}
```

MQApp1 usa uma lista de gerenciadores de filas locais a partir da linha de comandos, se conecta a cada sucessivamente e executa alguma operação. No entanto, quando a linha de comandos listar o mesmo gerenciador de filas várias vezes, será mais eficiente para se conectar apenas uma vez e reutilizar essa conexão várias vezes.

WebSphere MQ classes para Java fornece um conjunto de conexões padrão que você pode usar para fazer isso. Para ativar o conjunto, use um dos métodos MQEnvironment.addConnectionPoolToken(). Para desativar o conjunto, use MQEnvironment.removeConnectionPoolToken().

O aplicativo de exemplo a seguir, MQApp2, é funcionalmente idêntico ao MQApp1, mas se conecta apenas uma vez para cada gerenciador de filas.

```
import com.ibm.mq.*;
public class MQApp2
{
      public static void main(String[] args) throws MQException
      {
         MQPoolToken token=MQEnvironment.addConnectionPoolToken();

         for (int i=0; i<args.length; i++) {
            MQQueueManager qmgr=new MQQueueManager(args[i]);
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
         }

         MQEnvironment.removeConnectionPoolToken(token);

      }
}
```

A primeira linha em negrito ativa o conjunto de conexões padrão registrando um objeto MQPoolToken com MQEnvironment.

O construtor MQQueueManager agora procura esse conjunto para uma conexão apropriada e apenas criará uma conexão com o gerenciador de filas se ele não puder localizar uma existente. A chamada qmgr.disconnect() retorna a conexão ao conjunto para reutilização posterior. Essas chamadas de API são as mesmas que o aplicativo de amostra MQApp1.

A segunda linha destacada desativa o conjunto de conexões padrão, que destrói quaisquer conexões do gerenciador de filas armazenadas no conjunto. Isso é importante porque, caso contrário, o aplicativo finalizaria com um número de conexões do gerenciador de filas ativas no conjunto. Esta situação poderia causar erros que apareceriam nos logs do gerenciador de filas.

Se um aplicativo usar uma tabela de definição de canal do cliente (CCDT) para se conectar a um gerenciador de filas, o construtor MQQueueManager procurará primeiramente a tabela para uma definição de canal de conexão do cliente adequada. Se um for localizado, o construtor irá procurar o conjunto de conexões padrão para uma conexão que pode ser usada para o canal. Se o construtor não puder localizar uma conexão adequada no conjunto, ele irá procurar então na tabela de definições de canal do cliente pela próxima definição adequada do canal de conexão do cliente e continuará conforme descrito anteriormente. Se o construtor concluir sua procura da tabela de definição de canal do cliente e não localizar nenhuma conexão adequada no conjunto, o construtor iniciará uma segunda procura da tabela. Durante essa procura, o construtor tentará criar uma nova conexão para cada definição adequada de canal de conexão do cliente, por sua vez, e usará a primeira conexão que ele gerencia para criar.

O conjunto de conexões padrão armazena um máximo de dez conexões não usadas e mantém essas conexões ativas por um máximo de cinco minutos. O aplicativo pode alterar isto (para obter detalhes, consulte “Fornecendo um conjunto de conexões diferente no WebSphere MQ classes para Java” na página 704).

Em vez de usar MQEnvironment para fornecer um MQPoolToken, o aplicativo pode construir seu próprio:

```
MQPoolToken token=new MQPoolToken();
MQEnvironment.addConnectionPoolToken(token);
```

Alguns aplicativos ou fornecedores de middleware fornecem subclasses de MQPoolToken para transmitir informações a um conjunto de conexões customizado. Eles podem ser construídos e transmitidos para addConnectionPoolToken() desta maneira para que as informações extras possam ser transmitidas para o conjunto de conexões.

O conjunto de conexões padrão e diversos componentes nas classes WebSphere MQ para Java

Este exemplo mostra como incluir ou remover MQPoolTokens de um conjunto estático de objetos MQPoolToken registrados.

MQEnvironment contém um conjunto estático de objetos MQPoolToken registrados. Para incluir ou remover MQPoolTokens desse conjunto, use os métodos a seguir:

- MQEnvironment.addConnectionPoolToken()
- MQEnvironment.removeConnectionPoolToken()

Um aplicativo pode consistir em muitos componentes que existem independentemente e executam trabalho usando um gerenciador de filas. Nesse aplicativo, cada componente deve incluir um MQPoolToken no conjunto de MQEnvironment durante seu ciclo de vida.

Por exemplo, o aplicativo de exemplo MQApp3 cria dez encadeamentos e inicia cada um. Cada encadeamento registra seu próprio MQPoolToken, aguarda um período de tempo, em seguida, conecta-se ao gerenciador de filas. Depois que o encadeamento é desconectado, ele remove seu próprio MQPoolToken.

O conjunto de conexões padrão permanece ativo enquanto houver pelo menos um token no conjunto de MQPoolTokens, portanto, permanecerá ativo na duração desse aplicativo. O aplicativo não precisa manter um objeto principal em controle geral dos encadeamentos.

```
import com.ibm.mq.*;
public class MQApp3
{
      public static void main(String[] args)
      {
         for (int i=0; i<10; i++) {
            MQApp3_Thread thread=new MQApp3_Thread(i*60000);
            thread.start();
         }
      }
}

class MQApp3_Thread extends Thread
{
```

```
      long time;

      public MQApp3_Thread(long time)
      {
         this.time=time;
      }

      public synchronized void run()
      {
         MQPoolToken token=MQEnvironment.addConnectionPoolToken();
         try {
            wait(time);
            MQQueueManager qmgr=new MQQueueManager("my.qmgr.1");
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
         }
         catch (MQException mqe) {System.err.println("Error occurred!");}
         catch (InterruptedException ie) {}

         MQEnvironment.removeConnectionPoolToken(token);
      }
}
```

Fornecendo um conjunto de conexões diferente no WebSphere MQ classes para Java

Este exemplo mostra como usar a classe **com.ibm.mq.MQSimpleConnectionManager** para fornecer um conjunto de conexões diferente.

Essa classe fornece recursos básicos para definição do conjunto de conexões e os aplicativos podem usar essa classe para customizar o comportamento do conjunto.

Após ser instanciada, um MQSimpleConnectionManager pode ser especificado no construtor MQQueueManager. O MQSimpleConnectionManager, em seguida, gerencia a conexão subjacente ao MQQueueManager construído. Se o MQSimpleConnectionManager contiver uma conexão agrupada adequada, essa conexão será reutilizada e retornada ao MQSimpleConnectionManager após uma chamada MQQueueManager.disconnect().

O fragmento de código a seguir demonstra esse comportamento:

```
MQSimpleConnectionManager myConnMan=new MQSimpleConnectionManager();
myConnMan.setActive(MQSimpleConnectionManager.MODE_ACTIVE);
MQQueueManager qmgr=new MQQueueManager("my.qmgr.1", myConnMan);
:
: (do something with qmgr)
:
qmgr.disconnect();

MQQueueManager qmgr2=new MQQueueManager("my.qmgr.1", myConnMan);
:
: (do something with qmgr2)
:
qmgr2.disconnect();
myConnMan.setActive(MQSimpleConnectionManager.MODE_INACTIVE);
```

A conexão que é forjada durante o primeiro construtor MQQueueManager é armazenada em myConnMan após a chamada qmgr.disconnect(). A conexão é, então, reutilizada durante a segunda chamada ao construtor MQQueueManager.

A segunda linha ativa o MQSimpleConnectionManager. A última linha desativa MQSimpleConnectionManager, destruindo quaisquer conexões mantidas no conjunto. Um MQSimpleConnectionManager está, por padrão, em MODE_AUTO, que é descrito posteriormente nesta seção.

Um MQSimpleConnectionManager aloca conexões com base nas usadas mais recentemente e destrói conexões com base nas usadas menos recentemente. Por padrão, uma conexão será destruída se não tiver sido usada por cinco minutos ou se houver mais de dez conexões não usadas no conjunto. É possível alterar esses valores chamando MQSimpleConnectionManager.setTimeout().

Também é possível configurar um MQSimpleConnectionManager para uso como o conjunto de conexões padrão, para ser usado quando nenhum Connection Manager for fornecido no construtor MQQueueManager.

O aplicativo a seguir demonstra isso:

```
import com.ibm.mq.*;
public class MQApp4
{
      public static void main(String []args)
      {
         MQSimpleConnectionManager myConnMan=new MQSimpleConnectionManager();
         myConnMan.setActive(MQSimpleConnectionManager.MODE_AUTO);
         myConnMan.setTimeout(3600000);
         myConnMan.setMaxConnections(75);
         myConnMan.setMaxUnusedConnections(50);
         MQEnvironment.setDefaultConnectionManager(myConnMan);
         MQApp3.main(args);
      }
}
```

As linhas em negrito criam e configuram um objeto MQSimpleConnectionManager. A configuração faz o seguinte:

- Finaliza conexões que não são usadas para uma hora
- Limita o número de conexões gerenciadas por myConnMan para 75
- Limita o número de conexões não utilizadas no conjunto para 50
- Configura MODE_AUTO, que é o padrão. Isso significa que o conjunto está ativo somente se ele for o gerenciador de conexões padrão e houver pelo menos um token no conjunto de MQPoolTokens mantido por MQEnvironment.

O novo MQSimpleConnectionManager é, então, configurado como o gerenciador de conexões padrão.

Na última linha, o aplicativo chama MQApp3.main(). Isso executa vários encadeamentos, em que cada encadeamento usa WebSphere MQ de forma independente. Esses encadeamentos usam myConnMan quando forjam conexões.

Fornecendo seu próprio ConnectionManager para WebSphere MQ classes para Java

WebSphere MQ classes para Java fornece uma implementação parcial do Java EE Connector Architecture, permitindo que implementações de javax.resource.spi.ConnectionManager sejam usadas.

Aplicativos e provedores de middleware podem fornecer implementações alternativas de conjuntos de conexões. WebSphere MQ classes para Java fornece uma implementação parcial do Java EE Connector Architecture. Implementações de **javax.resource.spi.ConnectionManager** podem ser usadas como o Connection Manager padrão ou ser especificadas no construtor MQQueueManager

As classes do WebSphere MQ para Java estão em conformidade com o contrato de Gerenciamento de Conexão do Java EE Connector Architecture. Leia esta seção em conjunto com o contrato do Connection Management do Java EE Connector Architecture (consulte o Web site Java da Sun em https://java.sun.com).

A interface ConnectionManager define apenas um método:

```
package javax.resource.spi;
public interface ConnectionManager {
      Object allocateConnection(ManagedConnectionFactory mcf,
                                ConnectionRequestInfo cxRequestInfo);
}
```

O construtor MQQueueManager chama allocateConnection no ConnectionManagerapropriado.. Ele transmite implementações apropriadas de ManagedConnectionFactory e ConnectionRequestInfo como parâmetros para descrever a conexão necessária.

O ConnectionManager procura em seu conjunto um objeto javax.resource.spi.ManagedConnection que foi criado com objetos de Informações ManagedConnectionFactory e ConnectionRequestidênticos.

Se o ConnectionManager localizar quaisquer objetos ManagedConnection adequados, ele criará um java.util.Set que contém o candidato ManagedConnections Em seguida, o ConnectionManager chama o seguinte:

```
ManagedConnection mc=mcf.matchManagedConnections(connectionSet, subject,
cxRequestInfo);
```

A implementação do WebSphere MQ do ManagedConnectionFactory ignora o parâmetro subject. Esse método seleciona e retorna um ManagedConnection adequado do conjunto ou retorna nulo se ele não localizar um ManagedConnectionadequado. Se não houver um ManagedConnection adequado no conjunto, o ConnectionManager poderá criar um usando:

```
ManagedConnection mc=mcf.createManagedConnection(subject, cxRequestInfo);
```

Novamente, o parâmetro subject é ignorado. Este método se conecta a um gerenciador de filas do WebSphere MQ e retorna uma implementação de javax.resource.spi.ManagedConnection que representa a conexão recém-falsificada. Quando o ConnectionManager tiver obtido um ManagedConnection (a partir do conjunto ou recém-criado), ele criará uma manipulação de conexões usando:

```
Object handle=mc.getConnection(subject, cxRequestInfo);
```

Essa manipulação de conexões pode ser retornada de allocateConnection().

Um ConnectionManager deve registrar um interesse no ManagedConnection por meio de:

```
mc.addConnectionEventListener()
```

O Listener ConnectionEventé notificado se ocorrer um erro grave na conexão, ou quando MQQueueManager.disconnect () for chamado. Quando MQQueueManager.disconnect () é chamado, o Listener ConnectionEventpode fazer um dos seguintes:

- Reconfigure o ManagedConnection usando a chamada mc.cleanup(), em seguida, retorne o ManagedConnection para o conjunto
- Destruir o ManagedConnection usando a chamada mc.destroy()

Se o ConnectionManager for o padrão ConnectionManager, ele também poderá registrar um interesse no estado do conjunto gerenciado pelo MQEnvironment de MQPoolTokens Para fazer isso, primeiro construa um objeto MQPoolServices e, em seguida, registre um objeto MQPoolServicesEventListener com o objeto MQPoolServices :

```
MQPoolServices mqps=new MQPoolServices();
mqps.addMQPoolServicesEventListener(listener);
```

O listener é notificado quando um MQPoolToken é incluído ou removido do conjunto ou quando o padrão ConnectionManager muda. O objeto MQPoolServices também fornece uma maneira de consultar o tamanho atual de um conjunto de MQPoolTokens

Coordenação JTA/JDBC usando classes WebSphere MQ para Java

As classes do WebSphere MQ para Java suportam o método MQQueueManager.begin (), que permite que o WebSphere MQ aja como um coordenador para um banco de dados que fornece um driver compatível com JDBC tipo 2 ou JDBC tipo 4.

Esse suporte não está disponível em todas as plataformas. Para verificar quais plataformas suportam coordenação de JDBC, consulte https://www.ibm.com/software/integration/wmq/requirements/.

Para usar o suporte XA-JTA, deve-se usar a biblioteca de comutação de JTA especial. O método para usar essa biblioteca varia dependendo se você estiver usando o Windows ou uma das outras plataformas..

Configurando a coordenação JTA/JDBC no Windows

A biblioteca XA é fornecida como uma DLL com um nome do formato `jdbcxxx.dll`.

V 7.5.0.7 O `jdbcora12.dll` fornecido oferece compatibilidade com o Oracle 12C para uma instalação do servidor IBM WebSphere MQ Windows.

Em sistemas Windows, a biblioteca XA é fornecida como uma DLL completa. O nome desta DLL é `jdbcxxx.dll`, em que `xxx` indica o banco de dados para o qual a biblioteca de comutação foi compilada. Essa biblioteca está no diretório `java\lib\jdbc` ou `java\lib64\jdbc` de suas classes IBM WebSphere MQ para a instalação do Java Deve-se declarar a biblioteca XA, também descrita como o arquivo de carregamento do comutador, para o gerenciador de filas. Use o IBM WebSphere MQ Explorer. Especifique os detalhes do arquivo de carregamento do comutador no painel de propriedades do gerenciador de filas, sob o gerenciador de recursos XA. Deve-se fornecer somente o nome da biblioteca. Por exemplo:

Para um banco de dados Db2, configure o campo SwitchFile como: dbcdb2

Para um banco de dados Oracle, configure o campo SwitchFile como: `jdbcora`

Configurando a coordenação JTA/JDBC em plataformas diferentes do Windows

Arquivos de objeto são fornecidos. Vincule o apropriado usando o makefile fornecido e declare-o para o gerenciador de filas usando o arquivo de configuração.

Para cada sistema de gerenciamento de banco de dados, o WebSphere MQ fornece dois arquivos de objeto Deve-se vincular um arquivo de objeto para criar uma biblioteca de comutação de 32 bits e vincular o outro arquivo de objeto para criar uma biblioteca de comutação de 64 bits. Para DB2, o nome de cada arquivo de objeto é jdbcdb2.o e, para Oracle, o nome de cada arquivo de objeto é jdbcora.o.

Deve-se vincular cada arquivo de objeto usando o makefile apropriado fornecido com o WebSphere MQ Uma biblioteca de comutação requer outras bibliotecas, que podem ser armazenadas em locais diferentes em sistemas diferentes. No entanto, uma biblioteca de comutação não pode usar a variável de ambiente do caminho da biblioteca para localizar essas bibliotecas porque a biblioteca de comutação é carregada pelo gerenciador de filas, que é executado em um ambiente setuid. O makefile fornecido, portanto, assegura que uma biblioteca de comutação contém os nomes de caminhos completos dessas bibliotecas.

Para criar uma biblioteca de comutação, insira um comando **make** com o formato a seguir. Para criar uma biblioteca de comutação de 32 bits, insira o comando no diretório /java/lib/jdbc de sua instalação do WebSphere MQ Para criar uma biblioteca de comutação de 64 bits, insira o comando no diretório /java/lib64/jdbc.

```
make DBMS
```

em que *DBMS* é o sistema de gerenciamento de banco de dados para o qual você está criando a biblioteca de comutação. Os valores válidos são db2 para DB2 e `oracle` para Oracle.

Aqui está um exemplo de um comando **make**:

```
make db2
```

Note os seguintes pontos:

- Para executar aplicativos de 32 bits, deve-se criar uma biblioteca de comutação de 32 bits e uma de 64 bits para cada sistema de gerenciamento de banco de dados que você está usando. Para executar aplicativos de 64 bits, é necessário criar somente uma biblioteca de comutação de 64 bits. Para o DB2, o nome de cada biblioteca do comutador é jdbcdb2 e, para o Oracle, o nome de cada biblioteca do comutador é jdbcora. Os makefiles asseguram que bibliotecas de comutador de 32 bits e 64 bits sejam armazenadas em diferentes diretórios do WebSphere MQ . Uma biblioteca de comutação de 32 bits é armazenada no diretório /java/lib/jdbc e uma biblioteca de comutação de 64 bits é armazenada no diretório /java/lib64/jdbc.
- Como é possível instalar o Oracle em qualquer lugar de um sistema, os makefiles usam a variável de ambiente ORACLE_HOME para localizar onde o Oracle está instalado.

Depois de ter criado as bibliotecas do comutador para DB2, Oracle, ou ambos, você deve declará-las para seu gerenciador de filas Se o arquivo de configuração do gerenciador de filas (qm.ini) já contiver sub-rotinas XAResourceManager para bancos de dados DB2 ou Oracle , será necessário substituir a entrada SwitchFile em cada sub-rotina por um dos seguintes:

Para um banco de dados DB2

```
SwitchFile=jdbcdb2
```

Para um banco de dados Oracle

```
SwitchFile=jdbcora
```

Não especifique o nome do caminho completo da biblioteca de comutação de 32 bits ou de 64 bits. Especifique somente o nome da biblioteca.

Se o arquivo de configuração do gerenciador de filas ainda não contiver sub-rotinas XAResourceManager para DB2 ou Oracle bancos de dados ou se desejar incluir sub-rotinas XAResourceManager adicionais, consulte Administrando para obter informações sobre como construir uma sub-rotina XAResourceManager . No entanto, cada entrada SwitchFile em uma nova sub-rotina XAResourceManager deve ser exatamente conforme descrito anteriormente para um banco de dados DB2 ou Oracle .. Deve-se incluir também a entrada `ThreadOfControl=PROCESS`.

Após atualizar o arquivo de configuração do gerenciador de filas e certificar-se de que todas as variáveis de ambiente de banco de dados apropriadas foram configuradas, será possível reiniciar o gerenciador de filas.

Usando a coordenação de JTA/JDBC

Codifique suas chamadas API como no exemplo fornecido.

A sequência básica de chamadas API para um aplicativo de usuário é:

```
qMgr = new MQQueueManager("QM1")
Connection con = qMgr.getJDBCConnection( xads );
qMgr.begin()

< Perform MQ and DB operations to be grouped in a unit of work >

qMgr.commit() or qMgr.backout();
con.close()
qMgr.disconnect()
```

`xads` na chamada getJDBCConnection é uma implementação específica do banco de dados da interface XADataSource, que define os detalhes do banco de dados ao qual se conectar. Consulte a documentação de seu banco de dados para determinar como criar um objeto XADataSource apropriado para passar no getJDBCConnection.

Deve-se também atualizar o caminho de classe com os arquivos jar específicos do banco de dados apropriados executando o trabalho do JDBC.

Se você precisar se conectar a diversos bancos de dados, deverá chamar getJDBCConnection várias vezes para executar a transação em várias conexões diferentes.

Há duas formas de getJDBCConnection, refletindo as duas formas de XADataSource.getXAConnection:

```
public java.sql.Connection getJDBCConnection(javax.sql.XADataSource xads)
  throws MQException, SQLException, Exception

public java.sql.Connection getJDBCConnection(XADataSource dataSource,
                                             String userid, String password)
  throws MQException, SQLException, Exception
```

Esses métodos declaram Exceção em suas cláusulas throws para evitar problemas com o verificador da JVM para clientes que não estão usando as funções JTA. A exceção real lançada é

javax.transaction.xa.XAException que requer que o arquivo jta.jar seja incluído no caminho de classe para os programas que não o requeriam antes.

Para usar o suporte JTA/JDBC, deve-se incluir a instrução a seguir em seu aplicativo:

```
MQEnvironment.properties.put(CMQC.THREAD_AFFINITY_PROPERTY, new Boolean(true));
```

Problemas e limitações conhecidos com a coordenação de JTA/JDBC

Há certos problemas e limitações de suporte a JTA/JDBC, alguns dependendo do sistema de gerenciamento de banco de dados em uso.

Como este suporte faz chamadas a drivers JDBC, a implementação desses drivers JDBC pode ter um efeito significativo no comportamento do sistema. Especificamente, drivers JDBC testados se comportam de forma diferente quando o banco de dados é encerrado enquanto um aplicativo está em execução. **Sempre** evite encerrar abruptamente um banco de dados enquanto houver aplicativos retendo conexões abertas com ele.

Diversas sub-rotinas XAResourceManager

O uso de mais de uma sub-rotina XAResourceManager em um arquivo de configuração do gerenciador de filas, qm.ini, não é suportado. Qualquer sub-rotina XAResourceManager diferente da primeira será ignorada.

DB2

Às vezes, DB2 retorna um erro SQL0805N . Esse problema pode ser resolvido com o comando CLP a seguir:

```
DB2 bind @db2cli.lst blocking all grant public
```

Consulte a documentação do DB2 para obter mais informações.

A sub-rotina XAResourceManager deve ser configurada para usar ThreadOfControl=PROCESS. Para DB2 versão 8.1 e superior, isso não corresponde ao encadeamento padrão da configuração de controle para DB2, portanto toc=p deve ser especificado na Sequência Aberta XA. Uma sub-rotina XAResourceManager de exemplo para DB2 com coordenação JTA/JDBC é a seguinte:

```
XAResourceManager:
     Name=jdbcdb2
     SwitchFile=jdbcdb2
     XAOpenString=uid=userid,db=dbalias,pwd=password,toc=p
     ThreadOfControl=PROCESS
```

Isso não impede que os aplicativos Java que usam a coordenação JTA/JDBC sejam multiencadeados.

Oracle

Chamar o método JDBC Connection.close() após MQQueueManager.disconnect() gera uma SQLException. Chame Connection.close() antes de MQQueueManager.disconnect() ou omita a chamada a Connection.close().

Suporte SSL (Secure Sockets Layer) nas classes WebSphere MQ para Java

As classes do WebSphere MQ para aplicativos clientes Java suportam criptografia Secure Sockets Layer (SSL). Você requer um provedor JSSE para usar a criptografia SSL.

As classes do WebSphere MQ para aplicativos clientes Java que usam criptografia TRANSPORT (CLIENT) suportam Secure Sockets Layer (SSL). SSL fornece criptografia de comunicação, autenticação e integridade da mensagem. É geralmente usada para comunicações seguras entre qualquer dois pontos na Internet ou em uma intranet.

WebSphere MQ classes para Java usa Java Secure Socket Extension (JSSE) para manipular a criptografia SSL e, portanto, requer um provedor JSSE. JVMs JSE v1.4 têm um provedor JSSE integrado. Detalhes sobre como gerenciar e armazenar certificados podem variar de provedor para provedor. Para obter informações sobre isso, consulte a documentação do seu provedor JSSE.

Esta seção supõe que seu provedor JSSE esteja instalado e configurado corretamente e que os certificados adequados tenham sido instalados e disponibilizados para seu provedor JSSE.

Se suas classes WebSphere MQ para aplicativo cliente Java usarem uma tabela de definição de canal de cliente (CCDT) para se conectar a um gerenciador de filas, consulte “Usando uma tabela de definição de canal do cliente com o IBM WebSphere MQ classes for Java” na página 680.

Ativando SSL no IBM WebSphere MQ classes for Java

Para ativar o SSL, especifique um CipherSuite. Existem duas maneiras de especificar um CipherSuite.

SSL é suportado somente para conexões do cliente. Para ativar o SSL, deve-se especificar o CipherSuite a ser usado ao se comunicar com o gerenciador de filas e esse CipherSuite deve corresponder ao CipherSpec configurado no canal de destino. Além disso, o CipherSuite nomeado deve ser suportado pelo provedor JSSE. No entanto, CipherSuites são diferentes dos CipherSpecs e, portanto, têm nomes diferentes. O “SSL CipherSpecs e CipherSuites no WebSphere MQ classes para Java” na página 714 contém uma tabela mapeando os CipherSpecs suportados pelo IBM WebSphere MQ para seus CipherSuites equivalentes, conforme conhecidos pelo JSSE.

Para ativar o SSL, especifique o CipherSuite usando a variável de membro estático sslCipherSuite de MQEnvironment. O exemplo a seguir conecta-se a um canal SVRCONN denominado SECURE.SVRCONN.CHANNEL, que foi configurado para requerer o SSL com um CipherSpec de RC4_MD5_EXPORT:

```
MQEnvironment.hostname       = "your_hostname";
MQEnvironment.channel        = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.sslCipherSuite = "SSL_RSA_EXPORT_WITH_RC4_40_MD5";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

Embora o canal tenha um CipherSpec de RC4_MD5_EXPORT, o aplicativo Java deve especificar um CipherSuite de SSL_RSA_EXPORT_WITH_RC4_40_MD5. Consulte “SSL CipherSpecs e CipherSuites no WebSphere MQ classes para Java” na página 714 para obter uma lista de mapeamentos entre CipherSpecs e CipherSuites.

Um aplicativo também pode especificar um CipherSuite configurando a propriedade do ambiente CMQC.SSL_CIPHER_SUITE_PROPERTY.

Como alternativa, use o Client Channel Definition Table (CCDT). Para obter mais informações, consulte “Usando uma tabela de definição de canal do cliente com o IBM WebSphere MQ classes for Java” na página 680

Se você requerer que uma conexão do cliente use um CipherSuite que seja suportado pelo IBM Java provedor JSSE FIPS (IBMJSSEFIPS), um aplicativo poderá configurar o campo sslFipsNecessário na classe MQEnvironment como `true`. Como alternativa, o aplicativo pode configurar a propriedade do ambiente CMQC.SSL_FIPS_REQUIRED_PROPERTY. O valor padrão é `false`, que significa que uma conexão do cliente pode usar qualquer CipherSuite que seja suportado pelo IBM WebSphere MQ.

Se um aplicativo usar mais de uma conexão do cliente, o valor do campo sslFipsRequired que é usado quando o aplicativo cria a primeira conexão do cliente determinará o valor que é usado quando o aplicativo cria qualquer conexão do cliente subsequente. Portanto, quando o aplicativo cria uma conexão de cliente subsequente, o valor do campo sslFipsRequired é ignorado. Deve-se reiniciar o aplicativo se desejar usar um valor diferente para o campo sslFipsRequired.

Para se conectar com êxito usando SSL, o truststore JSSE deve ser configurado com certificados raiz da autoridade de certificação a partir da qual o certificado apresentado pelo gerenciador de filas pode ser autenticado. Da mesma forma, se SSLClientAuth no canal SVRCONN tiver sido configurado como MQSSL_CLIENT_AUTH_REQUIRED, o keystore JSSE deverá conter um certificado de identificação que seja confiável pelo gerenciador de filas.

Referências relacionadas

Federal Information Processing Standards (FIPS) para UNIX, Linux e Windows

Usando o nome distinto do gerenciador de filas no IBM WebSphere MQ classes for Java

O gerenciador de filas se identifica usando um certificado SSL, que contém um Nome Distinto (DN). Um aplicativo cliente do IBM WebSphere MQ classes for Java pode usar esse DN para assegurar que ele esteja se comunicando com o gerenciador de filas correto

Um padrão de DN é especificado usando a variável sslPeerName de MQEnvironment. Por exemplo, configurar:

```
MQEnvironment.sslPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

permite que a conexão seja bem-sucedida somente se o gerenciador de filas apresentar um certificado com um Nome Comum iniciando QMGR., e pelo menos dois nomes de Unidade Organizacional, o primeiro dos quais deve ser IBM e o segundo WebSphere

Se sslPeerName estiver configurado, as conexões serão bem-sucedidas somente se ele estiver configurado com um padrão válido e o gerenciador de filas apresentar um certificado correspondente.

Um aplicativo também pode especificar o nome distinto do gerenciador de filas, configurando a propriedade de ambiente CMQC.SSL_PEER_NAME_PROPERTY. Para obter mais informações sobre nomes distintos, consulte Nomes distintos.

Usando listas de revogação de certificado no IBM WebSphere MQ classes for Java

Especifique as listas de revogação de certificado a serem usadas pela classe java.security.cert.CertStore.IBM WebSphere MQ classes for Java em seguida, verifica os certificados com relação à CRL especificada

Uma lista de revogação de certificados (CRL) é um conjunto de certificados que foram revogados, pela autoridade de certificação emissora ou pela organização local. CRLs geralmente são hospedadas em servidores LDAP. Com o Java 2 v1.4, um servidor de CRL pode ser especificado no momento de conexão e o certificado apresentado pelo gerenciador de filas é verificado com relação à CRL antes que a conexão seja permitida. Para obter mais informações sobre listas de revogação de certificado e IBM WebSphere MQ, consulte Trabalhando com listas de revogação de certificado e listas de revogação de autoridade e Acessando CRLs e ARLs com classes WebSphere MQ para Java e WebSphere MQ classes para JMS.

Nota: Para usar um CertStore com sucesso com uma CRL hospedada em um servidor LDAP, certifique-se de que o Java Software Development Kit (SDK) seja compatível com a CRL. Alguns SDKs requerem que a CRL esteja em conformidade com o RFC 2587, que define um esquema para o LDAP v2. A maioria dos servidores LDAP v3 usa RFC 2256 em vez disso.

As CRLs a serem usadas são especificadas por meio da classe java.security.cert.CertStore. Consulte a documentação sobre essa classe para obter detalhes completos sobre como obter as instâncias de CertStore. Para criar um CertStore com base em um servidor LDAP, primeiro, crie uma instância LDAPCertStoreParameters inicializada com as configurações do servidor e da porta a serem usadas. Por exemplo:

```
import java.security.cert.*;
CertStoreParameters csp = new LDAPCertStoreParameters("crl_server", 389);
```

Tendo criado uma instância CertStoreParameters, use o construtor estático em CertStore para criar um CertStore do tipo LDAP:

```
CertStore cs = CertStore.getInstance("LDAP", csp);
```

Outros tipos de CertStore (por exemplo, Collection) também são suportados. Normalmente, há vários servidores CRL configurados com informações idênticas da CRL para fornecer redundância. Quando houver um objeto CertStore para cada um desses servidores CRL, coloque-os todos em um Collection apropriado. O exemplo a seguir mostra os objetos CertStore colocados em uma ArrayList:

```
import java.util.ArrayList;
Collection crls = new ArrayList();
crls.add(cs);
```

Esse Collection pode ser configurado para a variável estática MQEnvironment, sslCertStores, antes de se conectar para ativar a verificação de CRL:

```
MQEnvironment.sslCertStores = crls;
```

O certificado apresentado pelo gerenciador de filas quando uma conexão que está sendo configurada for validada da seguinte forma:

1. O primeiro objeto CertStore no Collection identificado por sslCertStores é usado para identificar um servidor de CRL.
2. Uma tentativa é feita para contatar o servidor de CRL.
3. Se a tentativa for bem-sucedida, uma correspondência do certificado é procurada no servidor.
   a. Se o certificado tiver sido revogado, o processo de procura terminou e a solicitação de conexão falhará com o código de razão MQRC_SSL_CERTIFICATE_REVOKED.
   b. Se o certificado não for localizado, o processo de procura termina e a conexão tem permissão para continuar.
4. Se a tentativa de contatar o servidor não for bem-sucedida, o próximo objeto CertStore será usado para identificar um servidor CRL e o processo é repetido a partir da etapa 2.

   Se esse era o último CertStore no Collection ou se o Collection não contiver nenhum objeto CertStore, o processo de procura falhou e a solicitação de conexão falhará com o código de razão MQRC_SSL_CERT_STORE_ERROR.

O objeto Collection determina a ordem na qual CertStores são usados.

Collection of CertStores também pode ser configurado usando CMQC.SSL_CERT_STORE_PROPERTY. Como conveniência, essa propriedade também permite que um único CertStore seja especificado sem ser um membro de um Collection.

Se sslCertStores for configurado como null, nenhuma verificação de CRL será executada. Essa propriedade será ignorada se sslCipherSuite não estiver configurado.

Renegociando a chave secreta em classes WebSphere MQ para Java

Um aplicativo cliente WebSphere MQ classes para Java pode controlar quando a chave secreta usada para criptografia em uma conexão do cliente é renegociada, em termos do número total de bytes enviados e recebidos.

O aplicativo pode fazer isso de uma das maneiras a seguir: se o aplicativo usar mais de uma dessas maneiras, as regras de precedência usuais se aplicam.

- Configurando o campo sslResetCount na classe MQEnvironment.
- Configurando uma propriedade de ambiente MQC.SSL_RESET_COUNT_PROPERTY em um objeto Hashtable. O aplicativo designa, então, a hashtable para o campo `properties` na classe MQEnvironment ou passa a hashtable para um objeto MQQueueManager em seu construtor.

O valor do campo de contagem sslResetou da propriedade de ambiente MQC.SSL_RESET_COUNT_PROPERTY representa o número total de bytes enviados e recebidos pelas classes WebSphere MQ para o código do cliente Java antes que a chave secreta seja renegociada. O número de bytes enviados é o número antes da criptografia e o número de bytes recebidos é o número após a decriptografia. O número de bytes também inclui informações de controle enviadas e recebidas pelas classes do WebSphere MQ para cliente Java.

Se a contagem de reconfiguração for zero, que é o valor padrão, a chave secreta nunca será renegociada. A contagem de reconfiguração será ignorada se nenhum CipherSuite for especificado.

Fornecendo um SSLSocketFactory customizado em IBM WebSphere MQ classes for Java

Se você usar um JSSE Socket Factory customizado, configure o MQEnvironment.sslSocketFactory para o objeto de factory customizado. Detalhes variam entre diferentes implementações de JSSE.

Diferentes implementações de JSSE podem fornecer diferentes recursos. Por exemplo, uma implementação de JSSE especializada pode permitir a configuração de um modelo específico de hardware de criptografia. Além disso, alguns provedores JSSE permitem a customização de keystores e de armazenamentos confiáveis por programa ou permitem que a opção de certificado de identidade do keystore seja alterada. No JSSE, todas essas customizações são abstraídas em uma classe de factory, javax.net.ssl.SSLSocketFactory.

Consulte a documentação de seu JSSE para obter detalhes sobre como criar uma implementação de SSLSocketFactory customizada. Os detalhes variam de provedor para provedor, mas uma sequência típica de etapas pode ser:

1. Criar um objeto SSLContext usando um método estático em SSLContext
2. Inicializar esse SSLContext com implementações de KeyManager e TrustManager apropriadas (criadas a partir de suas próprias classes de factory)
3. Criar um SSLSocketFactory a partir do SSLContext

Quando tiver um objeto SSLSocketFactory, configure o MQEnvironment.sslSocketFactory para o objeto de factory customizado. Por exemplo:

```
javax.net.ssl.SSLSocketFactory sf = sslContext.getSocketFactory();
MQEnvironment.sslSocketFactory = sf;
```

IBM WebSphere MQ classes for Java use este SSLSocketFactory para conectar ao gerenciador de filas do IBM WebSphere MQ . Essa propriedade também pode ser configurada usando CMQC.SSL_SOCKET_FACTORY_PROPERTY. Se sslSocketFactory estiver configurado como null, o SSLSocketFactory padrão da JVM será usado. Essa propriedade será ignorada se sslCipherSuite não estiver configurado.

Ao usar SSLSocketFactories customizados, considere o efeito de compartilhamento de conexão TCP/IP. Se o compartilhamento de conexão for possível, então, um novo soquete não será solicitado do SSLSocketFactory fornecido, mesmo se o soquete produzido fosse diferente de alguma maneira no contexto de uma solicitação de conexão subsequentes. Por exemplo, se um certificado de cliente diferente precisar ser apresentado em uma conexão subsequente, então, o compartilhamento de conexões não deverá ser permitido.

Fazendo mudanças no keystore ou armazenamento confiável do JSSE nas classes do WebSphere MQ para Java

Se você mudar o keystore ou armazenamento confiável JSSE, deverá executar determinadas ações para que as mudanças entrem em vigor.

Se você mudar o conteúdo do keystore ou do armazenamento confiável do JSSE ou mudar o local do keystore ou do arquivo de armazenamento confiável, as classes do WebSphere MQ para aplicativos Java que estão em execução no momento não selecionarão automaticamente as mudanças Para que as mudanças entrem em vigor, as seguintes ações devem ser executadas:

- Os aplicativos devem fechar todas as suas conexões e destruir quaisquer conexões não usadas em conjuntos de conexões.
- Se seu provedor JSSE armazenar em cache informações do keystore e do armazenamento confiável, essas informações deverão ser atualizadas.

Após essas ações terem sido executadas, os aplicativos poderão, então, recriar suas conexões.

Dependendo de como você projeta seus aplicativos e sobre a função fornecida pelo seu provedor JSSE, pode ser possível executar estas ações sem parar e reiniciar seus aplicativos. No entanto, parar e reiniciar o aplicativo poderá ser a solução mais simples.

Manipulação de erros ao usar SSL com classes WebSphere MQ para Java

Vários códigos de razão podem ser emitidos por classes WebSphere MQ para Java ao se conectarem a um gerenciador de filas usando SSL

Eles são explicados na lista a seguir:

MQRC_SSL_NOT_ALLOWED
A propriedade sslCipherSuite foi configurada, mas a conexão de ligações foi usada. Somente a conexão do cliente suporta SSL.

MQRC_JSSE_ERROR
O provedor JSSE relatou um erro que não pôde ser tratado pelo WebSphere MQ. Isso pode ter sido causado por um problema de configuração com o JSSE ou porque o certificado apresentado pelo gerenciador de filas não pôde ser validado. A exceção produzida pelo JSSE pode ser recuperada usando o método getCause() em MQException.

MQRC_SSL_INITIALIZATION_ERROR
Uma chamada MQCONN ou MQCONNX foi emitida com opções de configuração de SSL especificadas, mas ocorreu um erro durante a inicialização do ambiente SSL.

MQRC_SSL_PEER_NAME_MISMATCH
O padrão de DN especificado na propriedade sslPeerName não correspondia ao DN apresentado pelo gerenciador de filas.

MQRC_SSL_PEER_NAME_ERROR
O padrão de DN especificado na propriedade sslPeerName não era válido.

MQRC_UNSUPPORTED_CIPHER_SUITE
O CipherSuite denominado no sslCipherSuite não foi reconhecido pelo provedor JSSE. Uma lista completa de CipherSuites suportados pelo provedor JSSE pode ser obtida por um programa, usando o método SSLSocketFactory.getSupportedCipherSuites(). Uma lista de CipherSuites que pode ser usada para se comunicar com o WebSphere MQ pode ser localizada em "SSL CipherSpecs e CipherSuites no WebSphere MQ classes para Java" na página 714

MQRC_SSL_CERTIFICATE_REVOKED
O certificado apresentado pelo gerenciador de filas foi localizado em uma CRL especificada com a propriedade sslCertStores. Atualize o gerenciador de filas para usar certificados confiáveis.

MQRC_SSL_CERT_STORE_ERROR
Nenhum dos CertStores fornecidos pôde ser procurado pelo certificado apresentado pelo gerenciador de filas. O método MQException.getCause() retorna o erro que ocorreu ao faze a primeira tentativa de procura de CertStore. Se a exceção causal for NoSuchElementException, ClassCastException ou NullPointerException, verifique se o Collection especificado na propriedade sslCertStores contém pelo menos um objeto CertStore válido.

SSL CipherSpecs e CipherSuites no WebSphere MQ classes para Java

Se um aplicativo IBM WebSphere MQ classes for Java pode estabelecer uma conexão com um gerenciador de filas depende do CipherSpec especificado na extremidade do servidor do canal MQI e do CipherSuite especificado na extremidade do cliente.

Para cada combinação de CipherSpec e CipherSuite, se um aplicativo IBM WebSphere MQ classes for Java pode se conectar a um gerenciador de filas depende do valor do campo sslFipsNecessário na classe MQEnvironment ou do valor da propriedade do ambiente CMQC.SSL_FIPS_REQUIRED_PROPERTY

Na extremidade do servidor de um canal MQI, o nome de um CipherSpec pode ser especificado como o valor do parâmetro SSLCIPH em um comando DEFINE CHANNEL CHLTYPE (SVRCONn). Na extremidade do cliente de um canal MQI, um aplicativo do IBM WebSphere MQ classes for Java pode configurar o campo sslCipherSuite na classe MQEnvironment ou configurar a propriedade de ambiente CMQC.SSL_CIPHER_SUITE_PROPERTY.

Configurando seu aplicativo para usar mapeamentos do IBM Java ou Oracle Java CipherSuite

A partir do IBM WebSphere MQ Version 7.5.0, Fix Pack 5, é possível configurar se seu aplicativo usa o IBM Java CipherSuite padrão para mapeamentos do WebSphere MQ CipherSpec ou o Oracle CipherSuite para mapeamentos do WebSphere MQ CipherSpec . Portanto, é possível usar TLS CipherSuites se seu aplicativo usar um JRE IBM ou um JRE Oracle . A Propriedade de sistema Java `com.ibm.mq.cfg.useIBMCipherMappings` controla quais mapeamentos são usados. A propriedade pode ser um dos valores a seguir:

true
Use o IBM Java CipherSuite para WebSphere MQ CipherSpec mapeamentos.

Esse valor é o valor padrão.

false
Use os mapeamentos Oracle CipherSuite para WebSphere MQ CipherSpec .

A tabela a seguir lista os CipherSpecs suportados pelo IBM WebSphere MQ e seus CipherSuites equivalentes. A tabela também indica se um aplicativo IBM WebSphere MQ classes for Java pode estabelecer uma conexão com um gerenciador de filas se um CipherSpec for especificado na extremidade do servidor do canal MQI e o CipherSuite equivalente for especificado na extremidade do cliente.

Tabela 89. CipherSpecs suportados pelo WebSphere MQ e seus CipherSuites equivalentes

CipherSpec	Equivalente CipherSuite	Conexão possível se o SFIPS [1] estiver configurado como YES?
NULL_MD5	SSL_RSA_WITH_NULL_MD5	Não
NULL_SHA	SSL_RSA_WITH_NULL_SHA	Não
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5 (JREIBM) Nenhum equivalente para Oracle JRE.	Não
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5	Não
RC4_SHA_US	SSL_RSA_WITH_RC4_128_SHA (IBM JRE) Nenhum equivalente para Oracle JRE.	Não
RC2_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (IBM JRE) SSL_RSA_EXPORT_WITH_RC4_40_MD5 (Oracle JRE)	Não
DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA (IBM JRE) Nenhum equivalente para Oracle JRE.	Não
RC4_56_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_RC4_56_SHA (JREIBM) Nenhum equivalente para Oracle JRE.	Não

Tabela 89. CipherSpecs suportados pelo WebSphere MQ e seus CipherSuites equivalentes (continuação)

CipherSpec	**Equivalente CipherSuite**	**Conexão possível se o SFIPS [1] estiver configurado como YES?**
DES_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA (IBM JRE) Nenhum equivalente para Oracle JRE.	Não
TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA (IBM JRE) Nenhum equivalente para Oracle JRE.	Não
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256 (IBM JRE) TLS_RSA_WITH_NULL_SHA256 (Oracle JRE)	Nenhum [7]
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA (IBM JRE) TLS_RSA_WITH_AES_128_CBC_SHA (Oracle JRE)	Sim [5] [7]
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256 (IBM JRE) TLS_RSA_WITH_AES_128_CBC_SHA256 (Oracle JRE)	Sim [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA (IBM JRE) TLS_RSA_WITH_AES_256_CBC_SHA (Oracle JRE)	Sim [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256 (IBM JRE) TLS_RSA_WITH_AES_256_CBC_SHA256 (Oracle JRE)	Sim [5] [7]
AES_SHA_US[2]		
TLS_RSA_WITH_DES_CBC_SHA[8]	SSL_RSA_WITH_DES_CBC_SHA	Não[3]
TLS_RSA_WITH_3DES_EDE_CBC_SHA[8] [9]	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Sim
FIPS_WITH_DES_CBC_SHA	SSL_RSA_FIPS_WITH_DES_CBC_SHA (IBM JRE) Nenhum equivalente para Oracle JRE.	Nenhum[4]
FIPS_WITH_3DES_EDE_CBC_SHA	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (IBM JRE) Nenhum equivalente para Oracle JRE.	Nenhum[6]

Notes:

1. Em um aplicativo IBM WebSphere MQ classes for Java , indique que apenas algoritmos certificados por FIPS devem ser usados configurando o campo sslFipsObrigatório na classe MQEnvironment como `true` e indique que os algoritmos não certificados por FIPS também podem ser usados configurando o campo sslFipsObrigatório como `false` Como alternativa, configure a propriedade de ambiente CMQC.SSL_FIPS_REQUIRED_PROPERTY..
2. Este CipherSpec não tem nenhum CipherSuiteequivalente

3. Esse CipherSpec foi certificado pelo FIPS 140-2 antes de 19th de maio de 2007
4. Esse CipherSpec foi certificado pelo FIPS 140-2 antes de 19th de maio de 2007 O nome FIPS_WITH_DES_CBC_SHA é histórico e reflete o fato de que esse CipherSpec era anteriormente (mas não é mais) compatível com FIPS. Esse CipherSpec foi descontinuado e seu uso não é recomendado.
5. Estes CipherSpecs (TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) não podem ser usados para proteger uma conexão do WebSphere MQ Explorer para um gerenciador de filas, a menos que os arquivos de política não restritos apropriados sejam aplicados ao JRE.

   Consulte Informações de Segurança para obter informações adicionais sobre os arquivos de políticas
6. O nome FIPS_WITH_3DES_EDE_CBC_SHA é histórico e reflete o fato de que esse CipherSpec era anteriormente (mas não é mais) compatível com FIPS.. Esse CipherSpec foi descontinuado e seu uso não é recomendado.
7. Esses CipherSpecs (TLS_RSA_WITH_NULL_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) requerem IBM JREs 6.0 SR13 FP2 , 7.0 SR4 FP2 ou posterior.
8. Essas CipherSpecs (TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_DES_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA256) podem usar SSLv3 ou TLS. Por padrão, quando FIPS não está ativado, SSLv3 é usado. Para usar TLS, configure a Propriedade do sistema Java **com.ibm.mq.cfg.preferTLS** como `true`
9. Esse CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA foi descontinuado. No entanto, ele ainda pode ser usado para transferir até 32 GB de dados antes de a conexão ser finalizada com erro AMQ9288. Para evitar esse erro, você precisará evitar o uso de DES triplo ou ativar a reconfiguração de chave secreta ao usar esse CipherSpec.

Informações relacionadas

Especificando que Apenas CipherSpecs Certificados por FIPS São Usados no Tempo de Execução no Cliente de MQI

Federal Information Processing Standards (FIPS) para UNIX, Linux e Windows

Blog MQdev: MQ Java, Ciphers TLS, JREs e APARs nãoIBM IT06775, IV66840, IT09423, IT10837

Blog MQdev: O relacionamento entre MQ CipherSpecs e Java Cipher Suites

Executando classes WebSphere MQ para aplicativos Java

Se você gravar um aplicativo (uma classe que contém um método main ()), utilizando o cliente ou o modo de ligações, execute o programa utilizando o interpretador Java.

Use o comando:

```
java -Djava.library.path=library_path MyClass
```

em que *library_path* é o caminho para as classes do WebSphere MQ para bibliotecas Java (consulte O WebSphere MQ classes para bibliotecas Java).

WebSphere MQ classes para comportamento dependente do ambiente Java

As classes do WebSphere MQ para Java permitem criar aplicativos que podem ser executados em diferentes versões do WebSphere MQ. Esta coleção de tópicos descreve o comportamento das classes Java dependentes dessas diferentes versões

WebSphere MQ classes para Java fornece um núcleo de classes, que fornecem função e comportamento consistentes em todos os ambientes. Os recursos fora desse núcleo dependem do recurso do gerenciador de filas ao qual o aplicativo está conectado.

Exceto quando indicado aqui, o comportamento exibido é conforme descrito na Referência de programação de aplicativos apropriada para o gerenciador de filas.

Classes principais no WebSphere MQ classes para Java

WebSphere MQ classes para Java contém um conjunto principal de classes, que pode ser usado em todos os ambientes.

O seguinte conjunto de classes é considerado de classes principais e pode ser usado em todos os ambientes com apenas as variações menores listadas em "Restrições e variações para classes principais de classes do WebSphere MQ para Java" na página 719.

- MQEnvironment
- MQException
- MQGetMessageOptions

  Excluindo:

  - MatchOptions
  - GroupStatus
  - SegmentStatus
  - Segmentação
- MQManagedObject

  Excluindo:

  - inquire()
  - set()
- MQMessage

  Excluindo:

  - groupId
  - messageFlags
  - messageSequenceNumber
  - deslocamento
  - originalLength
- MQPoolServices
- MQPoolServicesEvent
- MQPoolServicesEventListener
- MQPoolToken
- MQPutMessageOptions

  Excluindo:

  - knownDestCount
  - unknownDestCount
  - invalidDestCount
  - recordFields
- MQProcess
- MQQueue
- MQQueueManager

  Excluindo:

  - begin()
  - accessDistributionList()
- MQSimpleConnectionManager
- MQTopic

- MQC

Nota:

1. Algumas constantes não são incluídas no núcleo (consulte “Restrições e variações para classes principais de classes do WebSphere MQ para Java” na página 719 para obter detalhes); não as use em programas completamente portáteis.
2. Algumas plataformas não suportam todos os modos de conexão. Nessas plataformas, é possível usar apenas as classes principais e as opções relacionadas aos modos suportados. (Veja “Opções de Conexão para Classes WebSphere MQ para Java” na página 661.)

Restrições e variações para classes principais de classes do WebSphere MQ para Java

As classes principais geralmente se comportam de forma consistente em todos os ambientes, mesmo se as chamadas MQI equivalentes normalmente tiverem diferenças de ambiente. O comportamento é como se um gerenciador de filas Windows, UNIX ou Linux WebSphere MQ fosse usado, exceto para as restrições e variações menores a seguir.

*Restrições para valores MQGMO_ * em classes WebSphere MQ para Java*
Determinados valores MQGMO_* não são suportados por todos os gerenciadores de fila.

O uso dos valores MQGMO_* a seguir pode resultar em uma MQException sendo emitida a partir de um MQQueue.get():

MQGMO_SYNCPOINT_IF _PERSISTENT
MQGMO_MARK_SKIP _BACKOUT
MQGMO_BROWSE_MSG_UNDER_CURSOR
MQGMO_LOCK
MQGMO_UNLOCK
MQGMO_LOGICAL_ORDER
MQGMO_COMPLETE_MESSAGE
MQGMO_ALL_MSGS_AVAILABLE
MQGMO_ALL_SEGMENTS_ AVAILABLE
MQGMO_UNMARKED_BROWSE_MSG
MQGMO_MARK_BROWSE_HANDLE
MQGMO_MARK_BROWSE_CO_OP
MQGMO_UNMARK_BROWSE_HANDLE
MQGMO_UNMARK_BROWSE_CO_OP

Além disso, MQGMO_SET_SIGNAL não é suportado quando usado de Java.

*Restrições para valores MQPMRF_ * em classes WebSphere MQ para Java*
Elas são usadas apenas ao colocar mensagens em uma lista de distribuição e são suportadas apenas por gerenciadores de filas que suportam listas de distribuição. Por exemplo, os gerenciadores de filas z/OS não suportam listas de distribuição.

*Restrições para valores MQPMO_ * em WebSphere MQ classes para Java*
Determinados valores de MQPMO_* não são suportados por todos os gerenciadores de filas

O uso dos valores de MQPMO_* a seguir pode resultar em uma MQException sendo emitida a partir de um MQQueue.put() ou um MQQueueManager.put():

MQPMO_LOGICAL_ORDER
MQPMO_NEW_CORREL_ID
MQPMO_NEW_MESSAGE_ID
MQPMO_RESOLVE_LOCAL_Q

*Restrições e variações para valores MQCNO_ * em classes WebSphere MQ para Java*
Determinados valores de MQCNO_* não são suportados.

- A reconexão automática do cliente não é suportada pelas classes WebSphere MQ para Java. Qualquer valor de MQCNO_RECONNECT_* que você configurar, a conexão continua a se comportar como se MQCNO_RECONNECT_DISABLED estivesse configurado.
- MQCNO_FASTPATH é ignorado em gerenciadores de filas que não suportam MQCNO_FASTPATH. Ele também é ignorado pelas conexões do cliente.

*Restrições para valores MQRO_ * em classes WebSphere MQ para Java*
As seguintes opções de relatório podem ser configuradas.

MQRO_EXCEPTION_WITH_FULL_DATA
MQRO_EXPIRATION_WITH_FULL_DATA
MQRO_COA_WITH_FULL_DATA
MQRO_COD_WITH_FULL_DATA
MQRO_DISCARD_MSG
MQRO_PASS_DISCARD_AND_EXPIRY

Para obter mais informações, consulte Relatar

Recursos fora das classes principais das classes do WebSphere MQ para Java

As classes do WebSphere MQ para Java contêm determinadas funções que são projetadas especificamente para usar extensões de API que não são suportadas por todos os gerenciadores de fila Esta coleção de tópicos descreve como elas se comportam ao usar um gerenciador de filas que *não* as suporta.

Variações na opção de construtor MQQueueManager
Alguns dos construtores MQQueueManager incluem um argumento de número inteiro opcional. Alguns valores desse argumento não são aceitos em todas as plataformas.

Quando um construtor MQQueueManager inclui um argumento de número inteiro opcional, ele é mapeado para o campo de opções MQCNO da MQI e é usado para alternar entre conexão normal e de atalho. Este formato estendido do construtor é aceito em todos os ambientes, se as únicas opções usadas forem MQCNO_STANDARD_BINDING ou MQCNO_FASTPATH_BINDING. Quaisquer outras opções fazem o construtor falhar com MQRC_OPTIONS_ERROR. A opção de atalho CMQC.MQCNO_FASTPATH_BINDING é honrada somente com uma conexão de ligações para um gerenciador de filas que a suporte. Em outros ambientes, ela será ignorada.

Restrições no método MQQueueManager.begin()
Esse método pode ser usado apenas em um gerenciador de filas do WebSphere MQ nos sistemas UNIX, Linuxou Windows no modo de ligações. Caso contrário, ele falha com MQRC_ENVIRONMENT_ERROR.

Consulte “Coordenação JTA/JDBC usando classes WebSphere MQ para Java” na página 706 para obter mais detalhes.

Variações nos campos MQGetMessageOptions
Alguns gerenciadores de filas não suportam a estrutura MQGMO Versão 2, portanto, deve-se configurar alguns campos para seus valores padrão.

Ao usar um gerenciador de filas que não suporta a estrutura MQGMO Versão 2, deixe os campos a seguir definidos para seus valores padrão:

GroupStatus
SegmentStatus
Segmentação

Além disso, o campo MatchOptions suporta apenas MQMO_MATCH_MSG_ID e MQMO_MATCH_CORREL_ID. Se você colocar valores não suportados nesses campos, o MQDestination.get() subsequente falhará com MQRC_GMO_ERROR. Se o gerenciador de filas não suportar a estrutura MQGMO Versão 2, esses campos não serão atualizados após um MQDestination.get() bem-sucedido.

Restrições em listas de distribuição em classes do WebSphere MQ para Java
Nem todos os gerenciadores de fila permitem abrir uma MQDistributionList.

As classes a seguir são usadas para criar listas de distribuição:

MQDistributionList
MQDistributionListItem
MQMessageTracker

É possível criar e preencher MQDistributionLists e MQDistributionListItems em qualquer ambiente, mas nem todos os gerenciadores de filas permitem abrir uma MQDistributionList. Em particular, os gerenciadores de fila do z/OS não suportam listas de distribuição Tentativa de abrir um MQDistributionList ao usar esse gerenciador de filas resulta em MQRC_OD_ERROR.

Variações nos campos MQPutMessageOptions
Se um gerenciador de filas não suportar listas de distribuição, determinados campos MQPMO serão tratados de forma diferente.

Quatro campos no MQPMO são renderizados como as variáveis de membro a seguir na classe MQPutMessageOptions:

knownDestCount
unknownDestCount
invalidDestCount
recordFields

Esses campos são destinados principalmente para uso com listas de distribuição. No entanto, um gerenciador de filas que suporta listas de distribuição também preenche os campos DestCount após um MQPUT para uma única fila. Por exemplo, se a fila for resolvida para uma fila local, knownDestCount será configurado para 1 e os outros dois campos de contagem serão configurados para 0.

Se o gerenciador de filas não suporta listas de distribuição, esses valores serão simuladas da seguinte forma:

- Se o put() for bem-sucedido, unknownDestCount será configurado para 1 e os outros serão configurados para 0.
- Se o put() falhar, invalidDestCount será configurado para 1 e os outros serão configurados para 0.

A variável recordFields é usada com listas de distribuição. Um valor pode ser gravado em recordFields a qualquer momento, independentemente do ambiente. Ele é ignorado se o objeto MQPutMessageOptions for usado em um MQDestination.put() ou MQQueueManager.put() subsequente, em vez de MQDistributionList.put().

Restrições em campos MQMD com classes WebSphere MQ para Java
Alguns campos do MQMD com respeito a segmentação de mensagens devem ser deixados com seus valores padrão ao usar um gerenciador de filas que não suporta segmentação.

Os seguintes campos MQMD são amplamente envolvidos com segmentação de mensagens:

GroupId
MsgSeqNumber
Offset
MsgFlags
OriginalLength

Se um aplicativo configura qualquer um desses campos do MQMD para valores diferentes de seus padrões e, em seguida, executa um put() ou get() em um gerenciador de filas que não os suporta, put() ou get() gerarão uma MQException com MQRC_MD_ERROR. Um put() ou get() bem-sucedido com esse gerenciador de filas sempre deixa os campos do MQMD configurados para seus valores padrão. Não envie uma mensagem agrupada ou segmentada para um aplicativo Java que é executado em um gerenciador de filas que não suporta agrupamento e segmentação de mensagens.

Se um aplicativo Java tentar obter () uma mensagem de um gerenciador de filas que não suporte esses campos e a mensagem física a ser recuperada fizer parte de um grupo de mensagens segmentadas (ou seja, ele tiver valores não padrão para os campos MQMD), ele será recuperado sem erro No entanto, os campos do MQMD no MQMessage não são atualizados, a propriedade de formato MQMessage é definida como MQFMT_MD_EXTENSION e os dados da mensagem verdadeira são prefixadas com uma estrutura MQMDE que contém os valores para os novos campos.

Restrições para classes do WebSphere MQ para Java no CICS Transaction Server

No ambiente do CICS Transaction Server for z/OS , apenas o encadeamento principal (primeiro) tem permissão para emitir chamadas CICS ou WebSphere MQ .

Observe que as classes JMS do WebSphere MQ não são suportadas para uso em um aplicativo Java do CICS

Portanto, não é possível compartilhar objetos MQQueueManager ou MQQueue entre encadeamentos nesse ambiente ou criar um novo MQQueueManager em um encadeamento filho.

Executando classes IBM WebSphere MQ para aplicativos Java dentro da Java plataforma Enterprise Edition

Há determinadas restrições e considerações de design que devem ser levadas em conta antes de usar as classes IBM WebSphere MQ para Java no Java EE

As classes IBM WebSphere MQ para Java têm restrições quando usadas em um ambiente Java EE Há também considerações adicionais que devem ser levadas em conta ao projetar, implementar e gerenciar um aplicativo IBM WebSphere MQ classes para Java que é executado dentro de um ambiente do Java EE Essas restrições e contraprestações são descritas nas seções a seguir.

Restrições de transações JTA

O único gerenciador de transações suportado para aplicativos usando classes IBM WebSphere MQ para Java é o próprio IBM WebSphere MQ . Embora um aplicativo sob o controle JTA possa usar classes IBM WebSphere MQ para Java, qualquer trabalho executado por meio dessas classes não é controlado pelas unidades de trabalho JTA. Ao invés disso, elas formam as unidades de trabalho locais a partir dessas gerenciadas pelo servidor de aplicativos através das interfaces de JTA. Em especial, qualquer retrocesso da transação de JTA não resulta em um retrocesso de quaisquer mensagens enviadas ou recebidas. Essa restrição se aplica a transações gerenciadas por aplicativo ou bean e a transações gerenciada por contêiner e a todos os contêineres do Java EE Para executar o trabalho do sistema de mensagens diretamente com IBM WebSphere MQ dentro de transações coordenadas pelo servidor de aplicativos, as classes do IBM WebSphere MQ para JMS devem ser usadas no lugar.

Criação de encadeamento

IBM WebSphere MQ classes para Java cria encadeamentos internamente para várias operações. Por exemplo, ao executar no modo BINDINGS para chamar diretamente em um gerenciador de filas locais, as chamadas são feitas em um encadeamento 'worker' criado internamente pelas classes IBM WebSphere MQ para Java Outros encadeamentos podem ser criados internamente, por exemplo, para limpar conexões não usadas a partir de um conjunto de conexões ou para remover assinaturas para aplicativos de publicação/assinatura finalizados.

Alguns aplicativos Java EE (por exemplo, aqueles em execução em contêineres EJB e Web) não devem criar novos encadeamentos. Em vez disso, todo o trabalho deve ser executado nos encadeamentos do aplicativo principal gerenciado pelo servidor de aplicativos. Quando os aplicativos usam classes IBM WebSphere MQ para Java, o servidor de aplicativos pode não ser capaz de distinguir entre o código do aplicativo e as classes IBM WebSphere MQ para o código Java , portanto, os encadeamentos descritos anteriormente fazem o aplicativo ficar fora de conformidade com a especificação do contêiner. As classes IBM WebSphere MQ para JMS não quebram essas especificações Java EE e, portanto, podem ser usadas no lugar.

Restrições de segurança

As políticas de segurança implementadas por um servidor de aplicativos podem evitar determinadas operações que são executadas pela API IBM WebSphere MQ classes para Java , como criar e operar novos encadeamentos de controle (conforme descrito nas seções anteriores).

Por exemplo, servidores de aplicativos geralmente são executados com o Java Security desativado por padrão e permite que seja ativado através de alguma configuração específica do servidor de aplicativo (alguns servidores de aplicativos também permitem a configuração mais detalhada das políticas usadas dentro do Java Security). Quando a Java Segurança está ativada, as classes IBM WebSphere MQ para Java podem quebrar as regras de encadeamento da política de segurança do Java definidas para o servidor de aplicativos e a API pode não ser capaz de criar todos os encadeamentos necessários para funcionar. Para evitar problemas com o gerenciamento de encadeamento, o uso de classes IBM WebSphere MQ para Java não é suportado em ambientes em que a Segurança do Java está ativada.

Contraprestações de isolamento do aplicativo

Um benefício desejado da execução de aplicativos em um ambiente Java EE é o isolamento do aplicativo. O design e a implementação de classes IBM WebSphere MQ para Java antecedem o ambiente Java EE . IBM WebSphere MQ para o Java pode ser usado de uma maneira que não suporta o conceito de isolamento de aplicativo Os exemplos de contraprestações específicas nesta área incluem:

- O uso de configurações estáticas (todo o processo da JVM) dentro da classe MQEnvironment, como:
  - o ID do usuário e a senha a serem usados para autenticação e identificação de conexão
  - o nome do host, porta e canal usados para conexões do cliente
  - configuração de SSL para conexões do cliente seguras

  A modificação de qualquer uma das propriedades MQEnvironment para o benefício de um aplicativo também afeta outros aplicativos usando as mesmas propriedades. Ao executar em um ambiente de vários aplicativos, como o Java EE, cada aplicativo deve usar sua própria configuração distinta por meio da criação de objetos MQQueueManager com um conjunto específico de propriedades, em vez de padronizar para as propriedades configuradas na classe MQEnvironment do processo.
- A classe MQEnvironment introduz um número de métodos estáticos que agem globalmente em todos os aplicativos usando IBM WebSphere MQ classes para Java dentro do mesmo processo JVM e não há maneira de substituir esse comportamento para aplicativos específicos. Os exemplos incluem:
  - configuração de propriedades de SSL, como o local do keystore
  - configurando as saídas de canais do cliente
  - ativando ou desativando o rastreio de diagnóstico
  - gerenciando o conjunto de conexões padrão usado para otimizar o uso de conexões para os gerenciadores de filas

  Chamar esses métodos afeta todos os aplicativos em execução no mesmo ambiente Java EE .
- A definição do conjunto de conexões é ativada para otimizar o processo de fazer várias conexões no mesmo gerenciador de filas. O gerenciador de conjunto de conexões padrão é todo o processo e compartilhado por vários aplicativos. Mudanças na configuração do conjunto de conexão, como substituir o gerenciador de conexões padrão para um aplicativo usando o método MQEnvironment.setDefaultConnectionManager(), portanto, afeta outros aplicativos em execução no mesmo servidor de aplicativos Java EE .
- SSL é configurado para aplicativos usando classes IBM WebSphere MQ para Java usando a classe MQEnvironment e as propriedades de objeto MQQueueManager . Ele não está integrado com a configuração de segurança gerenciada do próprio servidor de aplicativos. Deve-se assegurar que você configure classes IBM WebSphere MQ para Java adequadamente para fornecer seu nível necessário de segurança e não usar a configuração do servidor de aplicativos.

Restrições do modo de ligações

IBM WebSphere MQ e WebSphere Application Server podem ser instalados na mesma máquina, de forma que as versões principais do gerenciador de filas e do IBM WebSphere MQ resource adapter (RA) enviado no WebSphere Application Server sejam diferentes. Por exemplo, WebSphere Application Server Version 7.0, que envia um IBM WebSphere MQ nível RA de 7.0.1, pode ser instalado na mesma máquina que um gerenciador de filas do Version 6.0 .

Se as versões principais do adaptador de recursos e do gerenciador de filas forem diferentes, as conexões de ligações não poderão ser usadas. Quaisquer conexões do WebSphere Application Server para o gerenciador de fila usando o adaptador de recursos devem usar conexões do tipo de cliente As conexões de ligações poderão ser usadas se as versões forem as mesmas.

Usando classes do WebSphere MQ para JMS

WebSphere MQ classes para Java Message Service (WebSphere MQ classes para JMS) é o provedor JMS que é fornecido com o WebSphere MQ. Além de implementar as interfaces definidas no pacote javax.jms , as classes WebSphere MQ para JMS fornecem dois conjuntos de extensões para a API JMS.

A especificação JMS define um conjunto de interfaces que os aplicativos podem usar para executar operações do sistema de mensagens O pacote javax.jms define as interfaces JMS e um provedor JMS implementa essas interfaces para um produto de sistema de mensagens específico. O WebSphere MQ versão 7.5 usa atualmente a especificação JMS 1.1 WebSphere MQ classes para JMS é um provedor JMS que implementa as interfaces JMS para WebSphere MQ.

A especificação JMS espera que ConnectionFactory e objetos de Destino sejam objetos administrados. Um administrador cria e mantém objetos administrados em um repositório central e um aplicativo JMS recupera esses objetos usando a JNDI (Java Naming and Directory Interface) WebSphere MQ classes para JMS suporta o uso de objetos administrados, e um administrador pode usar o WebSphere MQ JMS ferramenta de administração ou WebSphere MQ Explorer para criar e manter objetos administrados.

WebSphere MQ classes para JMS também fornece dois conjuntos de extensões para a API JMS. O principal foco dessas extensões refere-se à criação e configuração de connection factories e destinos dinamicamente no tempo de execução, mas as extensões também fornecem a função que não está diretamente relacionada ao sistema de mensagens, como a função para determinação de problema.

As extensões JMS do WebSphere MQ
As liberações anteriores das classes WebSphere MQ para JMS contêm extensões que são implementadas em objetos como objetos MQConnectionFactory, MQQueue e MQTopic. Esses objetos possuem propriedades e métodos específicos do WebSphere MQ. Os objetos podem ser objetos administrados ou um aplicativo pode criar os objetos dinamicamente no tempo de execução. Esta liberação do WebSphere MQ classes para JMS mantém essas extensões, que agora são conhecidas como as extensões JMS do WebSphere MQ É possível continuar usando, sem mudança, quaisquer aplicativos que usam essas extensões.

As extensões JMS IBM
Esta liberação de classes do WebSphere MQ para JMS fornece um conjunto mais genérico de extensões para a API JMS, que não são específicas para o WebSphere MQ como o sistema de mensagens Essas extensões são conhecidas como as extensões JMS do IBM e têm os seguintes objetivos gerais:

- Para fornecer um nível maior de consistência nos provedores JMS do IBM
- Para facilitar a gravação de um aplicativo de ponte entre dois sistemas de mensagens IBM
- Para facilitar a porta de um aplicativo de um provedor JMS do IBM para outro

As extensões fornecem uma função semelhante àquela fornecida em Message Service Client for C/C++ e Message Service Client for .NET.

Por que devo usar classes do WebSphere MQ para JMS?

O uso das classes WebSphere MQ para JMS tem as seguintes vantagens:

- É possível reutilizar qualificações JMS.

  WebSphere MQ classes para JMS é um provedor JMS que implementa as interfaces JMS para WebSphere MQ como o sistema de mensagens. Se sua organização for nova no WebSphere MQ, mas já tiver qualificações de desenvolvimento de aplicativo JMS, você poderá achar mais fácil usar a API JMS familiar para acessar recursos do WebSphere MQ em vez de uma das outras APIs fornecidas com o WebSphere MQ.
- JMS é uma parte integrante do Java Platform, Enterprise Edition (Java EE).

  JMS é a API natural para usar para o sistema de mensagens na plataforma Java EE . Cada servidor de aplicativos compatível com Java EE deve incluir um provedor JMS. É possível usar JMS em aplicativos clientes, servlets, JavaServer páginas (JSPs), enterprise Java beans (EJBs), e beans acionados por mensagens (MDBs).. Observe em particular que os aplicativos Java EE usam MDBs para processar mensagens de forma assíncrona e todas as mensagens são entregues aos MDBs como mensagens JMS.
- Um administrador pode criar e manter objetos administrados JMS em um repositório central e as classes do WebSphere MQ para aplicativos JMS podem recuperar esses objetos usando o Java Naming and Directory Interface (JNDI).

  As connection factories e os destinos JMS encapsulam informações específicas do WebSphere MQ , como nomes de gerenciadores de filas, nomes de canais, opções de conexão, nomes de filas e nomes de tópicos. Se connection factories e destinos forem armazenados como objetos administrados, essas informações não serão codificadas permanentemente em um aplicativo. Este acordo, portanto, fornece ao aplicativo um grau de independência da configuração subjacente do WebSphere MQ .
- JMS é uma API padrão de mercado que pode fornecer a portabilidade do aplicativo

  Um aplicativo JMS pode usar JNDI para recuperar connection factories e destinos armazenados como objetos administrados e usar apenas as interfaces definidas no pacote javax.jms para executar operações do sistema de mensagens. O aplicativo é, então, totalmente independente de qualquer provedor JMS, como WebSphere MQ classes para JMS e pode ser portado de um provedor JMS para outro sem qualquer mudança no aplicativo.

  Se JNDI não estiver disponível em um ambiente de aplicativos específico, uma classe WebSphere MQ para o aplicativo JMS poderá usar extensões para a API JMS para criar e configurar connection factories e destinos dinamicamente no tempo de execução. O aplicativo é, então, completamente autocontido, mas está ligado às classes WebSphere MQ para JMS como o provedor JMS.
- Aplicativos de ponte podem ser mais fáceis de gravar usando JMS.

  Um aplicativo de ligação é um aplicativo que recebe as mensagens a partir de um sistema de mensagens e as envia para outro sistema de mensagens. A gravação de um aplicativo de ponte pode ser complicada usando APIs e formatos de mensagens específicos do produto Em vez disso, é possível gravar um aplicativo de bridge usando dois provedores JMS, um para cada sistema de mensagens. O aplicativo então usa apenas uma API, a API JMS e processa apenas mensagens JMS.

Introdução às classes do WebSphere MQ para JMS

Este tópico fornece uma visão geral das classes do WebSphere MQ para JMS e informa o que você precisa saber antes de usar classes do WebSphere MQ para JMS.

Pré-requisitos para classes do WebSphere MQ para JMS

Para desenvolver e executar classes WebSphere MQ para aplicativos JMS, são necessários determinados componentes de software como pré-requisitos.

Para obter as informações mais recentes sobre os pré-requisitos para as classes WebSphere MQ para JMS, consulte o arquivo leia-me WebSphere MQ .

Para desenvolver classes WebSphere MQ para aplicativos JMS, é necessário um Java 2 Software Development Kit (SDK). Os detalhes dos JDKs suportados com seu sistema operacional podem ser localizados na página de requisitos do sistema WebSphere MQ Consulte WebSphere MQ Requisitos.

Para executar classes WebSphere MQ para aplicativos JMS, você precisa dos seguintes componentes de software:

- Um gerenciador de filas do WebSphere MQ
- Um Java Runtime Environment (JRE), para cada sistema no qual você executa aplicativos

Se você precisar de conexões SSL para usar módulos criptográficos que são certificados pelo FIPS 140-2, precisará do provedor Java JSSE FIPS IBM (IBMJSSEFIPS). Cada IBM Java 2 SDK e JRE na Versão 5 ou posterior contém IBMJSSEFIPS.

É possível usar Internet Protocol Versão 6 (IPv6) endereços em suas classes WebSphere MQ para aplicativos JMS fornecidos IPv6 endereços são suportados por sua Java virtual machine (JVM) e a implementação TCP/IP em seu sistema operacional. A ferramenta de administração JMS do WebSphere MQ (consulte “Usando a ferramenta de administração JMS do WebSphere MQ” na página 946) também aceita endereços IPv6 .

A ferramenta de administração JMS do WebSphere MQ e o WebSphere MQ Explorer usam o Java Naming and Directory Interface (JNDI) para acessar um serviço de diretório, que armazena objetos administrados. As classes do WebSphere MQ para aplicativos JMS também podem usar JNDI para recuperar objetos administrados de um serviço de diretório. Um provedor de serviços é o código que fornece acesso a um serviço de diretório pelo mapeamento de chamadas JNDI para as chamadas ao serviço de diretório. Os provedores de serviços a seguir são fornecidos com classes WebSphere MQ para JMS:

- Um provedor de serviços LDAP (Lightweight Directory Access Protocol) nos arquivos ldap.jar e providerutil.jar. O provedor de serviços LDAP fornece acesso a um serviço de diretório baseado em um servidor LDAP.
- Um provedor de serviço do sistema de arquivos nos arquivos fscontext.jar e providerutil.jar. O provedor de serviços do sistema de arquivos fornece acesso a um serviço de diretório baseado no sistema de arquivos local.

Se você pretende usar um serviço de diretório baseado em um servidor LDAP, deve-se instalar e configurar um servidor LDAP ou ter acesso a um servidor LDAP existente. Especificamente, você deve configurar o servidor LDAP para armazenar objetos Java. Para obter informações sobre como instalar e configurar seu servidor LDAP, consulte a documentação que é fornecida com o servidor.

Preparando programas JMS para o cliente IBM WebSphere MQ para HP Integrity NonStop Server

Este tópico explica o que você precisa saber antes de desenvolver e executar programas JMS para o cliente do IBM WebSphere MQ para HP Integrity NonStop Server

As classes IBM WebSphere MQ para JMS são instaladas como parte do cliente IBM WebSphere MQ para HP Integrity NonStop Server instalação. Para obter detalhes de um resumo do conteúdo da instalação, consulte Sistema de arquivo.

Alguns aspectos de funcionalidade do cliente são específicos para o sistema operacional do host. Para obter mais informações sobre os recursos suportados para o IBM WebSphere MQ cliente para HP Integrity NonStop Server, consulte IBM WebSphere MQ cliente para HP Integrity NonStop Server ambientes e recursos suportados.

Pré-requisitos

Para construir e executar aplicativos JMS, o componente *HP Integrity NonStop Server for Java* deve estar instalado e disponível..

Instalação

Para obter informações sobre como configurar o ambiente para executar e construir aplicativos nos quais é possível usar as classes do IBM WebSphere MQ para JMS, consulte “Variáveis de ambiente usadas por classes IBM WebSphere MQ para JMS” na página 731

Para informações sobre as etapas necessárias para configurar um gerenciador de filas para aceitar conexões de aplicativos clientes, consulte “Configuração de pós-instalação para classes WebSphere MQ para aplicativos JMS” na página 781.

Para obter informações sobre como validar suas classes IBM WebSphere MQ para o ambiente JMS, consulte “O teste de verificação de instalação ponto a ponto para classes do WebSphere MQ para JMS” na página 785.

Gravando aplicativos

Para obter mais informações sobre a gravação de aplicativos JMS, consulte “Gravando classes do WebSphere MQ para aplicativos JMS” na página 816

Para obter mais informações sobre como usar a ferramenta de administração do JMS do IBM WebSphere MQ , consulte “Usando a ferramenta de administração JMS do WebSphere MQ” na página 946

Amostras

Os aplicativos de amostra são fornecidos no seguinte subdiretório da instalação: `opt/mqm/samp/jms`.

Para obter mais informações sobre as etapas de configuração necessárias para executar as amostras, consulte “Preparando e executando os programas de amostra” na página 111.

Resolução de problemas

Para obter informações sobre a resolução de problemas, consulte “Resolvendo problemas com classes IBM WebSphere MQ para JMS” na página 806.

Instalação e Configuração de Classes WebSphere MQ para JMS

Esta seção descreve os diretórios e arquivos que são criados ao instalar classes do WebSphere MQ para JMS e informa como configurar classes do WebSphere MQ para JMS após a instalação.

Conceitos relacionados

“O que é instalado para classes IBM WebSphere MQ para JMS” na página 728
Vários arquivos e diretórios são criados ao instalar classes do IBM WebSphere MQ para JMS. No Windows , alguma configuração é executada durante a instalação configurando automaticamente variáveis de ambiente. Em outras plataformas, e em determinados ambientes Windows, você deve configurar variáveis de ambiente antes de poder executar classes do IBM WebSphere MQ para aplicativos JMS.

“Executando classes WebSphere MQ para aplicativos JMS sob o gerenciador de segurança Java” na página 738
As classes WebSphere MQ para JMS podem ser executadas com o gerenciador de segurança Java ativado. Para executar aplicativos com sucesso com o gerenciador de segurança ativado, deve-se configurar a máquina virtual Java (JVM) com um arquivo de configuração de política adequado.

“O adaptador de recursos IBM WebSphere MQ” na página 742
O adaptador de recursos permite que aplicativos em execução em um servidor de aplicativos acessem recursos do IBM WebSphere MQ . Ele suporta a comunicação de entrada e de saída.

“Configuração de pós-instalação para classes WebSphere MQ para aplicativos JMS” na página 781
Este tópico informa quais autoridades WebSphere MQ classes para aplicativos JMS precisam para acessar os recursos de um gerenciador de filas. Ele também introduz os modos de conexão e descreve como configurar um gerenciador de filas para que os aplicativos possam se conectar no modo cliente.

“O teste de verificação de instalação ponto a ponto para classes do WebSphere MQ para JMS” na página 785
Um programa de teste de verificação de instalação ponto a ponto (IVT) é fornecido com classes WebSphere MQ para JMS. O programa se conecta a um gerenciador de filas no modo de ligações ou de cliente, envia uma mensagem à fila chamada SYSTEM.DEFAULT.LOCAL.QUEUE e, em seguida, recebe a mensagem da fila. O programa pode criar e configurar todos os objetos que requer dinamicamente no

tempo de execução ou pode usar JNDI para recuperar objetos administrados a partir de um serviço de diretório.

"O teste de verificação de instalação de publicação / assinatura para classes WebSphere MQ para JMS" na página 788
Um programa de teste de verificação de instalação (IVT) de publicação / assinatura é fornecido com classes WebSphere MQ para JMS. O programa se conecta a um gerenciador de filas no modo de ligações ou do cliente, assina um tópico, publica uma mensagem sobre o tópico e, em seguida, recebe a mensagem que acaba de publicar. O programa pode criar e configurar todos os objetos que requer dinamicamente no tempo de execução ou pode usar JNDI para recuperar objetos administrados a partir de um serviço de diretório.

"O programa de teste de verificação de instalação para o adaptador de recursos WebSphere MQ" na página 792
O programa IVT é fornecido como um arquivo EAR. Para usar o programa, deve-se implementá-lo e definir alguns objetos como recursos JCA.

"Configurando o adaptador de recursos para comunicação de saída" na página 761
Para configurar a comunicação de saída, defina as propriedades de um objeto ConnectionFactory e um objeto de destino administrado.

"Suporte para OSGi" na página 805
OSGi fornece uma estrutura que suporta a implementação de aplicativos como pacotes configuráveis. Nove pacotes configuráveis do OSGi são fornecidos como parte do IBM WebSphere MQ classes for JMS.

"Resolvendo problemas com classes IBM WebSphere MQ para JMS" na página 806
É possível investigar os problemas, executando os programas de verificação de instalação e utilizando os recursos de rastreio e de log.

Tarefas relacionadas

"Instalando e testando o adaptador de recursos MQ no WAS CE" na página 795
Instalando o adaptador de recursos IBM WebSphere MQ e executando o aplicativo de teste de verificação de instalação (IVT) no WebSphere Application Server CE.

"Implementando o aplicativo IVT no WAS CE com um ambiente customizado do MQ" na página 797
Se você desejar usar uma fila, um gerenciador de filas, uma porta, um host, um canal ou um modo de ligações diferentes em vez do modo cliente, deverá modificar o aplicativo IVT e os scripts associados no WebSphere Application Server CE antes de implementar o adaptador de recursos ou o aplicativo IVT.

"Implementando o aplicativo IVT no JBoss com um ambiente IBM WebSphere MQ customizado" na página 800
Ao instalar o adaptador de recursos do IBM WebSphere MQ no JBoss, se você desejar usar uma fila diferente, um gerenciador de filas, uma porta, um host, um canal ou um modo de ligações em vez do modo de cliente, deverá primeiro modificar o aplicativo IVT e os scripts associados no JBoss antes de implementar o adaptador de recursos ou o aplicativo IVT.

Determinação de problemas para o adaptador de recursos do IBM WebSphere MQ

Referências relacionadas

"Scripts fornecidos com classes do WebSphere MQ para JMS" na página 804
Vários scripts são fornecidos para ajudar com tarefas comuns que precisam ser executadas ao usar classes WebSphere MQ para JMS.

O que é instalado para classes IBM WebSphere MQ para JMS

Vários arquivos e diretórios são criados ao instalar classes do IBM WebSphere MQ para JMS. No Windows , alguma configuração é executada durante a instalação configurando automaticamente variáveis de ambiente. Em outras plataformas, e em determinados ambientes Windows, você deve configurar variáveis de ambiente antes de poder executar classes do IBM WebSphere MQ para aplicativos JMS.

Para a maioria dos sistemas operacionais, as classes IBM WebSphere MQ para JMS são instaladas como um componente opcional ao instalar o IBM WebSphere MQ. Para o cliente IBM WebSphere MQ para o HP

Integrity NonStop Server, as classes IBM WebSphere MQ para JMS são instaladas por padrão. Para obter mais informações sobre a instalação do IBM WebSphere MQ, consulte:

Instalando um Servidor do WebSphere MQ
Instalando um cliente IBM WebSphere MQ

A Tabela 90 na página 729 mostra onde as classes IBM WebSphere MQ para arquivos JMS são instaladas em cada plataforma

Tabela 90. Classes IBM WebSphere MQ para Diretórios de Instalação JMS

Plataforma	**Diretório**
AIX	*MQ_INSTALLATION_PATH*/java
HP Integrity NonStop Server	*MQ_INSTALLATION_PATH*/opt/mqm/java
HP-UX, Linuxe Solaris	*MQ_INSTALLATION_PATH*/java
Windows	*MQ_INSTALLATION_PATH*\java
O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o IBM WebSphere MQ está instalado.	

O diretório de instalação inclui:

- As classes do IBM WebSphere MQ para arquivos JAR JMS, que estão localizados no diretório *MQ_INSTALLATION_PATH*\java\lib
- As bibliotecas nativas do IBM WebSphere MQ , que são usadas por aplicativos que usam a Interface Nativa Java

  As bibliotecas nativas de 32 bit bits são instaladas no diretório *MQ_INSTALLATION_PATH*\java\lib e as bibliotecas nativas de 64 bit podem ser localizadas no diretório *MQ_INSTALLATION_PATH*\java\lib64

  Para obter informações adicionais sobre as bibliotecas nativas do IBM WebSphere MQ, consulte "Configurando as bibliotecas do Java Native Interface (JNI)" na página 733.
- Scripts adicionais descritos em "Scripts fornecidos com classes do WebSphere MQ para JMS" na página 804.. Esses scripts estão localizados em um diretório *MQ_INSTALLATION_PATH*\java\bin
- As especificações das classes IBM WebSphere MQ para API JMS. A ferramenta Javadoc foi usada para gerar as páginas HTML que contêm as especificações da API.

  As páginas HTML estão no diretório *MQ_INSTALLATION_PATH*\java\doc\WMQJMSClasses..

  Em sistemas UNIX, Linuxe Windows, esse subdiretório contém as páginas HTML individuais
- Suporte para OGSi. Pacotes configuráveis OSGi são instalados no diretório java \lib\OSGi e descritos em "Suporte para OSGi" na página 805.
- O IBM WebSphere MQ Resource Adapter, que pode ser implementado em qualquer servidor de aplicativos compatível com JCA 1.5 (ou posterior).

  O Adaptador de Recursos IBM WebSphere MQ está localizado no diretório *MQ_INSTALLATION_PATH*\java\lib\jca; para obter informações adicionais, consulte "O adaptador de recursos IBM WebSphere MQ" na página 742
- No Windows, símbolos que podem ser usados para depuração são instalados no diretório *MQ_INSTALLATION_PATH*\java\lib\symbols.

O diretório de instalação também inclui alguns arquivos que pertencem a outros componentes do IBM WebSphere MQ. Esses diretórios são os seguintes:

- O transporte IBM WebSphere MQ para SOAP, que fornece um transporte JMS para SOAP, é instalado no diretório *MQ_INSTALLATION_PATH*\java\lib\soap. Para obter informações adicionais sobre o transporte do IBM WebSphere MQ para SOAP, consulte a seção do centro de informações que descreve "Transporte do WebSphere MQ para SOAP" na página 959

- Em plataformas distribuídas, o IBM WebSphere MQ Bridge for HTTP é instalado no diretório *MQ_INSTALLATION_PATH*\java\lib\http.. Para obter informações adicionais sobre a ponte IBM WebSphere MQ para HTTP, consulte a seção do centro de informações que descreve "WebSphere MQ ponte para HTTP" na página 1035

Alguns aplicativos de amostra são fornecidos com classes IBM WebSphere MQ para JMS. O Tabela 91 na página 730 mostra onde os aplicativos de amostra são instalados em cada plataforma.

Tabela 91. Diretórios de amostras

Plataforma	**Diretório**
AIX	*MQ_INSTALLATION_PATH*/samp/jms
HP Integrity NonStop Server	*MQ_INSTALLATION_PATH*/opt/mqm/samp/jms
HP-UX, Linuxe Solaris	*MQ_INSTALLATION_PATH*/samp/jms
Windows	*MQ_INSTALLATION_PATH*\tools\jms
O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o IBM WebSphere MQ está instalado.	

Após a instalação, pode ser necessário executar algumas tarefas de configuração para compilar e executar aplicativos.

"Variáveis de ambiente usadas por classes IBM WebSphere MQ para JMS" na página 731 descreve o classpath necessário para executar classes simples do IBM WebSphere MQ para aplicativos JMS. Este tópico também descreve arquivos JAR adicionais que precisam ser referenciados em circunstâncias especiais e as variáveis de ambiente que devem ser configuradas para executar os scripts fornecidos com classes IBM WebSphere MQ para JMS.

Caso você precise de suas classes IBM WebSphere MQ para o aplicativo JMS para vincular ao código gravado em linguagens diferentes de Java (por exemplo, para usar o transporte de ligações ao conectar-se a um Gerenciador de Filas), "Configurando as bibliotecas do Java Native Interface (JNI)" na página 733 explica onde localizar o local das bibliotecas Java Native Interface (JNI) para especificar como um parâmetro do comando Java.

Para controlar propriedades, como o rastreio e criação de log de um aplicativo, será necessário fornecer um arquivo de propriedades de configuração. As classes IBM WebSphere MQ para o arquivo de propriedades de configuração do JMS são descritas em "O arquivo de configuração IBM WebSphere MQ classes for JMS" na página 735

Instalando e fazendo upgrade das classes do WebSphere MQ para arquivos JAR JMS

A única maneira suportada de obter as classes IBM WebSphere MQ para arquivos JAR JMS em um sistema é instalar o produto IBM WebSphere MQ ou o WebSphere MQ V7.5 Clients SupportPac-MQC75ou usando uma ferramenta de gerenciamento de software como o Apache Maven, para obter mais informações, consulte "IBM WebSphere MQ classes for JMS e ferramentas de gerenciamento de software" na página 737.

Não mova ou copie as classes IBM WebSphere MQ para arquivos JAR JMS ou bibliotecas nativas para outras máquinas ou para um local diferente em uma máquina em que as classes IBM WebSphere MQ para JMS foram instaladas, a menos que você esteja usando uma ferramenta de gerenciamento de software.

- Os fix packs não podem ser aplicados a uma "instalação" na qual os arquivos JAR foram copiados de outra máquina, pois isso torna muito mais difícil assegurar que todos os arquivos JAR sejam mantidos em etapas entre si e estejam em níveis compatíveis..
- Copiar as classes IBM WebSphere MQ para arquivos JAR JMS entre máquinas também pode resultar em várias cópias dos arquivos que residem na mesma máquina, o que pode causar problemas de manutenção do código e problemas de depuração..

- O comando dspmqver , usado para exibir informações de versão de uma instalação do IBM WebSphere MQ , exibe apenas informações de versão para as classes IBM WebSphere MQ para JMS instaladas no diretório `\java\lib` .

  Se várias cópias dos arquivos residirem na mesma máquina, a execução de **dspmqver** poderá não fornecer informações precisas sobre a versão das classes IBM WebSphere MQ para JMS que estão sendo usadas por um aplicativo

Não inclua as classes IBM WebSphere MQ para arquivos JAR JMS nos arquivos de aplicativos (como archives de aplicativo corporativo ou arquivos EAR)...

- As atualizações para as classes IBM WebSphere MQ para JMS não podem ser aplicadas usando um Fix Pack IBM WebSphere MQ .
- Não é possível que o Suporte IBM determine facilmente a versão das classes do IBM WebSphere MQ para JMS que estão sendo usadas pelo aplicativo
- Podem surgir problemas se vários aplicativos em execução dentro do mesmo Java Runtime Environment incluírem diferentes versões das classes IBM WebSphere MQ para JMS, já que várias versões das classes IBM WebSphere MQ para JMS são carregadas no Java Runtime Environment ao mesmo tempo.

  Exemplos desses problemas incluem as seguintes exceções:

```
java.lang.ClassCastException :
com.ibm.mq.jmqi.system.JmqiSystemEnvironment incompatible with
com.ibm.mq.jmqi.system.JmqiSystemEnvironment

java.lang.ClassCastException :
com.ibm.mq.jms.MQQueue incompatible with com.ibm.mq.jms.MQQueue
```

- Se um aplicativo usar o transporte BINDINGS para se conectar a um gerenciador de filas, quaisquer upgrades principais para o gerenciador de filas também exigirão que o aplicativo seja atualizado para incluir o nível correspondente das classes IBM WebSphere MQ para JMS

  Por exemplo, se um gerenciador de filas for atualizado para o nível IBM WebSphere MQ Versão 7.5 , quaisquer aplicativos que se conectem ao gerenciador de filas usando o transporte BINDINGS também precisarão ser atualizados para incluir as classes IBM WebSphere MQ Version 7.5 para JMS.

Variáveis de ambiente usadas por classes IBM WebSphere MQ para JMS

Antes de poder compilar e executar as classes IBM WebSphere MQ para aplicativos JMS, a configuração para sua variável de ambiente CLASSPATH deve incluir as classes IBM WebSphere MQ para o arquivo JAR (Java archive) JMS. Dependendo de seus requisitos, pode ser necessário incluir outros arquivos JAR no caminho de classe. Para executar os scripts fornecidos com classes IBM WebSphere MQ para JMS, outras variáveis de ambiente devem ser configuradas.

Para compilar e executar classes IBM WebSphere MQ para aplicativos JMS, use a configuração CLASSPATH para sua plataforma conforme mostrado em Tabela 92 na página 731. A configuração inclui o diretório de amostras para que seja possível compilar e executar as classes IBM WebSphere MQ para aplicativos de amostra JMS. Como alternativa, é possível especificar o caminho de classe no comando **java** em vez de usar a variável de ambiente.

Tabela 92. Configuração de CLASSPATH para compilar e executar classes IBM WebSphere MQ para aplicativos JMS, incluindo os aplicativos de amostra

Plataforma	**Configuração de CLASSPATH**
AIX	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms:
HP Integrity NonStop Server	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms:

Tabela 92. Configuração de CLASSPATH para compilar e executar classes IBM WebSphere MQ para aplicativos JMS, incluindo os aplicativos de amostra (continuação)

Plataforma	**Configuração de CLASSPATH**
HP-UX, Linuxe Solaris	CLASSPATH=*MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms:
IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.jar: /QIBM/ProdData/mqm/java/samples/jms:
Windows	CLASSPATH=*MQ_INSTALLATION_PATH*\java\lib\com.ibm.mqjms.jar; *MQ_INSTALLATION_PATH*\tools\jms;
z/OS	CLASSPATH=*MQ_INSTALLATION_PATH*/mqm/V7R0M0/java/lib/ com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/mqm/V6R0M0/java/samples/jms:
O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o IBM WebSphere MQ está instalado.	

O manifesto do arquivo JAR com.ibm.mqjms.jar contém referências à maioria dos outros arquivos JAR necessários pelas classes IBM WebSphere MQ para aplicativos JMS e, portanto, não é necessário incluir esses arquivos JAR em seu caminho de classe. Esses arquivos JAR incluem aqueles requeridos por aplicativos que usam o Java Naming and Directory Interface (JNDI) para recuperar objetos administrados de um serviço de diretório e por aplicativos que usam o Java Transaction API (JTA).

No entanto, deve-se incluir arquivos JAR adicionais em seu caminho de classe nas circunstâncias a seguir:

- Se estiver usando classes de saída de canal que implementam as interfaces de saída de canal definidas no pacote com.ibm.mq , em vez daquelas definidas no pacote com.ibm.mq.exits , deve-se incluir as classes IBM WebSphere MQ para o arquivo JAR Java, com.ibm.mq.jar, em seu caminho de classe.
- Se você compilar seu código Java usando um Java 2 Software Development Kit (SDK) na Versão 1.4.2, deverá incluir os seguintes arquivos JAR em seu caminho de classe:
  - jms.jar
  - com.ibm.mq.jmqi.jar

  Além disso, se seu aplicativo usar JNDI para recuperar objetos administrados de um serviço de diretório, você também deverá incluir os seguintes arquivos JAR em seu caminho de classe:
  - fscontext.jar
  - jndi.jar
  - ldap.jar
  - providerutil.jar

  E se seu aplicativo usar a JTA, deve-se incluir também o jta.jar em seu caminho de classe.

  Observe que esses arquivos JAR adicionais são necessários somente para compilar seus aplicativos, não para executá-los.

Se você compilar usando a opção -Xlint, poderá ver uma mensagem que avise que o com.ibm.mq.ese.jar não está presente. É possível ignorar o aviso. Esse arquivo estará presente apenas se você tiver instalado o Extended Security Edition (Edição de Segurança Estendida)

Os scripts fornecidos com classes IBM WebSphere MQ para JMS usam as variáveis de ambiente a seguir:

MQ_JAVA_DATA_PATH
Essa variável de ambiente especifica o diretório para log e saída de rastreio.

MQ_JAVA_INSTALL_PATH
Essa variável de ambiente especifica o diretório no qual as classes WebSphere MQ para JMS estão instaladas.

MQ_JAVA_LIB_PATH
Essa variável de ambiente especifica o diretório no qual as classes WebSphere MQ para bibliotecas JMS são armazenadas, conforme mostrado em Tabela 93 na página 734.

No Windows, todas as variáveis de ambiente são configuradas automaticamente durante a instalação. Em qualquer outra plataforma, deve-se configurá-las sozinho.

Para configurar as variáveis de ambiente se você estiver usando uma JVM de 32 bits em sistemas UNIX, HP Integrity NonStop Server, ou Linux, é possível usar o script setjmsenv. Para configurar as variáveis de ambiente, se você estiver usando uma JVM de 64 bits em um sistema UNIX ou Linux, é possível usar o script setjmsenv64. Estes scripts estão no diretório *MQ_INSTALLATION_PATH*`/java/bin`, em que *MQ_INSTALLATION_PATH* representa o diretório de alto nível em que o IBM WebSphere MQ está instalado.

É possível usar o script setjmsenv ou setjmsenv64 de uma variedade de maneiras: é possível usá-lo como base para configurar as variáveis de ambiente necessárias, conforme mostrado na tabela ou incluí-las em `.profile` usando um editor de texto. Se você tiver uma configuração atípica, edite o conteúdo do script conforme necessário. Como alternativa, é possível executar o script em cada sessão a partir da qual scripts de inicialização JMS devem ser executados. Se você escolher essa opção, será necessário executar o script em cada janela shell iniciada durante o processo de verificação JMS digitando `. ./setjmsenv` ou `. ./setjmsenv64`

Configurando as bibliotecas do Java Native Interface (JNI)

Classes IBM WebSphere MQ para aplicativos JMS, que se conectam a um gerenciador de filas usando o transporte de ligações ou que se conectam a um gerenciador de filas usando o transporte do cliente e usam os programas de saída do canal gravados em linguagens diferentes de Java, precisam ser executados em um ambiente que acesse as bibliotecas Java Native Interface (JNI).

Sobre esta tarefa

Para configurar esse ambiente, deve-se configurar o caminho da biblioteca do ambiente para que a Java virtual machine (JVM) possa carregar a biblioteca mqjbnd antes de iniciar as classes IBM WebSphere MQ para o aplicativo JMS.

O IBM WebSphere MQ fornece duas bibliotecas Java Native Interface (JNI):

mqjbnd
Esta biblioteca é usada pelos aplicativos que se conectam a um gerenciador de filas que usam o transporte de ligações. Ele fornece a interface entre as classes IBM WebSphere MQ para JMS e o gerenciador de filas. A biblioteca mqjbnd instalada com IBM WebSphere MQ Versão 7.5 pode ser usada para se conectar a qualquer gerenciador de filas IBM WebSphere MQ Versão 7.5 (ou anterior).

mqjexitstub02
A biblioteca mqjexitstub02 é carregada pelas classes IBM WebSphere MQ para JMS quando um aplicativo se conecta a um gerenciador de fila usando o transporte do cliente e usa um programa de saída do canal gravado em uma linguagem diferente de Java

Em algumas plataformas, o IBM WebSphere MQ instala as versões de 32 bist e 64 bits destas bibliotecas JNI. O local das bibliotecas para cada plataforma é mostrado na Tabela 1

Tabela 93. O local das classes IBM WebSphere MQ para bibliotecas JMS para cada plataforma	
Plataforma	**Diretório que contém as classes do IBM WebSphere MQ para bibliotecas JMS**
AIX	*MQ_INSTALLATION_PATH*/java/lib (bibliotecas de 32 bits) *MQ_INSTALLATION_PATH*/java/lib64 (bibliotecas de 64 bits)
HP-UX	*MQ_INSTALLATION_PATH*/java/lib (bibliotecas de 32 bits) *MQ_INSTALLATION_PATH*/java/lib64 (bibliotecas de 64 bits)
Linux (POTÊNCIA , x86-64 e zSeries s390x plataformas)	*MQ_INSTALLATION_PATH*/java/lib (bibliotecas de 32 bits) *MQ_INSTALLATION_PATH*/java/lib64 (bibliotecas de 64 bits)
Linux (plataformax86) Linux (Plataforma zSeries)	*MQ_INSTALLATION_PATH* /java/lib
Solaris (plataformasx86-64 e SPARC)	*MQ_INSTALLATION_PATH*/java/lib (bibliotecas de 32 bits) *MQ_INSTALLATION_PATH*/java/lib64 (bibliotecas de 64 bits)
Windows	*MQ_INSTALLATION_PATH*\java\lib (bibliotecas de 32 bits) *MQ_INSTALLATION_PATH*\java\lib64 (bibliotecas de 64 bits)
O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o IBM WebSphere MQ está instalado.	

Procedimento

1. Configure a propriedade **java.library.path** da JVM, que pode ser feito de duas maneiras:
   - Especificando o argumento JVM conforme mostrado no exemplo a seguir:

     ```
     -Djava.library.path=<path_to_library_directory>
     ```

     Linux Por exemplo, para uma JVM de 64 bits no Linux para uma instalação de local padrão, especifique:

     ```
     -Djava.library.path=/opt/mqm/java/lib64
     ```

   - Configurando o ambiente de shell de forma que a JVM irá configurar seu próprio `java.library.path`. Esse caminho varia por plataforma e pelo local no qual você instalou o IBM WebSphere MQ. Por exemplo, para uma JVM de 64 bits e um local de instalação padrão do IBM WebSphere MQ, é possível usar as configurações a seguir:

AIX `export LIBPATH=/usr/mqm/java/lib64:$LIBPATH`

Solaris HP-UX Linux
```
export LD_LIBRARY_PATH=/opt/mqm/java/
lib64:$LD_LIBRARY_PATH
```

Windows `set PATH=C:\Program Files\IBM\MQ\java\lib64;%PATH%`

Um exemplo da pilha de exceções que você vê quando o ambiente não foi configurado corretamente é o seguinte:

```
Causado por: com.ibm.mq.jmqi.local.LocalMQ$4: CC=2;RC=2495;
AMQ8598: Falha ao carregar a biblioteca JNI nativa do WebSphere MQ: 'mqjbnd'.
    em com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1268)
    em com.ibm.mq.jmqi.local.LocalMQ$1.run(LocalMQ.java:309)
    em java.security.AccessController.doPrivileged(AccessController.java:400)
    em com.ibm.mq.jmqi.local.LocalMQ.initialise_inner(LocalMQ.java:259)
    em com.ibm.mq.jmqi.local.LocalMQ.initialise(LocalMQ.java:221)
    at com.ibm.mq.jmqi.local.LocalMQ.<init>(LocalMQ.java:1350)
    at com.ibm.mq.jmqi.local.LocalServer.<init>(LocalServer.java:230)
    em sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
    em
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:86)

    em
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.jav
a:58)
    em java.lang.reflect.Constructor.newInstance(Constructor.java:542)
    em com.ibm.mq.jmqi.JmqiEnvironment.getInstance(JmqiEnvironment.java:706)
    em com.ibm.mq.jmqi.JmqiEnvironment.getMQI(JmqiEnvironment.java:640)
    em
com.ibm.msg.client.wmq.factories.WMQConnectionFactory.createV7ProviderConnection(WMQConnectionF
actory.java:8437)
    ... 7 mais
Causado por: java.lang.UnsatisfiedLinkError: mqjbnd (não localizado em java.library.path)
    em java.lang.ClassLoader.loadLibraryWithPath(ClassLoader.java:1235)
    em java.lang.ClassLoader.loadLibraryWithClassLoader(ClassLoader.java:1205)
    em java.lang.System.loadLibrary(System.java:534)
    em com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1240)
    ... 20 mais
```

2. Após o ambiente de 32 bits ou 64 bits ter sido configurado, inicie as classes IBM WebSphere MQ para o aplicativo JMS usando o comando:

```
java application-name
```

em que *application-name* é o nome das classes do IBM WebSphere MQ para o aplicativo JMS a ser executado

Uma exceção contendo IBM WebSphere MQ Código de Razão 2495 (MQRC_MODULE_NOT_FOUND) é lançada pelas classes IBM WebSphere MQ para JMS se:

- As classes IBM WebSphere MQ para o aplicativo JMS são executadas em um ambiente de tempo de execução Java de 32 bits e um ambiente de 64 bits foi configurado para as classes IBM WebSphere MQ para JMS, pois o ambiente de tempo de execução Java de 32 bits não pode carregar a Biblioteca Nativa Java de 64 bits.
- As classes IBM WebSphere MQ para o aplicativo JMS são executadas em um ambiente de tempo de execução Java de 64 bits e um ambiente de 32 bits foi configurado para as classes IBM WebSphere MQ para JMS, pois o ambiente de tempo de execução Java de 64 bits não pode carregar a Biblioteca Nativa Java de 32 bits.

O arquivo de configuração IBM WebSphere MQ classes for JMS

Uma classe WebSphere MQ para o arquivo de configuração JMS especifica propriedades que são usadas para configurar classes WebSphere MQ para JMS.

O formato de um WebSphere MQ classes para o arquivo de configuração JMS é aquele de um arquivo de propriedades Java padrão. Um arquivo de configuração de amostra chamado jms.config é fornecido

no subdiretório `bin` das classes do WebSphere MQ para o diretório de instalação JMS. Este arquivo documenta todas as propriedades suportadas e seus valores padrão.

É possível escolher o nome e o local de um WebSphere MQ classes para o arquivo de configuração JMS. Ao iniciar o seu aplicativo, use um comando **java** com o seguinte formato:

```
java -Dcom.ibm.msg.client.config.location=config_file_url application_name
```

No comando, *config_file_url* é um localizador uniforme de recursos (URL) que especifica o nome e o local das classes WebSphere MQ para o arquivo de configuração JMS. URLs dos seguintes tipos são suportadas: http, arquivo, ftp e jar.

A seguir está um exemplo de um comando **java**:

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config MyAppClass
```

Esse comando identifica as classes WebSphere MQ para o arquivo de configuração JMS como o arquivo D:\mydir\mjms.config no sistema Windows local.

Quando um aplicativo é iniciado, as classes WebSphere MQ para JMS lê o conteúdo do arquivo de configuração e armazena as propriedades especificadas em um armazenamento de propriedade interno. Se o comando **java** não identificar um arquivo de configuração ou se o arquivo de configuração não puder ser localizado, as classes WebSphere MQ para JMS usarão os valores padrão para todas as propriedades. Se necessário, será possível substituir qualquer propriedade no arquivo de configuração especificando-a como uma propriedade de sistema no comando **java**.

Um arquivo de configuração do WebSphere MQ classes para JMS pode ser utilizado com qualquer um dos transportes suportados entre um aplicativo e um gerenciador de filas ou broker

Observe que você não pode especificar o rastreio de inicialização configurando uma propriedade nas classes WebSphere MQ para o arquivo de configuração JMS. É possível especificar o rastreio de inicialização apenas configurando uma propriedade do sistema no comando **java** , conforme mostrado no exemplo a seguir:

```
java -Dcom.ibm.msg.client.commonservices.trace.startup=true
     -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config
      MyAppClass
```

Substituindo propriedades especificadas em um arquivo de configuração do cliente MQI do WebSphere MQ

Um arquivo de configuração do cliente MQI do WebSphere MQ também pode especificar propriedades que são usadas para configurar WebSphere MQ classes para JMS. No entanto, as propriedades especificadas em um arquivo de configuração do cliente MQI do WebSphere MQ se aplicam apenas quando um aplicativo se conecta a um gerenciador de filas no modo cliente.

Se necessário, é possível substituir qualquer atributo em um arquivo de configuração do cliente MQI do WebSphere MQ especificando-o como uma propriedade em uma WebSphere MQ classes para arquivo de configuração JMS. Para substituir um atributo em um arquivo de configuração do cliente MQI do WebSphere MQ , use uma entrada com o seguinte formato nas classes WebSphere MQ para o arquivo de configuração JMS:

```
com.ibm.mq.cfg.stanza.propName=propValue
```

As variáveis na entrada possuem os seguintes significados:

stanza
O nome da sub-rotina no arquivo de configuração do cliente MQI do WebSphere MQ que contém o atributo

propName
O nome do atributo conforme especificado no arquivo de configuração do cliente MQI do WebSphere MQ

propValue
O valor da propriedade que substitui o valor do atributo especificado no arquivo de configuração do cliente MQI do WebSphere MQ

Como alternativa, é possível substituir um atributo em um arquivo de configuração do cliente MQI do WebSphere MQ especificando a propriedade como uma propriedade do sistema no comando **java** . Use o formato anterior para especificar a propriedade como uma propriedade de sistema.

Apenas os atributos a seguir em um arquivo de configuração do cliente MQI do WebSphere MQ são relevantes para as classes do WebSphere MQ para JMS Se você especificar ou substituir outros atributos, isso não terá efeito.

Sub-rotina	**Atribuir**
Sub-rotina ClientExitPath do arquivo de configuração do cliente	Caminho Padrão das Saídas
Sub-rotina ClientExitPath do arquivo de configuração do cliente	ExitsDefaultPath64
Sub-rotina ClientExitPath do arquivo de configuração do cliente	JavaExitsClasspath
Sub-rotina MessageBuffer do arquivo de configuração do cliente	MaximumSize
Sub-rotina MessageBuffer do arquivo de configuração do cliente	PurgeTime
Sub-rotina MessageBuffer do arquivo de configuração do cliente	UpdatePercentage
Sub-rotina TCP do Arquivo de Configuração do Cliente	ClntRcvBufSize
Sub-rotina TCP do Arquivo de Configuração do Cliente	ClntSndBufSize
Sub-rotina TCP do Arquivo de Configuração do Cliente	Connect_Timeout
Sub-rotina TCP do Arquivo de Configuração do Cliente	KeepAlive

IBM WebSphere MQ classes for JMS e ferramentas de gerenciamento de software

Ferramentas de gerenciamento de software como Apache Maven podem ser usadas com o IBM WebSphere MQ classes for JMS.

Muitas das grandes organizações de desenvolvimento usam essas ferramentas para gerenciar centralmente os repositórios de bibliotecas de terceiros.

Os IBM WebSphere MQ classes for JMS são compostos por um número de arquivos JAR. Quando você está desenvolvendo aplicativos de linguagem Java usando esta API, uma instalação de um IBM WebSphere MQ Servidor, Cliente ou Cliente SupportPac é necessária na máquina na qual o aplicativo está sendo desenvolvido

Se desejar usar essa ferramenta e incluir os arquivos JAR que formam o IBM WebSphere MQ classes for JMS em um repositório gerenciado centralmente, os pontos a seguir deverão ser observados:

- Um repositório ou contêiner deve ser disponibilizado somente para os desenvolvedores em sua organização. Qualquer distribuição fora da organização não é permitida.
- O repositório precisa conter um conjunto completo e consistente de arquivos JAR a partir de uma liberação única ou fix pack do IBM WebSphere MQ.

- Você é responsável por atualizar o repositório com qualquer manutenção fornecida pelo Suporte do IBM

Para o IBM WebSphere MQ Version 7.5, os seguintes arquivos JAR precisam ser instalados no repositório:

- com.ibm.mqjms.jar.
- com.ibm.mq.jar
- com.ibm.mq.jmqi.jar
- com.ibm.mq.pcf.jar
- com.ibm.mq.headers.jar
- CL3Export.jar será necessário se você estiver usando IBM WebSphere MQ classes for JMS.
- CL3Nonexport.jar será necessário se você estiver usando IBM WebSphere MQ classes for JMS.
- jndi.jar será necessário se você estiver usando IBM WebSphere MQ classes for JMS.
- ldap.jar será necessário se você estiver usando IBM WebSphere MQ classes for JMS.
- rmm.jar será necessário se você estiver usando IBM WebSphere MQ classes for JMS.
- dhbcore.jar será necessário se você estiver usando IBM WebSphere MQ classes for JMS.
- jms.jar será necessário se você estiver usando o IBM WebSphere MQ classes for JMS.
- fscontext.jar será necessário se você estiver usando o IBM WebSphere MQ classes for JMS e acessando os objetos administrados pelo JMS que são armazenados em um contexto JNDI do sistema de arquivos..
- providerutil.jar se você estiver usando o IBM WebSphere MQ classes for JMS e acessando objetos administrados pelo JMS que estão armazenados em um contexto JNDI do sistema de arquivos.

Executando classes WebSphere MQ para aplicativos JMS sob o gerenciador de segurança Java

As classes WebSphere MQ para JMS podem ser executadas com o gerenciador de segurança Java ativado. Para executar aplicativos com sucesso com o gerenciador de segurança ativado, deve-se configurar a máquina virtual Java (JVM) com um arquivo de configuração de política adequado.

A maneira mais simples de fazer isso é alterar o arquivo de configuração de política fornecido com seu JRE Na maioria dos sistemas, esse arquivo é armazenado no caminho `lib/security/java.policy`, em relação a seu diretório JRE. É possível editar o arquivo de configuração de política usando seu editor preferido ou o programa `policytool` fornecido com seu JRE.

Importante: V 7.5.0.8 Sempre que possível, o termo *lista de permissões* substitui o termo *lista de desbloqueio*. Uma exceção são os nomes de propriedade do sistema Java a seguir.

Se você usar o mecanismo do gerenciador de segurança do Java com seu aplicativo, deverá conceder as permissões a seguir:

- FilePermission em qualquer arquivo da lista de permissões que você usa, com permissão de leitura para o modo de CUMPRIMENTO, permissão de gravação para o modo de DESCOBERTA.
- PropertyPermission (leitura) nas propriedades com.ibm.mq.jms.whitelist, com.ibm.mq.jms.whitelist.discover e com.ibm.mq.jms.whitelist.mode.

A lista de permissões ClassName é suportada com APAR IT14385 e IBM WebSphere MQ Version 7.5.0, Fix Pack 8. Para obter mais informações, consulte “ClassName lista de permissões no JMS ObjectMessage” na página 739.

Aqui está um exemplo de duas entradas em um arquivo de configuração de política que permitem que as classes do WebSphere MQ para JMS sejam executadas com êxito no gerenciador de segurança padrão:

```
grant codeBase "file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.jmqi.jar" {
  //Required
  permission java.util.PropertyPermission "user.name","read";
  permission java.util.PropertyPermission "os.name","read";
  //Required if mqclient.ini/mqs.ini configuration files are used
```

```
  permission java.io.FilePermission "/var/mqm/mqclient.ini","read";
  permission java.io.FilePermission "/var/mqm/mqs.ini","read";
  //For the client transport type.
  permission java.net.SocketPermission "*","connect";
  //For the bindings transport type.
  permission java.lang.RuntimePermission "loadLibrary.*";
  //For applications that use CCDT tables (access to the CCDT
AMQCLCHL.TAB)
  permission java.io.FilePermission
"/var/mqm/qmgrs/QMGR/@ipcc/AMQCLCHL.TAB","read";
  //For applications that use User Exits
  permission java.io.FilePermission "/var/mqm/exits/*","read";
  permission java.lang.RuntimePermission "createClassLoader";
  //Required for the z/OS platform
  permission java.util.PropertyPermission
"com.ibm.vm.bitmode","read";
};
grant codeBase "file:MQ_INSTALLATION_PATH/java/lib/com.ibm.mqjms.jar" {
  permission java.util.PropertyPermission "user.name","read";
  permission java.util.PropertyPermission "os.name","read";
  permission java.util.PropertyPermission "console.encoding","read";
  permission java.lang.RuntimePermission "setContextClassLoader";
  //tracing permissions
  permission java.util.PropertyPermission "com.ibm.msg.client.commonservices.*","read";
  permission java.util.PropertyPermission "MQJMS_TRACE_LEVEL","read";
  permission java.util.logging.LoggingPermission "control";
  //Wherever trace output is expected
  permission java.io.FilePermission "/tmp/*","read,write";
  //Required for the z/OS platform
  permission java.util.PropertyPermission
"com.ibm.vm.bitmode","read";
};
```

Notes:

- O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado.
- A primeira instrução `grant` contém as permissões requeridas pelas classes WebSphere MQ para JMS e a segunda instrução `grant` contém as permissões requeridas por uma classe WebSphere MQ para o aplicativo JMS.
- Para permitir que as classes do WebSphere MQ para JMS acessem os arquivos Java archive (JAR) de um aplicativo, inclua a seguinte permissão na primeira instrução `grant` :

```
permission java.io.FilePermission "/path_to_your_app/-", "read";
```

- Para usar essas instruções `grant` em seu arquivo de configuração de política, pode ser necessário modificar os nomes de caminho dependendo de onde você instalou as classes do WebSphere MQ para JMS e onde você armazena seus aplicativos.
- Os aplicativos de amostra fornecidos com classes WebSphere MQ para JMS e scripts para executá-los, não ativam o gerenciador de segurança.

V 7.5.0.8 ClassName lista de permissões no JMS ObjectMessage

V 7.5.0.8 Nas classes do WebSphere MQ para JMS, o suporte para a listagem de permissões de classes na implementação da interface JMS ObjectMessage fornece uma mitigação potencial com relação a alguns dos riscos de segurança potencialmente relacionados ao mecanismo de serialização e desserialização do objeto Java

Nota: Sempre que possível, o termo *lista de permissões* substitui o termo *lista de desbloqueio*. Uma exceção são os nomes de propriedade de sistema Java mencionados neste tópico.

O mecanismo de serialização e desserialização de objeto do Java foi identificado como um risco de segurança em potencial porque a desserialização instancia objetos arbitrários do Java, em que há muita possibilidade de dados serem enviados intencionalmente para causar vários problemas. Um aplicativo notável de serialização está em Java Serviço de Mensagens (JMS) ObjectMessages que usa a serialização para encapsular e transferir objetos arbitrários..

A listagem de permissões de serialização é uma mitigação em potencial em relação a alguns dos riscos que a serialização representa. Ao especificar explicitamente quais classes podem ser encapsuladas no ObjectMessages e extraídas dele, a listagem de permissões fornece alguma proteção com relação a alguns riscos de serialização.

Listagem de permissões em classes do WebSphere MQ para JMS.

Com APAR IT14385 e IBM WebSphere MQ Version 7.5.0, Fix Pack 8, WebSphere MQ classes para JMS suporta a lista de permissões de classes na implementação da interface JMS ObjectMessage . A lista de permissões define quais classes do Java podem ser serializadas com ObjectMessage.setObject() e desserializadas com ObjectMessage.getObject().

As tentativas de serializar ou desserializar uma instância de uma classe não incluída na lista de permissões com o ObjectMessage fazem com que um javax.jms.MessageFormatException seja lançado, com um java.io.InvalidClassException como sua causa.

Produzindo a lista de permissões

Importante: WebSphere MQ classes para JMS não podem ser distribuídas com uma lista de permissões. A opção de classes a serem transferidas usando ObjectMessages é uma opção de design do aplicativo e o IBM WebSphere MQ não pode priorizar isso.

Por essa razão, o mecanismo de listagem de permissões permite dois modos de operação:

DISCOVERY
Nesse modo, o mecanismo produz uma listagem de nomes completos de classe, relatando todas as classes que foram observadas para serem serializadas ou desserializadas em ObjectMessages.

ENFORCEMENT
Nesse modo, o mecanismo impõe a listagem de permissões, rejeitando as tentativas de serializar ou desserializar as classes que não estão na lista de permissões.

Para usar esse mecanismo, a execução deverá ser feita inicialmente no modo de DESCOBERTA para reunir a lista de classes atualmente serializadas e desserializadas, revisá-la e usá-la como base para a sua lista de permissões. Pode até ser apropriado usar a lista inalterada, mas a lista deve ser revisada primeiro, antes de decidir fazer isso.

Controlando o mecanismo de listagem de permissões

Três propriedades do sistema estão disponíveis para controlar o mecanismo de listagem de permissões:

com.ibm.mq.jms.whitelist
Essa propriedade pode ser especificada de uma das seguintes maneiras:

- O nome do caminho do arquivo que contém a lista de permissões, no formato de URI de arquivo (ou seja, começando com `file:`). No modo de DESCOBERTA, esse arquivo é gravado pelo mecanismo de listagem de permissões. O arquivo não deve existir. Se o arquivo existir, o mecanismo lançará uma exceção, em vez de sobrescrevê-lo. No modo de CUMPRIMENTO, esse arquivo é lido pelo mecanismo de listagem de permissões.
- Uma lista separada por vírgula de nomes completos de classe que constituem a lista de permissões.

Se essa propriedade estiver desconfigurada, o mecanismo da lista de permissões estará inativo.

Se você estiver usando um gerenciador de segurança Java , deverá assegurar que as classes WebSphere MQ para arquivos JAR JMS tenham acesso de leitura e gravação a esse arquivo.

com.ibm.mq.jms.whitelist.discover

- Se essa propriedade for desconfigurada ou configurada como false, o mecanismo da lista de permissões será executado no modo de CUMPRIMENTO.
- Se essa propriedade for configurada como true e a lista de permissões tiver sido especificada como um URI de arquivo, o mecanismo da lista de permissões será executado no modo de DESCOBERTA.

- Se essa propriedade for configurada como true e a lista de permissões tiver sido especificada como uma lista de nomes de classes, o mecanismo da lista de permissões lançará uma exceção adequada.
- Se essa propriedade for configurada como true e a lista de permissões não tiver sido especificada usando a propriedade com.ibm.mq.jms.whitelist, o mecanismo da lista de permissões estará inativo.
- Se essa propriedade for configurada como true e o arquivo da lista de permissões já existir, o mecanismo da lista de permissões lançará uma java.io.InvalidClassException e as entradas não serão incluídas no arquivo.

com.ibm.mq.jms.whitelist.mode
Essa propriedade de sequência pode ser especificada em qualquer uma de três maneiras:

- Se essa propriedade for configurada como SERIALIZE, o modo de CUMPRIMENTO realizará a validação da lista de permissões apenas no método ObjectMessage.setObject().
- Se essa propriedade for configurada como DESERIALIZE, o modo de CUMPRIMENTO realizará a validação da lista de permissões apenas no método ObjectMessage.getObject().
- Se essa propriedade for desconfigurada ou configurada como qualquer outro valor, o modo de CUMPRIMENTO executará a validação da lista de permissões em ambos os métodos ObjectMessage.getObject() e ObjectMessage.setObject().

Formato do arquivo da lista de permissões

Estes são os principais recursos do formato do arquivo da lista de permissões:

- O arquivo da lista de permissões está em codificação de arquivo de plataforma padrão com finais de linha apropriadas para plataforma.

  Nota: O arquivo poderá precisar conversão se você o mover entre sistemas heterogêneos.
- Cada linha não vazia contém um nome completo de classe. As linhas vazias são ignoradas.
- Comentários podem ser incluídos - qualquer coisa após um caractere '#' até o final da linha é ignorada.
- Há um mecanismo curinga muito básico:
  - '*' pode ser o **último** elemento de um nome de classe.
  - '*' corresponde a um **único** elemento de um nome de classe, ou seja, a classe, mas nenhuma parte do pacote.

  Por isso, `com.ibm.mq.*` corresponderia a `com.ibm.mq.MQMessage`, mas não a `com.ibm.mq.jmqi.remote.api.RemoteFAP`.

  O curinga não funciona para classes no pacote padrão, ou seja, para classes sem um nome de pacote explícito, por isso, um nome de classe de "*" é rejeitado.
- Arquivos da lista de permissões mal formatados, por exemplo, arquivos que contêm uma entrada como `com.ibm.mq.*.Message`, em que o curinga não é o último elemento, resultam em uma java.lang.IllegalArgumentException sendo lançada.
- Um arquivo da lista de permissões vazio tem o efeito de desativar totalmente o uso do ObjectMessage.

Formato da lista de permissões como uma lista separada por vírgula

O mesmo mecanismo de curinga está disponível para uma lista de permissões como uma lista separada por vírgula.

- O '*' poderá ser expandido pelo sistema operacional se especificado em uma linha de comandos, shell script ou arquivo em lote, por isso, ele poderá precisar de manipulação especial.
- O caractere de comentário '#' é aplicável apenas quando um arquivo é especificado. Se a lista de permissões for especificada como uma lista separada por vírgula de nomes de classes, supondo que o sistema operacional ou shell não o processe, pois ele é o caractere de comentário padrão em muitos shells UNIX ou Linux , ele será tratado como um caractere normal.

Quando a listagem de permissões acontece?

A listagem de permissões é iniciada quando o aplicativo executa primeiro um método ObjectMessage setMessage() ou getMessage().

As propriedades do sistema são avaliadas, o arquivo da lista de permissões é aberto e no modo de CUMPRIMENTO, a lista de classes da lista de permissões é carregada quando o mecanismo é inicializado. Neste ponto, uma entrada é gravada no arquivo de registro JMS do IBM WebSphere MQ para o aplicativo..

Quando o mecanismo é inicializado, seus parâmetros podem não ser mudados. O horário da inicialização não é facilmente previsto, pois ele depende do comportamento do aplicativo. As configurações de propriedade do sistema e os conteúdos do arquivo da lista de permissões devem, portanto, ser considerados como fixos a partir do momento em que o aplicativo é iniciado. Não mude as propriedades ou os conteúdos do arquivo da lista de permissões enquanto o aplicativo estiver em execução, pois os resultados não são garantidos.

Pontos a serem considerados

A melhor abordagem para minimizar os riscos intrínsecos ao mecanismo de serialização do Java seria explorar abordagens alternativas para transferência de dados, como usar JSON, em vez de ObjectMessage. Usar os mecanismos do IBM WebSphere MQ Advanced Message Security (AMS) pode incluir segurança adicional, assegurando que as mensagens venham de fontes confiáveis.

Se você usar o mecanismo do gerenciador de segurança do Java com seu aplicativo, deverá conceder as permissões a seguir:

- FilePermission em qualquer arquivo da lista de permissões que você usa, com permissão de leitura para o modo de CUMPRIMENTO, permissão de gravação para o modo de DESCOBERTA.
- PropertyPermission (leitura) nas propriedades com.ibm.mq.jms.whitelist, com.ibm.mq.jms.whitelist.discover e com.ibm.mq.jms.whitelist.mode.

Conceitos relacionados

“Executando classes WebSphere MQ para aplicativos JMS sob o gerenciador de segurança Java” na página 738

As classes WebSphere MQ para JMS podem ser executadas com o gerenciador de segurança Java ativado. Para executar aplicativos com sucesso com o gerenciador de segurança ativado, deve-se configurar a máquina virtual Java (JVM) com um arquivo de configuração de política adequado.

O adaptador de recursos IBM WebSphere MQ

O adaptador de recursos permite que aplicativos em execução em um servidor de aplicativos acessem recursos do IBM WebSphere MQ . Ele suporta a comunicação de entrada e de saída.

O Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) fornece uma maneira padrão de conectar aplicativos em execução em um ambiente do Java EE a um Enterprise Information System (EIS), como IBM WebSphere MQ ou Db2. O adaptador de recursos IBM WebSphere MQ implementa as interfaces JCA 1.5 e contém o IBM WebSphere MQ classes for JMS. Ele permite que aplicativos JMS e beans acionados por mensagem (MDBs) em execução em um servidor de aplicativos, acessem os recursos de um gerenciador de filas do IBM WebSphere MQ . O adaptador de recursos suporta tanto o domínio ponto a ponto e o domínio de publicar/assinar.

O adaptador de recursos do IBM WebSphere MQ suporta dois tipos de comunicação entre um aplicativo e um gerenciador de filas:

Comunicação de saída
: Um aplicativo inicia uma conexão com um gerenciador de filas e, em seguida, envia mensagens JMS para destinos JMS e recebe mensagens JMS de destinos JMS de uma maneira síncrona

Comunicação de entrada
: Uma mensagem JMS que chega a um destino JMS é entregue a um MDB, que processa a mensagem de forma assíncrona

Para obter informações adicionais sobre a IBM WebSphere MQ classes for JMS, consulte “Usando classes do WebSphere MQ para JMS” na página 724.

O adaptador de recursos também contém o IBM WebSphere MQ classes for Java As classes estão automaticamente disponíveis para aplicativos em execução em um servidor de aplicativos no qual o adaptador de recursos foi implementado e permitem que aplicativos em execução nesse servidor de aplicativos usem a API do IBM WebSphere MQ classes for Java ao acessar recursos de um gerenciador de filas do IBM WebSphere MQ . Para obter informações adicionais sobre oIBM WebSphere MQ classes for Java, consulte “Usando classes do WebSphere MQ para Java” na página 660.

O uso do IBM WebSphere MQ classes for Java em um ambiente Java EE é suportado com restrições. Para obter informações sobre essas restrições, consulte “Executando classes IBM WebSphere MQ para aplicativos Java dentro da Java plataforma Enterprise Edition” na página 722.

Outra documentação necessária para suportar um adaptador de recursos JCA

Consulte a documentação para seu servidor de aplicativos para obter informações sobre como configurar um adaptador de recursos JCA

Cada servidor de aplicativos fornece seu próprio conjunto de interfaces de administração. Alguns servidores de aplicativos fornecem interfaces gráficas com o usuário para definir recursos de JCA, mas outros requerem que o administrador escreva planos de implementação de XML. Portanto, está além do escopo desta documentação fornecer informações sobre como configurar o adaptador de recursos do WebSphere MQ para cada servidor de aplicativos. Esta documentação se concentra apenas no que você precisa configurar Consulte a documentação fornecida com o seu servidor de aplicativos para obter informações sobre como configurar um adaptador de recursos JCA.

Para entender essa documentação, você deve estar familiarizado com as classes JMS e WebSphere MQ para JMS. Muitas das propriedades usadas para configurar o adaptador de recursos WebSphere MQ são equivalentes às propriedades das classes WebSphere MQ para objetos JMS e têm a mesma função.

Instalação do adaptador de recursos do WebSphere MQ

O adaptador de recursos do WebSphere MQ é fornecido como um arquivo RAR (Resource Archive). Instale o arquivo RAR em seu servidor de aplicativos. Pode ser necessário incluir diretórios no caminho do sistema.

O adaptador de recursos WebSphere MQ é fornecido como um arquivo de archive de recursos (RAR) chamado wmq.jmsra.rar. Esse arquivo é instalado com classes WebSphere MQ para JMS no diretório mostrado em Tabela 94 na página 743.

Tabela 94. O diretório contendo wmq.jmsra.rar para cada plataforma

Plataforma	**Diretório**
AIX, HP-UX, Linux e Solaris	*MQ_INSTALLATION_PATH* /java/lib/jca
IBM i	/QIBM/ProdData/mqm/java/lib/jca
Windows	*MQ_INSTALLATION_PATH* \java\lib\jca
O *MQ_INSTALLATION_PATH* representa o diretório de alto nível no qual o WebSphere MQ está instalado.	

O arquivo RAR contém classes WebSphere MQ para JMS e a implementação WebSphere MQ das interfaces JCA.

Você deve instalar o arquivo RAR do adaptador de recursos do WebSphere MQ em seu servidor de aplicativos, mas a maneira de fazer isso depende do servidor de aplicativos. Consulte a documentação para seu servidor de aplicativos para obter informações sobre como instalar um arquivo RAR do adaptador de recursos.

Para conexões de ligações em sistemas UNIX and Linux , você deve assegurar que o diretório contendo as bibliotecas Java Native Interface (JNI) esteja no caminho do sistema. Para o local desse diretório, que também contém as classes WebSphere MQ para bibliotecas JMS, consulte Tabela 93 na página 734. No

Windows, esse diretório é automaticamente incluído no caminho do sistema durante a instalação das classes do WebSphere MQ para JMS

Transações são suportadas no modo cliente e no modo ligações.

O adaptador de recursos do WebSphere MQ e a versão das classes do WebSphere MQ para JMS usadas pelo adaptador de recursos devem estar no mesmo nível de release.

WebSphere Servidor de Aplicativos e o adaptador de recursos WebSphere MQ
Não use o adaptador de recursos do WebSphere MQ com o WebSphere Application Server Versão 6 O WebSphere Application Server, V7, inclui uma versão do WebSphere MQ V7 Resource Adapter.

Não utilize o adaptador de recursos do WebSphere MQ no WebSphere Application Server, V6. Para acessar os recursos de um gerenciador de filas do WebSphere MQ a partir do WebSphere Application Server a partir de um aplicativo JMS, use o provedor de sistemas de mensagens do WebSphere MQ . O provedor de sistemas de mensagens WebSphere MQ contém uma versão do WebSphere MQ classes para JMS.

O WebSphere Application Server, V7, inclui uma versão do WebSphere MQ V7 Resource Adapter.

Para obter mais informações, consulte a nota técnica Qual versão do WebSphere MQ Resource Adapter (RA) é fornecida com o WebSphere Application Server?.

WebSphere Application Server Liberty e o adaptador de recurso IBM WebSphere MQ
O adaptador de recursos IBM WebSphere MQ Version 7.5 pode ser instalado no WebSphere Application Server Liberty Versão 8.5.5, Fix Pack 2 ou posterior, usando o recurso wmqJmsClient-1.1 . Como alternativa, é possível, sujeito a algumas restrições, instalar o adaptador de recurso usando o suporte genérico Java Platform, Enterprise Edition Connector Architecture (Java EE JCA).

Restrições gerais ao instalar o adaptador de recursos no Liberty

As restrições a seguir se aplicam ao adaptador de recursos do Version 7.5 ao usar o recurso wmqJmsClient-1.1 e também ao usar o suporte JCA genérico:

- Os IBM WebSphere MQ classes for Java não são suportados no Liberty Eles não devem ser usados com o recurso de sistema de mensagens do Liberty do IBM WebSphere MQ nem com o suporte genérico do JCA Para obter mais informações, consulte Usando as interfaces Java do WebSphere MQ em ambientes J2EE/JEE.
- O adaptador de recursos do IBM WebSphere MQ tem um tipo de transporte de BINDINGS_THEN_CLIENT. Esse tipo de transporte não é suportado no recurso de sistema de mensagens do IBM WebSphere MQ Liberty.
- O recurso IBM WebSphere MQ Advanced Message Security (IBM WebSphere MQ AMS) não está incluído no recurso de sistema de mensagens do IBM WebSphere MQ Liberty.

O adaptador de recursos do IBM WebSphere MQ Version 7.5 não pode ser usado com o recurso wmqJmsClient-2.0

Configuração do adaptador de recursos do WebSphere MQ

Para configurar o adaptador de recursos WebSphere MQ , defina vários recursos JCA e propriedades do sistema.

Defina recursos JCA nas seguintes categorias:

- As propriedades do objeto ResourceAdapter , que representam as propriedades globais do adaptador de recursos, como o nível de rastreio de diagnóstico.. Essas propriedades estão descritas em “Configuração do Objeto ResourceAdapter” na página 745.
- As propriedades de um objeto ActivationSpec que determinam como um MDB é ativado para comunicação de entrada. Essas propriedades estão descritas em “Configurando o adaptador de recursos para comunicação de entrada” na página 747.

- As propriedades de um objeto ConnectionFactory , que o servidor de aplicativos usa para criar um objeto JMS ConnectionFactory para comunicação de saída. Essas propriedades estão descritas em “Configurando o adaptador de recursos para comunicação de saída” na página 761.
- As propriedades de um objeto de destino administrado, que o servidor de aplicativos usa para criar um objeto de Fila JMS ou um objeto de Tópico JMS para comunicação de saída Essas propriedades também são descritas em “Configurando o adaptador de recursos para comunicação de saída” na página 761

O arquivo RAR do adaptador de recursos do WebSphere MQ contém um arquivo chamado META-INF/ra.xml, que contém um descritor de implementação para o adaptador de recursos Esse descritor de implementação é definido pelo esquema XML no https://java.sun.com/xml/ns/j2ee/connector_1_5.xsd e contém informações sobre o adaptador de recursos e os serviços que ele fornece. Um servidor de aplicativos também pode requerer um plano de implementação para o adaptador de recursos. Este plano de implementação é específico ao servidor de aplicativos. Por exemplo, WebSphere Application Server Community Edition requer um plano de implementação chamado geronimo-ra.xml.

Se você estiver usando SSL (Secure Sockets Layer), especifique os locais do arquivo keystore e o arquivo de armazenamento confiável como propriedades de sistemas JVM, como no exemplo a seguir:

```
java ... -Djavax.net.ssl.keyStore=
key_store_location
         -Djavax.net.ssl.trustStore=trust_store_location
         -Djavax.net.ssl.keyStorePassword=key_store_password
```

Essas propriedades não podem ser propriedades de um objeto ActivationSpec ou ConnectionFactory e não é possível especificar mais de um keystore para um servidor de aplicativos. As propriedades se aplicam a todo JVM e pode, portanto, afetar o servidor de aplicativos se outros aplicativos em execução no servidor de aplicativos estiverem usando conexões SSL. O servidor de aplicativos pode também reconfigurar essas propriedades para valores diferentes. Para obter mais informações sobre como usar SSL com classes WebSphere MQ para JMS, consulte “Usando Secure Sockets Layer (SSL) com classes WebSphere MQ para JMS” na página 917.

Um programa de teste de verificação de instalação (IVT) é fornecido com o adaptador de recursos WebSphere MQ , mas você deve configurar o adaptador de recursos antes de poder executar o programa. Para obter informações sobre o que é preciso configurar para executar o programa IVT, consulte “O programa de teste de verificação de instalação para o adaptador de recursos WebSphere MQ” na página 792.

Os logs do adaptador de recursos, mensagens de aviso e erro usam o mesmo mecanismo que as classes IBM WebSphere MQ para JMS, para obter detalhes, consulte “Criação de log eIBM WebSphere MQ classes for JMS” na página 806. Para o WebSphere Application Server, essas mensagens são automaticamente redirecionadas para o log de saída do servidor de aplicativos Para outros servidores de aplicativos, como WAS CE e JBoss, eles irão, por padrão, acessar um arquivo chamado mqjms.log. Para configurar o adaptador de recursos para registrar adicionalmente mensagens de aviso no log de saída padrão dos seus servidores de aplicativos, configure a propriedade de sistema JVM a seguir para seu servidor de aplicativos:

```
-Dcom.ibm.msg.client.commonservices.log.outputName=mqjms.log,stdout
```

Para obter detalhes sobre como configurar uma propriedade de sistema da JVM, consulte a documentação do servidor de aplicativos

Configuração do Objeto ResourceAdapter
O objeto ResourceAdapter contém as propriedades globais do adaptador de recursos do WebSphere MQ . Defina essas propriedades usando as instalações do adaptador de recursos.

O objeto ResourceAdapter possui dois conjuntos de propriedades:

- Propriedades associadas ao rastreio de diagnóstico
- Propriedades associadas ao conjunto de conexões gerenciado pelo adaptador de recursos

A maneira como você define essas propriedades depende das interfaces de administração fornecidas por seu servidor de aplicativos.

Para obter mais informações sobre como definir as propriedades associadas ao rastreio de diagnóstico, consulte Rastreando o IBM WebSphere MQ adaptador de recursos

O adaptador de recursos gerencia um conjunto de conexões internas de conexões JMS que são usadas para entregar mensagens para MDBs. Tabela 95 na página 746 lista as propriedades do objeto ResourceAdapter associadas ao conjunto de conexões.

Tabela 95. Propriedades do Objeto ResourceAdapter que estão associadas ao conjunto de conexões

Nome da propriedade	**Tipo**	**Valor padrão**	**Descrição**
maxConnections	Sequência	50	O número máximo de conexões com um gerenciador de filas do WebSphere MQ e o número máximo de MDBs implementados
connectionConcurrency	Sequência	1	O número máximo de MDBs para compartilhar uma conexão JMS. O compartilhamento de conexões não é possível e essa propriedade sempre tem o valor 1.
reconnectionRetryCount	Sequência	5	O número máximo de tentativas feitas pelo adaptador de recursos para reconectar a um gerenciador de filas do WebSphere MQ se uma conexão falhar.
reconnectionRetryInterval	Sequência	300 000	O tempo, em milissegundos, que o adaptador de recursos aguarda antes de tentar reconectar-se a um gerenciador de filas do WebSphere MQ
startupRetryCount	Sequência	0	O número padrão de vezes para tentar e conectar um MDB na inicialização, se o gerenciador de filas não estiver em execução quando o servidor de aplicativos é iniciado.
startupRetryInterval	Sequência	30.000	O tempo de suspensão padrão entre as tentativas de conexão de inicialização (em milissegundos).

Quando um MDB é implementado no servidor de aplicativos, uma nova conexão JMS é criada e uma conversa iniciada com o gerenciador de filas, desde que o número máximo de conexões especificado pela propriedade maxConnection não seja excedido O número máximo de MDBs é igual a, portanto, o número máximo de conexões. Se o número de MDBs implementados atingir esse máximo, qualquer tentativa de implementar outro MDB falhará. Se um MDB for interrompido, sua conexão poderá ser usada por outro MDB.

Em geral, se vários MDBs tiverem que ser implementados, o valor da propriedade maxConnections deverá ser aumentado.

As propriedades do intervalo reconnectionRetrye reconnectionRetrycontrolam o comportamento do adaptador de recursos quando as conexões com um gerenciador de filas do WebSphere MQ falham, devido a uma falha de rede, por exemplo. Quando uma conexão falha, o adaptador de recursos suspende a entrega de mensagens para todos os MDBs fornecidos por essa conexão por um intervalo especificado pela propriedade reconnectionRetryInterval. O adaptador de recursos, então, tenta se reconectar ao gerenciador de filas. Se a tentativa falhar, o adaptador de recursos fará outras tentativas de reconexão nos intervalos especificados pela propriedade reconnectionRetryInterval até o limite imposto pela propriedade reconnectionRetryCount ser atingido. Se todas as tentativas falharem, a entrega será interrompida permanentemente até os MDBs serem reiniciados manualmente.

Em geral, o objeto ResourceAdapter não requer administração. No entanto, para ativar o rastreio de diagnóstico em sistemas UNIX and Linux, por exemplo, é possível definir as seguintes propriedades:

```
traceEnabled:       true
traceLevel:         10
```

Essas propriedades não terão efeito se o adaptador de recursos não tiver sido iniciado, que é o caso, por exemplo, quando aplicativos que usam recursos do WebSphere MQ estiverem em execução apenas no contêiner do cliente. Nessa situação, é possível configurar as propriedades para rastreio de diagnóstico como propriedades do sistema da Java Virtual Machine (JVM) É possível configurar as propriedades usando o sinalizador -D no comando **java**, como no seguinte exemplo:

```
java ... -DtraceEnabled=true -DtraceLevel=6
```

Não é necessário definir todas as propriedades do objeto ResourceAdapter. Quaisquer propriedades deixadas sem especificação usam seus valores padrão. Em um ambiente gerenciado, é melhor não misturar as duas maneiras de especificar propriedades. Se você misturá-las, as propriedades do sistema JVM terão precedência sobre as propriedades do objeto ResourceAdapter.

Configurando o adaptador de recursos para comunicação de entrada
Para configurar a comunicação de entrada, defina as propriedades de um ou mais objetos ActivationSpec.

As propriedades de um objeto ActivationSpec determinam como um bean da unidade de mensagens (MDB) recebe mensagens JMS de uma fila do WebSphere MQ . O comportamento transacional do MDB é definido em seu descritor de implementação.

Um objeto ActivationSpec possui dois conjuntos de propriedades:

- Propriedades que são usadas para criar uma conexão JMS com um gerenciador de filas do WebSphere MQ
- Propriedades que são usadas para criar um consumidor de conexão JMS que entrega mensagens de forma assíncrona conforme elas chegam em uma fila especificada

A maneira na qual você define as propriedades de um objeto ActivationSpec depende das interfaces de administração fornecidas por seu servidor de aplicativos.

Tabela 96 na página 747 lista as propriedades de um objeto ActivationSpec que são usadas para criar uma conexão JMS para um gerenciador de fila do WebSphere MQ

Tabela 96. Propriedades de um objeto ActivationSpec usado para criar uma conexão JMS

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
applicationName	Sequência	• O nome da classe de chamada, se estiver disponível, ajustado para não ter mais de 28 caracteres. Se ela não estiver disponível, a sequência `WebSphere MQ Client for Java` será usada.	O nome pelo qual um aplicativo é registrado com o gerenciador de filas. Esse nome do aplicativo é mostrado pelo comando **DISPLAY CONN MQSC/PCF** (em que o campo é chamado **APPLTAG**) ou na exibição IBM WebSphere MQ Explorer **Conexões de Aplicativos** (em que o campo é chamado **App name**).
brokerCCDurSubQueue [1]	Sequência	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Um nome da fila	O nome da fila a partir da qual um consumidor de conexão recebe mensagens de assinaturas duráveis

Tabela 96. Propriedades de um objeto ActivationSpec usado para criar uma conexão JMS (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
brokerCCSubQueue [1]	Sequência	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Um nome da fila	O nome da fila a partir da qual um consumidor de conexão recebe mensagens de assinaturas não duráveis
brokerControlQueue [1]	Sequência	• **SYSTEM.BROKER.CONTROL.QUEUE** • Um nome da fila	O nome da fila de controle do broker
brokerQueueManager [1]	Sequência	• **"" (empty string)** • Um nome do gerenciador de filas	O nome do gerenciador de filas no qual o broker está em execução
brokerSubQueue[1]	Sequência	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Um nome da fila	O nome da fila a partir da qual um consumidor de mensagens não duráveis recebe as mensagens
brokerVersion[1]	Sequência	• **unspecified** – Quando o broker for migrado da V6 para V7, configure essa propriedade de modo que os cabeçalhos RFH2 não sejam mais usados. Após a migração, essa propriedade não é mais relevante. • **V1** -Para usar um broker de publicação / assinatura WebSphere MQ . Ou para usar um broker do WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker ou WebSphere Business Integration Message Broker no modo de compatibilidade. Este valor é o valor padrão, se TRANSPORT estiver definido como BIND ou CLIENT. • **V2** -Para usar um broker do WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker ou WebSphere Business Integration Message Broker no modo nativo Este valor é o valor padrão, se TRANSPORT estiver definido como DIRECT ou DIRECTHTTP.	A versão do broker que está sendo usada
ccdtURL	Sequência	• **null** • Um localizador uniforme de recursos (URL)	Uma URL que identifica o nome e o local do arquivo contendo a tabela de definição de canal do cliente (CCDT) e especifica como o arquivo pode ser acessado
CCSID	Sequência	• **819** • Um identificador do conjunto de caracteres codificados suportado pela Java virtual machine (JVM)	O identificador do conjunto de caracteres codificados para uma conexão

Tabela 96. Propriedades de um objeto ActivationSpec usado para criar uma conexão JMS (continuação)			
Nome da propriedade	**tipo**	**Valores válidos (valor padrão em negrito)**	**Descrição**
channel	Sequência	• **SYSTEM.DEF.SVRCONN** • O nome de um canal de MQI	O nome do canal de MQI a ser usado
cleanupInterval [1]	int	• **3 600 000** • Um inteiro positivo	O intervalo, em milissegundos, entre as execuções em segundo plano do utilitário de limpeza de publicar/assinar
cleanupLevel[1]	Sequência	• **SAFE** • Nenhum • STRONG • FORCE • NONDUR	O nível de limpeza para um armazenamento de assinatura baseado no broker
clientID	Sequência	• **null** • Um identificador de cliente	O identificador do cliente para uma conexão
cloneSupport	Sequência	• **DISABLED** – Apenas uma instância de um assinante de tópico durável pode ser executada de cada vez. • ENABLED-Duas ou mais instâncias do mesmo assinante de tópico durável podem ser executadas simultaneamente, mas cada instância deve executar em uma Java virtual machine (JVM) separada.	Se duas ou mais instâncias do mesmo assinante de tópico durável puderem ser executadas simultaneamente
connectionNameList	Sequência	• **localhost(1414)** • Uma sequência composta de itens separados por vírgulas, em que cada item tem o formato: `HOSTNAME(PORT)` em que *HOSTNAME* é um nome DNS ou um endereço IP.	Uma lista de nomes de conexão TCP/IP usada para comunicações de entrada. Quando especificado, **connectionNameList** substitui as propriedades **hostname** e **port** . Essa propriedade é usada para se reconectar a gerenciadores de filas de várias instâncias. **connectionNameList** é semelhante em forma a **localAddress**, mas não deve ser confundida com ela. **localAddress** especifica as características das comunicações locais, enquanto **connectionNameList** especifica como alcançar um gerenciador de filas remotas.

Tabela 96. Propriedades de um objeto ActivationSpec usado para criar uma conexão JMS (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
failIfQuiesce	Booleana	• **true** • false	Indica se as chamadas para certos métodos falharão se o gerenciador de filas estiver em um estado de quiesce
headerCompression	Sequência	• **NONE** • SYSTEM - A compactação do cabeçalho da mensagem de RLE é executada	Uma lista de técnicas que podem ser usadas para compactar os dados do cabeçalho em uma conexão
hostName	Sequência	• **localhost** • Um nome do host • Um endereço IP	O nome do host ou o endereço IP do sistema em que o gerenciador de filas reside. As propriedades **hostname** e **port** são substituídas pela propriedade **connectionNameList** quando especificada.
localAddress	Sequência	• **null** • Uma sequência no formato: `[host_name][(low_port[,high_port])]` em que *host_name* é um nome do host ou endereço IP, *low_port* e *high_port* são números de porta TCP e colchetes denotam um componente opcional	Para uma conexão com um gerenciador de filas, esta propriedade especifica um ou ambos os seguintes procedimentos: • A interface de rede local a ser usada • A porta local ou um intervalo de portas locais a ser usado **localAddress** é semelhante em forma a **connectionNameList**, mas não deve ser confundida com ela. **localAddress** especifica as características das comunicações locais, enquanto **connectionNameList** especifica como alcançar um gerenciador de filas remotas.
messageCompression	Sequência	• **NONE** • Uma lista de um ou mais dos seguintes valores separados por caracteres em branco: RLE ZLIBFAST ZLIBHIGH	Uma lista de técnicas que podem ser usadas para compactar os dados da mensagem em uma conexão

Tabela 96. Propriedades de um objeto ActivationSpec usado para criar uma conexão JMS (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
messageRetention[1]	Booleana	• **true** – Mensagens indesejadas permanecem na fila de entrada • false – As mensagens não desejadas são tratadas de acordo com suas opções de disposição	Indica se o consumidor da conexão manterá mensagens indesejadas na fila de entrada
messageSelection [1]	Sequência	• **CLIENT** • BROKER	Determina se a seleção de mensagens é feita pelas classes do WebSphere MQ para JMS ou pelo broker A seleção de mensagens pelo broker não é suportada quando brokerVersion tem o valor 1.
senha	Sequência	• **null** • Uma senha	A senha padrão a ser usada ao criar uma conexão com o gerenciador de filas
pollingInterval[1]	int	• **5000** • Qualquer número inteiro positivo	Se cada listener de mensagem dentro de uma sessão não tiver mensagens adequadas em sua fila, este valor será o intervalo máximo, em milissegundos, que decorrerá antes que cada listener da mensagem tente novamente obter uma mensagem de sua fila. Se ocorrer com frequência o fato de nenhuma mensagem adequada estar disponível para qualquer um dos listeners da mensagem em uma sessão, considere aumentar o valor desta propriedade. Esta propriedade é relevante apenas se TRANSPORT tiver o valor BIND ou CLIENT.
port	int	• **1414** • Um número da porta TCP	A porta na qual o gerenciador de filas atende. As propriedades **hostname** e **port** são substituídas pela propriedade **connectionNameList** quando especificada.

Tabela 96. Propriedades de um objeto ActivationSpec usado para criar uma conexão JMS (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
providerVersion	sequência	• **unspecified** • Uma sequência em um dos formatos a seguir – V.R.M.F – V.R.M – V.R – V em que V, R, M e F são valores de números inteiros maiores ou iguais a zero.	A versão, liberação, nível de modificação e fix pack do gerenciador de filas ao qual o MDB pretende se conectar.
queueManager	Sequência	• **"" (empty string)** • Um nome do gerenciador de filas	O nome do gerenciador de filas ao qual se conectar
receiveExit [3]	Sequência	• **null** • Uma sequência composta por um ou mais itens separados por vírgulas, em que cada item é o nome completo de uma classe que implementa as classes WebSphere MQ para interface Java, MQReceiveExit	Identifica um programa de saída de recebimento de canal ou uma sequência de programas de saída de recebimento a ser executada na sucessão
receiveExitInit	Sequência	• **null** • Uma sequência composta de um ou mais itens de dados do usuário separados por vírgulas	Os dados do usuário que são transmitidos para programas de saída de recebimento do canal quando são chamados

Tabela 96. Propriedades de um objeto ActivationSpec usado para criar uma conexão JMS (continuação)			
Nome da propriedade	**tipo**	**Valores válidos (valor padrão em negrito)**	**Descrição**
rescanInterval[1]	int	• **5000** • Qualquer número inteiro positivo	Quando um consumidor de mensagens no domínio ponto a ponto usa um seletor de mensagem para selecionar quais mensagens ele deseja receber, as classes do WebSphere MQ para JMS procuram na fila do WebSphere MQ mensagens adequadas na sequência determinada pelo atributo *MsgDeliverySequence* da fila. Quando as classes do WebSphere MQ para JMS localizam uma mensagem adequada e a entregam ao consumidor, as classes do WebSphere MQ para JMS retoma a procura para a próxima mensagem adequada a partir de sua posição atual na fila. WebSphere MQ classes para JMS continua a procurar a fila desta maneira até atingir o final da fila ou até o intervalo de tempo em milissegundos, conforme determinado pelo valor desta propriedade, ter expirado. Em cada caso, as classes WebSphere MQ para JMS retornam ao início da fila para continuar sua procura e um novo intervalo de tempo começa.
securityExit [3]	Sequência	• **null** • O nome completo de uma classe que implementa as classes WebSphere MQ para a interface Java, MQSecurityExit	Identifica um programa de saída de segurança do canal
securityExitInit	Sequência	• **null** • Uma sequência de dados do usuário	Os dados do usuário que são transmitidos para um programa de saída de segurança do canal quando são chamados

Tabela 96. Propriedades de um objeto ActivationSpec usado para criar uma conexão JMS (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
sendExit [3]	Sequência	• **null** • Uma sequência composta por um ou mais itens separados por vírgulas, em que cada item é o nome completo de uma classe que implementa as classes WebSphere MQ para interface Java, MQSendExit	Identifica um programa de saída de envio de canal ou uma sequência de programas de saída de envio a ser executada na sucessão
sendExitInit	Sequência	• **null** • Uma sequência composta de um ou mais itens de dados do usuário separados por vírgulas	Os dados do usuário que são transmitidos para programas de saída de envio do canal quando são chamados
shareConvAllowed	Booleana	• NO-Uma conexão do cliente não pode compartilhar seu soquete.. • **YES** -Uma conexão do cliente pode compartilhar seu soquete	Se uma conexão do cliente pode compartilhar seu soquete com outras conexões JMS de nível superior a partir do mesmo processo para o mesmo gerenciador de filas, se as definições de canal corresponderem
sparseSubscriptions[1]	Booleana	• **false** – As assinaturas recebem mensagens de correspondência frequentes. • true – As assinaturas recebem mensagens de correspondência não frequentes. Esse valor requer que a fila de assinaturas possa ser aberta para procurar.	Controla a política de recuperação de mensagens de um objeto TopicSubscriber
sslCertStores	Sequência	• **null** • Uma sequência de uma ou mais URLs de LDAP separadas por espaços em branco. Cada URL de LDAP possui o formato: `ldap://host_name[:port]` em que *host_name* é um nome de host ou endereço IP, *port* é um número de porta TCP e colchetes denotam um componente opcional.	Os servidores Lightweight Directory Access Protocol (LDAP) que retêm as listas de revogação de certificado (CRLs) para uso em uma conexão SSL
sslCipherSuite	Sequência	• **null** • O nome de um CipherSuite	O CipherSuite a ser usado para uma conexão SSL
sslFipsRequired [2]	Booleana	• **falso** • true	Se uma conexão SSL deve usar um CipherSuite que seja suportado pelo provedor IBM Java JSSE FIPS (IBMJSSEFIPS)

Tabela 96. Propriedades de um objeto ActivationSpec usado para criar uma conexão JMS (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
sslPeerName	Sequência	• **null** • Um modelo para nomes distintos	Para uma conexão SSL, um modelo que é usado para verificar o nome distinto no certificado digital fornecido pelo gerenciador de filas
sslResetCount	int	• **0** • Um número inteiro no intervalo de 0 – 999999999	O número total de bytes enviados e recebidos por uma conexão SSL antes de as chaves secretas usadas pelo SSL serem renegociadas
sslSocketFactory	Sequência	Uma sequência que representa o nome de classe completo de uma classe que fornece uma implementação da interface javax.net.ssl.SSLSocketFactory. Opcionalmente, incluindo um argumento a ser transmitido para o método construtor, entre parênteses.	Quaisquer conexões estabelecidas no escopo do objeto administrado usam soquetes obtidos a partir desta implementação da interface SSLSocketFactory.
statusRefreshInterval[1]	int	• **60000** • Qualquer número inteiro positivo	O intervalo, em milissegundos, entre atualizações da transação de execução longa, que detecta quando um assinante perde sua conexão com o gerenciador de filas. Esta propriedade é relevante apenas se subscriptionStore tiver o valor QUEUE.
subscriptionStore [1]	Sequência	• **Broker** • MIGRATE • FILA	Determina onde as classes do WebSphere MQ para JMS armazenem dados persistentes sobre assinaturas ativas
transportType	Sequência	• **CLIENT** • BINDINGS • BINDINGS_THEN_CLIENT	Indica se uma conexão com um gerenciador de filas usa o modo cliente ou o modo de ligações. Se o valor BINDINGS_THEN_CLIENT for especificado, o adaptador de recursos primeiro tenta fazer uma conexão no modo de ligações. Se essa tentativa de conexão falhar, o adaptador de recursos tentará estabelecer uma conexão no modo cliente.

Tabela 96. Propriedades de um objeto ActivationSpec usado para criar uma conexão JMS (continuação)			
Nome da propriedade	**tipo**	**Valores válidos (valor padrão em negrito)**	**Descrição**
nome do usuário	Sequência	• **null** • Um nome de usuário	O nome do usuário padrão a ser usado ao criar uma conexão com um gerenciador de filas
wildcardFormat	Sequência	• CHAR - Reconhece somente os caracteres curingas, conforme usados na versão 1 do broker • **TOPIC** – Reconhece somente curingas em nível do tópico, conforme usados na versão 2 do broker	Qual versão de sintaxe curinga deve ser usada

Notas:

1. Essa propriedade pode ser usada com a Versão 7.0 de classes WebSphere MQ para JMS Ela não afeta um aplicativo conectado a um gerenciador de filas da Versão 7.0 , a menos que a propriedade providerVersion seja configurada para um número da versão menor que 7
2. Para obter informações importantes sobre o uso da propriedade sslFipsRequired, consulte “Limitações do adaptador de recursos IBM WebSphere MQ” na página 781.
3. Para obter informações sobre como configurar o adaptador de recursos para que ele possa localizar uma saída, veja “Configurando o IBM WebSphere MQ classes for JMS para usar saídas de canal” na página 924.

Tabela 97 na página 756 lista as propriedades de um objeto ActivationSpec usado para criar um consumidor de conexão JMS.

Tabela 97. Propriedades de um objeto ActivationSpec usado para criar um consumidor de conexão JMS			
Nome da propriedade	**tipo**	**Valores válidos (valor padrão em negrito)**	**Descrição**
destino	Sequência	Um nome do destino	O destino a partir do qual receber mensagens. A propriedade useJNDI determina como o valor dessa propriedade é interpretado.
destinationType	Sequência	• javax.jms.Queue • javax.jms.Topic	O tipo de destino, uma fila ou um tópico
maxMessages	int	• **1** • Um inteiro positivo	O número máximo de mensagens que podem ser designadas a uma sessão do servidor de uma vez. Se a especificação de ativação estiver entregando mensagens para um MDB em uma transação XA, um valor de 1 será usado independentemente da configuração dessa propriedade.
maxPoolDepth	int	• **10** • Um inteiro positivo	O número máximo de sessões do servidor no conjunto de sessões do servidor usado pelo consumidor de conexão

Tabela 97. Propriedades de um objeto ActivationSpec usado para criar um consumidor de conexão JMS (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
messageSelector	Sequência	• **null** • Uma expressão do seletor de mensagem SQL92	Uma expressão do seletor de mensagem especificando quais mensagens devem ser entregues
nonASFTimeout	int	• **0** • Um inteiro positivo	Um valor positivo indica que a entrega de não ASF é usada. O valor é o tempo, em milissegundos, que uma solicitação get aguarda por mensagens que podem ainda não ter chegado (um get com chamada de espera). O valor padrão, 0, indica que a entrega ASF é usada. Esse parâmetro é válido apenas quando o aplicativo está em execução no WebSphere Application Server versão 7 ou posterior
nonASFRollbackEnabled	Booleana	• **false** – A mensagem é consumida mesmo se o MDB falhar • true – A falha no MDB faz com que a mensagem retroceda para a fila.	Se a entrega de mensagem está dentro de um ponto de sincronização do WebSphere MQ se o MDB não for transacionado Ignorado se o MDB for transacionado ou se nonASFTimeout for configurado como 0.
poolTimeout	int	• **300000** • Um inteiro positivo	O tempo, em milissegundos, que uma sessão de servidor não usada é mantida aberta no conjunto de sessões do servidor antes de ser fechada devido à inatividade
readAheadAllowed	int	• **DESTINATION** – Determine se a leitura antecipada será permitida ao consultar a definição da fila ou do tópico. • DISABLED - A leitura antecipada não é permitida. • ENABLED – A leitura antecipada é permitida. • QUEUE – Determine se a leitura antecipada é permitida, fazendo referência à definição de fila. • TOPIC – Determine se a leitura antecipada é permitida, fazendo referência à definição de tópico.	Indica se o MDB tem permissão para usar a leitura antecipada para obter as mensagens não persistentes do destino em um buffer interno antes de recebê-las

Tabela 97. Propriedades de um objeto ActivationSpec usado para criar um consumidor de conexão JMS (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
readAheadClosePolicy	int	• **ALL** – Todas as mensagens no buffer interno de leitura antecipada são entregues para o MDB antes de parar. • CURRENT – Somente a chamada do MDB atual é concluída, possivelmente deixando as mensagens no buffer interno de leitura antecipada, que depois são descartadas.	O que acontece com as mensagens no buffer interno de leitura antecipada quando o MDB é interrompido pelo administrador.
receiveCCSID	int	• **0** - Use JVM `Charset.defaultCharset` • 1208 - UTF-8 • Um identificador de conjunto de caracteres codificados suportados	A propriedade de destino que configura o destino CCSID para a conversão de mensagens do gerenciador de filas. O valor é ignorado a menos que **`receiveConversion`** seja configurado como `QMGR`
receiveConversion	Sequência	• **CLIENT_MSG** • QMGR	A propriedade de destino que determina se a conversão de dados será executada pelo gerenciador de filas.
startTimeout	int	• **10 000** • Um inteiro positivo	O tempo, em milissegundos, dentro do qual uma entrega de mensagem para um MDB deverá iniciar depois de o trabalho de entrega da mensagem ter sido planejado. Se esse tempo expirar, a mensagem será retrocedida na fila.
subscriptionDurability	Sequência	• **Não durável** – A assinatura não durável é usada para entregar mensagens para uma assinatura MDB para o tópico. • – Durável Uma assinatura durável é usada para entregar mensagens para uma assinatura MDB para o tópico.	Se uma assinatura durável ou não durável é usada para entregar mensagens para uma assinatura MDB para o tópico
subscriptionName	Sequência	• **"" (empty string)** • Um nome de assinatura	O nome da assinatura durável

Tabela 97. Propriedades de um objeto ActivationSpec usado para criar um consumidor de conexão JMS (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
useJNDI	Boole ana	• **false** -A propriedade chamada destino é interpretada como o nome de uma fila ou um tópico do WebSphere MQ . • true – A propriedade chamada de destino é interpretada como o nome de um objeto javax.jms.Queue ou javax.jms.Topic no namespace JNDI do servidor de aplicativos.	Determina como o valor da propriedade chamada de destino é interpretado

As propriedades ActivationSpec chamadas de destino e destinationType devem ser definidas explicitamente. Todas as outras propriedades são opcionais.

Um objeto ActivationSpec pode ter propriedades conflitantes. Por exemplo, é possível especificar propriedades SSL para uma conexão no modo de ligações. Neste caso, o comportamento será determinado pelo tipo de transporte e o domínio de mensagens, que é um ponto a ponto ou de publicação/assinatura, conforme determinado pela propriedade destinationType. Todas as propriedades que não são aplicáveis ao tipo de transporte especificado ou domínio de sistema de mensagens são ignoradas.

Se você definir uma propriedade que requer que outras propriedades sejam definidas, mas não definir estas outras propriedades, o objeto ActivationSpec emitirá uma exceção InvalidPropertyException quando seu método de validação() for chamado durante a implementação de um MDB. A exceção é relatada ao administrador do servidor de aplicativos de uma maneira que depende do servidor de aplicativos. Por exemplo, se você configurar a propriedade subscriptionDurability para Durável, indicando que você deseja utilizar assinaturas duráveis, também deverá definir a propriedade subscriptionName.

Se a propriedade chamada ccdtURL e o canal forem definidos, uma exceção InvalidPropertyException será emitida. No entanto, se você definir a propriedade ccdtURL apenas, deixando a propriedade chamada canal com seu valor padrão SYSTEM.DEF.SVRCONN, nenhuma exceção é lançada e a tabela de definições de canal do cliente identificada pela propriedade ccdtURL é usada para iniciar uma conexão JMS.

A maioria das propriedades de um objeto ActivationSpec são equivalentes a propriedades de classes WebSphere MQ para objetos JMS ou parâmetros de classes WebSphere MQ para métodos JMS. No entanto, três propriedades de ajuste e uma propriedade de usabilidade não possuem equivalentes nas classes WebSphere MQ para JMS:

startTimeout
O tempo, em milissegundos, em que o gerenciador de trabalho do servidor de aplicativos aguarda para que os recursos fiquem disponíveis depois que o adaptador de recursos planeja um objeto de trabalho para entregar uma mensagem para um MDB. Se esse tempo decorrer antes da entrega da mensagem ser iniciada, o objeto de trabalho atinge o tempo limite, a mensagem é retrocedida para a fila e o adaptador de recursos pode tentar entregar a mensagem novamente. Um aviso é gravada para rastreio de diagnóstico, se ativado, mas, do contrário, não afeta o processo de entrega de mensagens. Você pode esperar essa condição somente nos momentos em que o servidor de aplicativos estiver passando por uma carga muito alta. Se a condição ocorrer regularmente, considere aumentar o valor desta propriedade para dar o gerenciador de trabalho mais tempo para planejar a entrega de mensagens.

maxPoolDepth
O número máximo de sessões do servidor no conjunto de sessões do servidor usado por um consumidor de conexão. Quando uma sessão do servidor é criada, ela inicia uma conversa com um gerenciador de filas. O consumidor de conexão usa uma sessão do servidor para entregar uma mensagem para um MDB. Uma profundidade de conjunto maior permite que mais mensagens sejam entregues simultaneamente em situações de alto volume, mas usa mais recursos do servidor de aplicativos. Se vários MDBs tiverem que ser implementados, considere tornar a profundidade do conjunto menor para manter o carregamento no servidor de aplicativos em um nível gerenciável. Cada consumidor de conexão usa seu próprio servidor do conjunto de sessão, de modo que essa propriedade não define o número total de sessões de servidor disponíveis para todos os consumidores de conexão.

poolTimeout
O tempo, em milissegundos, que uma sessão de servidor não usada é mantida aberta no conjunto de sessões do servidor antes de ser fechada devido à inatividade. Um aumento temporário na carga de trabalho da mensagem faz com que as sessões do servidor adicionais sejam criadas a fim de distribuir a carga mas, após a carga de trabalho da mensagem retorna ao normal, as sessões adicionais do servidor permanecem no conjunto e não são usadas.

Toda vez que uma sessão do servidor for usada, ela será marcado com um registro de data e hora. Periodicamente, um encadeamento scavenger verifica que cada sessão de servidor foi usada dentro do período especificado por essa propriedade. Se uma sessão do servidor não foi usada, ela é fechada e removida do conjunto de sessões do servidor. Uma sessão do servidor não pode ser fechada imediatamente após o período especificado ter decorrido, esta propriedade representa o período mínimo de inatividade antes da remoção.

useJNDI
Para obter uma descrição desta propriedade, consulte Tabela 97 na página 756.

Para implementar um MDB, primeiro defina as propriedades de um objeto ActivationSpec, especificando as propriedades que o MDB requer. O exemplo a seguir é um conjunto típico de propriedades que podem ser definidas explicitamente:

```
channel:           SYSTEM.DEF.SVRCONN
destination:       SYSTEM.DEFAULT.LOCAL.QUEUE
destinationType:   javax.jms.Queue
hostName:          192.168.0.42
messageSelector:   color='red'
port:              1414
queueManager:      ExampleQM
transportType:     CLIENT
```

O servidor de aplicativos usa as propriedades para criar um objeto ActivationSpec, que é, então, associado a um MDB. As propriedades do objeto ActivationSpec determinam como as mensagens são entregues para o MDB. A implementação do MDB falhará se o MDB requer transações distribuídas, mas o adaptador de recursos não suporta transações distribuídas. Para obter informações sobre como instalar o adaptador de recursos para que as transações distribuídas sejam suportados, consulte “Instalação do adaptador de recursos do WebSphere MQ” na página 743.

Se mais de um MDB está recebendo mensagens do mesmo destino, uma mensagem enviada no domínio ponto a ponto é recebida então por apenas um MDB, mesmo se outros MDBs são elegíveis para receber a mensagem. Em particular, se dois MDBs estiverem usando seletores de mensagens diferentes, e uma mensagem recebida corresponde a ambos os seletores de mensagens, apenas um dos MDBs recebe a mensagem. O MDB escolhido para receber uma mensagem é indefinido, e não é possível confiar em um MDB específico que recebe a mensagem. As mensagens enviadas no domínio de publicação/assinatura são recebidas por todos os MDBs elegíveis.

Manipulação da mensagem de entrada suspeita no Adaptador de recursos

Em algumas circunstâncias, uma mensagem entregue para um MDB pode ser retrocedida para uma fila do WebSphere MQ Esse retrocesso pode ocorrer, por exemplo, se uma mensagem é entregue em uma unidade de trabalho que é, então, retrocedida. Uma mensagem que é retrocedida é entregue novamente, mas uma mensagem formatada incorretamente pode causar repetidamente a falha de um MDB e, portanto, não pode ser entregue. Essa mensagem é chamada de uma mensagem suspeita. É possível

configurar o WebSphere MQ para que as classes WebSphere MQ para JMS transfira automaticamente uma mensagem suspeita para outra fila para investigação adicional ou descarte a mensagem.

Para obter detalhes sobre como manipular mensagens suspeitas, consulte “Manipulando mensagens suspeitas no IBM WebSphere MQ classes for JMS” na página 901.

Tarefas relacionadas

Especificando que Apenas CipherSpecs Certificados por FIPS São Usados no Tempo de Execução no Cliente de MQI

Referências relacionadas

Federal Information Processing Standards (FIPS) para UNIX, Linux e Windows

Configurando o adaptador de recursos para comunicação de saída

Para configurar a comunicação de saída, defina as propriedades de um objeto ConnectionFactory e um objeto de destino administrado.

Ao usar a comunicação de saída, um aplicativo em execução em um servidor de aplicativos inicia uma conexão com um gerenciador de filas e, em seguida, envia mensagens para suas filas e recebe mensagens de suas filas de maneira síncrona. Por exemplo, o método de servlet a seguir, doGet(), usa comunicação de saída:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException {

...

// Look up ConnectionFactory and Queue objects from the JNDI namespace

     InitialContext ic = new InitialContext();
     ConnectionFactory cf = (javax.jms.ConnectionFactory) ic.lookup("myCF");
     Queue q = (javax.jms.Queue) ic.lookup("myQueue");

// Create and start a connection

     Connection c = cf.createConnection();
     c.start();

// Create a session and message producer

     Session s = c.createSession(false, Session.AUTO_ACKNOWLEDGE);
     MessageProducer pr = s.createProducer(q);

// Create and send a message

     Message m = s.createTextMessage("Hello, World!");
     pr.send(m);

// Create a message consumer and receive the message just sent

     MessageConsumer co = s.createConsumer(q);
     Message mr = co.receive(5000);

// Close the connection

      c.close();
}
```

Quando o servlet recebe uma solicitação HTTP GET, ele recupera um objeto ConnectionFactory e um objeto Queue do namespace JNDI e usa os objetos para enviar uma mensagem para uma fila WebSphere MQ . O servlet, então, recebe a mensagem que ele enviou.

Para configurar a comunicação de saída, defina recursos JCA nas seguintes categorias:

- As propriedades de um objeto ConnectionFactory , que o servidor de aplicativos usa para criar um objeto JMS ConnectionFactory .
- As propriedades de um objeto de destino administrado, que o servidor de aplicativos usa para criar um objeto de Fila JMS ou um objeto de Tópico JMS

A maneira como você define essas propriedades depende das interfaces de administração fornecidas por seu servidor de aplicativos. Os objetos ConnectionFactory, Queue e Topic criados pelo servidor de

aplicativos são ligados a um namespace JNDI a partir de onde eles podem ser recuperados por um aplicativo.

Geralmente, você define um objeto ConnectionFactory para cada gerenciador de filas para os quais os aplicativos podem precisar se conectar. Você define um objeto Queue para cada fila que os aplicativos podem precisar acessar no domínio ponto a ponto. E define um objeto Topic para cada tópico para os quais os aplicativos podem desejar publicar ou assinar. Um objeto ConnectionFactory pode ser independente do domínio. Como alternativa, ele pode ser específico do domínio, um objeto QueueConnectionFactory para o domínio ponto a ponto ou um objeto TopicConnectionFactory para o domínio de publicar/assinar.

O Tabela 98 na página 762 lista as propriedades de um objeto ConnectionFactory.

Tabela 98. Propriedades de um objeto ConnectionFactory

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
applicationName	Sequência	• O nome da classe de chamada, se estiver disponível, ajustado para não ter mais de 28 caracteres. Se ela não estiver disponível, a sequência `WebSphere MQ Client for Java` será usada.	O nome pelo qual um aplicativo é registrado com o gerenciador de filas. Esse nome do aplicativo é mostrado pelo comando **DISPLAY CONN MQSC/PCF** (em que o campo é chamado **APPLTAG**) ou na exibição IBM WebSphere MQ Explorer **Conexões de Aplicativos** (em que o campo é chamado **App name**).
brokerCCSubQueue [1]	Sequência	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Um nome da fila	O nome da fila da qual um consumidor de conexão recebe mensagens de assinaturas não duráveis.
brokerControlQueue [1]	Sequência	• **SYSTEM.BROKER.CONTROL.QUEUE** • Um nome da fila	O nome da fila de controle do broker.
brokerPubQueue[1]	Sequência	• **SYSTEM.BROKER.DEFAULT.STREAM** • Um nome da fila	O nome da fila para onde as mensagens publicadas são enviadas (a fila de fluxo).
brokerQueueManager [1]	Sequência	• **"" (empty string)** • Um nome do gerenciador de filas	O nome do gerenciador de filas no qual o broker está em execução.
brokerSubQueue[1]	Sequência	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Um nome da fila	O nome da fila a partir da qual um consumidor de mensagens não duráveis recebe as mensagens. Consulte a propriedade BROKERSUBQ para obter mais informações.

Tabela 98. Propriedades de um objeto ConnectionFactory (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
brokerVersion[1]	Sequência	• **unspecified** – Depois que o broker foi migrado da V6 para V7, configure essa propriedade de modo que os cabeçalhos RFH2 não sejam mais usados. Após a migração, essa propriedade não é mais relevante. • **V1** -Para usar um broker de Publicação / Assinatura do IBM WebSphere MQ . Ou para usar um broker de IBM WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker ou WebSphere Business Integration Message Broker no modo de compatibilidade.. Este valor é o valor padrão, se TRANSPORT estiver definido como BIND ou CLIENT. • **V2** -Para usar um broker do IBM WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker, ou WebSphere Business Integration Message Broker no modo nativo. Este valor é o valor padrão, se TRANSPORT estiver definido como DIRECT ou DIRECTHTTP.	A versão do broker que está sendo usada.
ccdtURL	Sequência	• **null** • Um localizador uniforme de recursos (URL)	Uma URL que identifica o nome e o local do arquivo contendo a tabela de definição de canal de cliente (CCDT) e especifica como o arquivo pode ser acessado.
CCSID	Sequência	• **819** • Um identificador de conjunto de caracteres codificados suportados pela máquina virtual Java (JVM)	O identificador do conjunto de caracteres codificado para uma conexão.
channel	Sequência	• **SYSTEM.DEF.SVRCONN** • O nome de um canal de MQI	O nome do canal de MQI a ser usado.
cleanupInterval [1]	int	• **3 600 000** • Um inteiro positivo	O intervalo, em milissegundos, entre as execuções em segundo plano do utilitário de limpeza de publicação/assinatura.

Tabela 98. Propriedades de um objeto ConnectionFactory (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
cleanupLevel[1]	Sequência	• **SAFE** • Nenhum • STRONG • FORCE • NONDUR	O nível de limpeza para um armazenamento de assinaturas baseado no broker.
clientID	Sequência	• **null** • Um identificador de cliente	O identificador do cliente para uma conexão.
cloneSupport	Sequência	• **DISABLED** – Apenas uma instância de um assinante de tópico durável pode ser executada de cada vez. • ENABLED - Duas ou mais instâncias do mesmo assinante de tópico durável podem ser executadas simultaneamente, mas cada instância deve ser executada em uma máquina virtual Java (JVM) separada.	Se duas ou mais instâncias do mesmo assinante de tópico durável puderem ser executadas simultaneamente.
connectionNameList	Sequência	• **localhost(1414)** • Uma sequência composta de itens separados por vírgulas, em que cada item tem o formato: `HOSTNAME(PORT)` em que *HOSTNAME* é um nome DNS ou um endereço IP.	Uma lista de nomes de conexão TCP/IP usada para comunicações de saída. **connectionNameList** substitui as propriedades **hostname** e **port** . Essa propriedade é usada para se reconectar a gerenciadores de filas de várias instâncias. **connectionNameList** é semelhante em forma a **localAddress**, mas não deve ser confundida com ela. **localAddress** especifica as características das comunicações locais, enquanto **connectionNameList** especifica como alcançar um gerenciador de filas remotas.
failIfQuiesce	Booleana	• **true** • false	Indica se as chamadas para certos métodos falharão se o gerenciador de filas estiver em um estado de quiesce.
headerCompression	Sequência	• **NONE** • SYSTEM - a compactação do cabeçalho da mensagem de RLE é executada.	Uma lista de técnicas que podem ser usadas para compactar os dados do cabeçalho em uma conexão.

Tabela 98. Propriedades de um objeto ConnectionFactory (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
hostName	Sequência	• **localhost** • Um nome do host • Um endereço IP	O nome do host ou o endereço IP do sistema em que o gerenciador de filas reside. As propriedades **hostname** e **port** são substituídas pela propriedade **connectionNameList** quando especificada.

localAddress	Sequência	• **null**
		• Uma sequência no formato:
		`[host_name]`
		`[(low_port[,high_port])]`
		em que *host_name* é um nome do host ou endereço IP, *low_port* e *high_port* são números de porta TCP e colchetes denotam um componente opcional
			Para uma conexão com um gerenciador de filas, esta propriedade especifica um ou ambos:
			• A interface de rede local a ser usada
			• A porta local ou um intervalo de portas locais a ser usado
			localAddress é semelhante em forma a **connectionNameList**, mas não deve ser confundida com ela. **localAddress** especifica as características das comunicações locais, enquanto **connectionNameList** especifica como alcançar um gerenciador de filas remotas.

messageCompression	Sequência	• **NONE** • Uma lista de um ou mais dos seguintes valores separados por caracteres em branco: RLE ZLIBFAST ZLIBHIGH	Uma lista de técnicas que podem ser usadas para compactar os dados da mensagem em uma conexão.
messageSelection [1]	Sequência	• **CLIENT** • BROKER	Determina se a seleção de mensagens é feita por classes IBM WebSphere MQ para JMS ou pelo broker A seleção de mensagens pelo broker não é suportada quando brokerVersion tem o valor 1.
senha	Sequência	• **null** • Uma senha	A senha padrão a ser usada ao criar uma conexão com o gerenciador de filas.

Tabela 98. Propriedades de um objeto ConnectionFactory (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
pollingInterval[1]	int	• **5000** • Qualquer número inteiro positivo	Se cada listener de mensagem dentro de uma sessão não tiver mensagens adequadas em sua fila, este valor será o intervalo máximo, em milissegundos, que decorrerá antes que cada listener da mensagem tente novamente obter uma mensagem de sua fila. Se ocorrer com frequência o fato de nenhuma mensagem adequada estar disponível para qualquer um dos listeners da mensagem em uma sessão, considere aumentar o valor desta propriedade. Esta propriedade é relevante apenas se TRANSPORT tiver o valor BIND ou CLIENT.
port	int	• **1414** • Um número da porta TCP	A porta na qual o gerenciador de filas atende. As propriedades **hostname** e **port** são substituídas pela propriedade **connectionNameList** quando especificada.
providerVersion	sequência	• **unspecified** • Uma sequência em um dos formatos a seguir – V.R.M.F – V.R.M – V.R – V em que V, R, M e F são valores de números inteiros maiores ou iguais a zero.	A versão, liberação, nível de modificação e fix pack do gerenciador de filas ao qual o aplicativo pretende se conectar.
pubAckInterval[1]	int	• **25** • Um inteiro positivo	O número de mensagens publicadas por um publicador antes de o IBM WebSphere MQ classes for JMS solicitar uma confirmação do broker.
queueManager	Sequência	• **"" (empty string)** • Um nome do gerenciador de filas	O nome do gerenciador de filas para conexão.

Tabela 98. Propriedades de um objeto ConnectionFactory (continuação)			
Nome da propriedade	**tipo**	**Valores válidos (valor padrão em negrito)**	**Descrição**
receiveExit [3]	Sequência	• **null** • Uma sequência composta por um ou mais itens separados por vírgulas, em que cada item é o nome completo de uma classe que implementa a interface do IBM WebSphere MQ classes for Java, MQReceiveExit	Identifica um programa de saída de recebimento de canal ou uma sequência de programas de saída de recebimento a ser executada na sucessão.
receiveExitInit	Sequência	• **null** • Uma sequência composta de um ou mais itens de dados do usuário separados por vírgulas	Os dados do usuário que são transmitidos para programas de saída de recebimento do canal quando são chamados.
rescanInterval[1]	int	• **5000** • Qualquer número inteiro positivo	Quando um consumidor de mensagens no domínio ponto a ponto usa um seletor de mensagem para selecionar quais mensagens ele deseja receber, o WebSphere MQ classes para JMS procura na fila IBM WebSphere MQ mensagens adequadas na sequência determinada pelo atributo *MsgDeliverySequence* da fila. Quando as classes do WebSphere MQ para JMS localizam uma mensagem adequada e a entregam ao consumidor, as classes do WebSphere MQ para JMS retoma a procura para a próxima mensagem adequada a partir de sua posição atual na fila. WebSphere MQ classes para JMS continua a procurar a fila desta maneira até atingir o final da fila ou até o intervalo de tempo em milissegundos, conforme determinado pelo valor desta propriedade, ter expirado. Em cada caso, as classes WebSphere MQ para JMS retornam ao início da fila para continuar sua procura e um novo intervalo de tempo começa.
securityExit [3]	Sequência	• **null** • O nome completo de uma classe que implementa as classes WebSphere MQ para a interface Java, MQSecurityExit	Identifica um programa de saída de segurança do canal.

Tabela 98. Propriedades de um objeto ConnectionFactory (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
securityExitInit	Sequência	• **null** • Uma sequência de dados do usuário	Os dados do usuário que são transmitidos para um programa de saída de segurança do canal quando ele é chamado.
sendCheckCount	int	• **0** • Qualquer número inteiro positivo	O número de chamadas de envio a serem permitidas entre a verificação de erros de colocação assíncronos, em uma única sessão JMS não transacionada.
sendExit [3]	Sequência	• **null** • Uma sequência composta por um ou mais itens separados por vírgulas, em que cada item é o nome completo de uma classe que implementa as classes WebSphere MQ para interface Java, MQSendExit	Identifica um programa de saída de envio de canal ou uma sequência de programas de saída de envio a serem executadas na sucessão.
sendExitInit	Sequência	• **null** • Uma sequência composta de um ou mais itens de dados do usuário separados por vírgulas	Os dados do usuário que são transmitidos para programas de saída de envio do canal quando são chamados.
shareConvAllowed	Booleana	• NO-Uma conexão do cliente não pode compartilhar seu soquete.. • **YES** -Uma conexão do cliente pode compartilhar seu soquete	Se uma conexão do cliente pode compartilhar seu soquete com outras conexões JMS de nível superior do mesmo processo para o mesmo gerenciador de filas, se as definições de canal corresponderem.
sparseSubscriptions[1]	Booleana	• **false** – As assinaturas recebem mensagens de correspondência frequentes. • true – As assinaturas recebem mensagens de correspondência não frequentes. Esse valor requer que a fila de assinaturas possa ser aberta para procurar.	Controla a política de recuperação de mensagens de um objeto TopicSubscriber.

Tabela 98. Propriedades de um objeto ConnectionFactory (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
sslCertStores	Sequência	• **null** • Uma sequência de uma ou mais URLs de LDAP separadas por espaços em branco. Cada URL de LDAP possui o formato: `ldap://host_name[:port]` em que *host_name* é um nome de host ou endereço IP, *port* é um número de porta TCP e colchetes denotam um componente opcional.	Os servidores Lightweight Directory Access Protocol (LDAP) que retêm as listas de revogação de certificado (CRLs) para uso em uma conexão SSL.
sslCipherSuite	Sequência	• **null** • O nome de um CipherSuite	O CipherSuite a ser usado para uma conexão SSL.
sslFipsRequired [2]	Booleana	• **falso** • true	Se uma conexão SSL deve usar um CipherSuite que seja suportado pelo provedor JSSE FIPS IBM Java (IBMJSSEFIPS).
sslPeerName	Sequência	• **null** • Um modelo para nomes distintos	Para uma conexão SSL, um modelo que é usado para verificar o nome distinto no certificado digital fornecido pelo gerenciador de filas.
sslResetCount	int	• **0** • Um número inteiro no intervalo de 0 – 999999999	O número total de bytes enviados e recebidos por uma conexão SSL antes de as chaves secretas usadas pelo SSL serem renegociadas.
sslSocketFactory	Sequência	Uma sequência que representa o nome de classe completo de uma classe que fornece uma implementação da interface javax.net.ssl.SSLSocketFactory, incluindo opcionalmente um argumento a ser transmitido para o método do construtor, entre parênteses.	Quaisquer conexões estabelecidas no escopo do objeto de destino administrado usam soquetes obtidos a partir desta implementação da interface SSLSocketFactory.
statusRefreshInterval[1]	int	• **60000** • Qualquer número inteiro positivo	O intervalo, em milissegundos, entre atualizações da transação de execução longa, que detecta quando um assinante perde sua conexão com o gerenciador de filas. Esta propriedade é relevante apenas se SUBSTORE possuir o valor QUEUE.
subscriptionStore [1]	Sequência	• **Broker** • MIGRATE • FILA	Determina onde as classes do WebSphere MQ para JMS armazenem dados persistentes sobre assinaturas ativas.

Tabela 98. Propriedades de um objeto ConnectionFactory (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
targetClientMatching	Booleana	• **true** • false	Se uma mensagem de resposta, enviada para a fila identificada pelo campo de cabeçalho JMSReplyTo de uma mensagem recebida, terá um cabeçalho MQRFH2 apenas se a mensagem recebida tiver um cabeçalho MQRFH2.

Tabela 98. Propriedades de um objeto ConnectionFactory (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
temporaryModel	Sequência	• **SYSTEM.DEFAULT.MODEL.QUEUE** • SYSTEM.JMS.TEMPQ.MODEL • Qualquer sequência	O nome da fila modelo a partir da qual filas temporárias JMS são criadas. Use SYSTEM.DEFAULT.MODEL.QUEUE se as duas condições a seguir forem verdadeiras: • Seu aplicativo usa uma fila temporária que aceitará mensagens não persistentes. • Apenas um aplicativo criará uma fila temporária no gerenciador de filas para a qual o ConnectionFactory aponta de cada vez. Observe que SYSTEM.DEFAULT.MODEL.QUEUE só pode ser aberto por um aplicativo de cada vez. Use SYSTEM.JMS.TEMPQ.MODEL. nas seguintes situações: • Quando o aplicativo usa uma fila temporária que aceitará mensagens persistentes. • Se vários aplicativos podem se conectar ao gerenciador de filas para o qual o ConnectionFactory aponta, e esses aplicativos precisam criar filas temporárias ao mesmo tempo. Defina uma nova fila modelo com o atributo DEFPSIST configurado para YES, e o atributo DEFSOPT configurado como SHARED na seguinte situação: • Quando o aplicativo usa uma fila temporária que aceitará mensagens não persistentes e vários aplicativos se conectarão ao gerenciador de filas para o qual o ConnectionFactory, e esses aplicativos precisam criar filas temporárias ao mesmo tempo. Quando a nova fila modelo é criada, configure a propriedade temporaryModel para o nome da nova fila modelo.

Tabela 98. Propriedades de um objeto ConnectionFactory (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
tempQPrefix	Sequência	• **"" (empty string)** • Um prefixo que pode ser usado para formar o nome de uma fila dinâmica do IBM WebSphere MQ As regras para formar o prefixo são as mesmas que as regras para formar o conteúdo do campo *DynamicQName* em um descritor de objeto IBM WebSphere MQ , estrutura MQOD, mas o último caractere não em branco deve ser um asterisco (*). Se o valor da propriedade for a sequência de caracteres vazia, as classes do WebSphere MQ para JMS usará o valor AMQ.* ao criar uma fila dinâmica.	O prefixo que é usado para formar o nome de uma fila dinâmica do IBM WebSphere MQ.
tempTopicPrefix	Sequência	Qualquer sequência não nula que consiste apenas de caracteres válidos para uma sequência de tópico do IBM WebSphere MQ	Ao criar tópicos temporários, o JMS gera uma sequência de tópicos no formato "TEMP/*TEMPTOPICPREFIX*/*unique_id*" ou, se essa propriedade for deixada com o valor padrão, apenas "TEMP/*unique_id*". Especificar um TEMPTOPICPREFIX não vazio permite que filas modelo específicas sejam definidas para criar as filas gerenciadas para assinantes para tópicos temporários criados nessa conexão.
transportType	Sequência	• **CLIENT** • BINDINGS • BINDINGS_THEN_CLIENT	Indica se uma conexão com um gerenciador de filas usa o modo cliente ou o modo de ligações. Se o valor BINDINGS_THEN_CLIENT for especificado, o adaptador de recursos primeiro tenta fazer uma conexão no modo de ligações. Se essa tentativa de conexão falhar, o adaptador de recursos tentará estabelecer uma conexão do modo cliente.
nome do usuário	Sequência	• **null** • Um nome de usuário	O nome do usuário padrão a ser usado ao criar uma conexão com um gerenciador de filas.
wildcardFormat	int	• CHAR - Reconhece somente os caracteres curingas, conforme usados na versão 1 do broker • TOPIC – Reconhece somente curingas em nível do tópico, conforme usados na versão 2 do broker	Qual versão de sintaxe curinga deve ser usada.

Tabela 98. Propriedades de um objeto ConnectionFactory (continuação)			
Nome da propriedade	**tipo**	**Valores válidos (valor padrão em negrito)**	**Descrição**
Notas: 1. Essa propriedade pode ser usada com a versão 7.0 do IBM WebSphere MQ classes for JMS , mas não afeta um aplicativo conectado a um gerenciador de filas da versão 7.0 , a menos que a propriedade providerVersion seja configurada para um número de versão menor que 7 2. Para obter informações importantes sobre o uso da propriedade sslFipsRequired, consulte "Limitações do adaptador de recursos IBM WebSphere MQ" na página 781. 3. Para obter informações sobre como configurar o adaptador de recursos para que ele possa localizar uma saída, veja "Configurando o IBM WebSphere MQ classes for JMS para usar saídas de canal" na página 924.			

O exemplo a seguir mostra um conjunto típico de propriedades de um objeto ConnectionFactory:

```
channel:        SYSTEM.DEF.SVRCONN
hostName:       192.168.0.42
port:           1414
queueManager:   ExampleQM
transportType:  CLIENT
```

O Tabela 99 na página 773 lista as propriedades que são comuns para um objeto Fila e um objeto Tópico.

Tabela 99. Propriedades que são comuns para um objeto Fila e um objeto Tópico			
Nome da propriedade	**tipo**	**Valores válidos (valor padrão em negrito)**	**Descrição**
CCSID	Sequência	• **1208** • Um identificador de conjunto de caracteres codificados suportados pela máquina virtual Java (JVM)	O identificador do conjunto de caracteres codificado para um destino.
codificação	Sequência	• **NATIVE** • Uma sequência de três caracteres: – O primeiro caractere especifica a representação de números inteiros binários: - *N* denota a codificação normal. - *R* denota a codificação reversa. – O segundo caractere especifica a representação de números inteiros decimais compactados: - *N* denota a codificação normal. - *R* denota a codificação reversa. – O terceiro caractere especifica a representação de números de vírgula flutuante: - *N* denota a codificação IEEE padrão. - *R* denota a codificação IEEE reversa. - *3* denota a codificação zSeries NATIVE é equivalente à sequência NNN.	A representação de números inteiros binários, números inteiros decimais compactados e números de vírgula flutuante, para o destino.

Tabela 99. Propriedades que são comuns para um objeto Fila e um objeto Tópico (continuação)

Nome da propriedade	tipo	Valores válidos (valor padrão em negrito)	Descrição
expiração	Sequência	• **APP** – O tempo de expiração de uma mensagem é determinado pelo produtor da mensagem. • UNLIM - Uma mensagem nunca expira. • 0 – Uma mensagem nunca expira. • Um número inteiro positivo representando o tempo de expiração de uma mensagem em milissegundos.	O tempo de expiração de uma mensagem enviada para o destino.
failIfQuiesce	Sequência	• **true** • false	Se uma tentativa de acessar o destino falhar se o gerenciador de filas estiver em um estado de quiesce.
persistence	Sequência	• **APP** – A persistência de uma mensagem é determinada pelo produtor de mensagem. • QDEF-A persistência de uma mensagem é determinada pelo atributo *DefPersistence* da fila WebSphere MQ . • PERS - Uma mensagem é persistente. • NON - Uma mensagem é não persistente. • HIGH-A persistência de uma mensagem é determinada pelo atributo *NonPersistentMessageClass* da fila WebSphere MQ de acordo com a explicação em "Mensagens persistentes do JMS" na página 916.	A persistência de uma mensagem enviada para o destino.
priority	Sequência	• **APP** – A prioridade de uma mensagem é determinada pelo produtor da mensagem. • QDEF – A prioridade de uma mensagem é determinado pelo atributo *DefPriority* da fila do IBM WebSphere MQ. • Um número inteiro no intervalo de 0, a prioridade mais baixa, e 9 para prioridade mais alta.	A prioridade de uma mensagem enviada para o destino.

Tabela 99. Propriedades que são comuns para um objeto Fila e um objeto Tópico (continuação)			
Nome da propriedade	**tipo**	**Valores válidos (valor padrão em negrito)**	**Descrição**
putAsyncAllowed	Sequência	• QUEUE – Determine se as colocações assíncronas são permitidas consultando a definição de fila. • TOPIC – Determine se as colocações assíncronas são permitidas consultando a definição de tópico. • DESTINO – Determine se as colocações assíncronas são permitidas consultando a definição de fila ou tópico. • DISABLED – Colocações assíncronas não são permitidas. • ENABLED - Colocações assíncronas são permitidas.	Indica se os produtores de mensagens têm permissão para usar postagens assíncronas para enviar mensagens para este destino.
readAheadAllowed	int	• **DESTINATION** – Determine se a leitura antecipada será permitida ao consultar a definição da fila ou do tópico. • DISABLED - A leitura antecipada não é permitida. • ENABLED – A leitura antecipada é permitida. • QUEUE – Determine se a leitura antecipada é permitida, fazendo referência à definição de fila. • TOPIC – Determine se a leitura antecipada é permitida, fazendo referência à definição de tópico.	Se os consumidores de mensagens e os navegadores de filas têm permissão para usar a leitura antecipada para obter as mensagens não persistentes do destino em um buffer interno antes de recebê-las.
receiveCCSID	int	• **0** - Use JVM `Charset.defaultCharset` • 1208 - UTF-8 • Um identificador de conjunto de caracteres codificados suportados	A propriedade de destino que configura o destino CCSID para a conversão de mensagens do gerenciador de filas. O valor é ignorado a menos que **`receiveConversion`** seja configurado como `QMGR`
receiveConversion	Sequência	• **CLIENT_MSG** • QMGR	A propriedade de destino que determina se a conversão de dados será executada pelo gerenciador de filas.
targetClient	Sequência	• **JMS** -O destino de uma mensagem é um aplicativo JMS.. • MQ -O destino de uma mensagem é um aplicativo IBM WebSphere MQ não JMS.	Se o destino de uma mensagem enviada ao destino é um aplicativo JMS. Uma mensagem com um destino que é um aplicativo JMS contém um cabeçalho MQRFH2 .

O Tabela 100 na página 776 lista as propriedades que são específicas para um objeto Fila.

Tabela 100. Propriedades específicas de um objeto Fila			
Nome da propriedade	**tipo**	**Valores válidos (valor padrão em negrito)**	**Descrição**
baseQueueManagerName	Sequência	• **"" (empty string)** • Um nome do gerenciador de filas	O nome do gerenciador de filas que possui a fila subjacente do IBM WebSphere MQ.
baseQueueName	Sequência	• **"" (empty string)** • Um nome da fila	O nome da fila subjacente do IBM WebSphere MQ

O Tabela 101 na página 776 lista as propriedades que são específicas para um objeto Tópico.

Tabela 101. As propriedades que são específicas para um objeto Tópico			
Nome da propriedade	**tipo**	**Valores válidos (valor padrão em negrito)**	**Descrição**
baseTopicName	Sequência	• **"" (empty string)** • Um nome de tópico	O nome do tópico subjacente.
brokerCCDurSubQueue [1]	Sequência	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Um nome da fila	O nome da fila a partir da qual um consumidor de conexão recebe mensagens de assinaturas duráveis.
brokerDurSubQueue [1]	Sequência	• **SYSTEM.JMS.D.SUBSCRIBER.QUEUE** • Um nome da fila	O nome da fila a partir da qual um assinante de tópico durável recebe mensagens. Consulte a propriedade BROKEDURRSUBQ na documentação do WebSphere MQ Explorer para obter mais informações.
brokerPubQueue[1]	Sequência	• **Not set** • Um nome da fila	O nome da fila para onde as mensagens publicadas são enviadas (a fila de fluxo). O valor desta propriedade substitui o valor da propriedade brokerPubQueue do objeto ConnectionFactory. No entanto, se você não configurar o valor desta propriedade, o valor da propriedade brokerPubQueue do objeto ConnectionFactory será usado em seu lugar.
brokerPubQueueManager [1]	Sequência	• **"" (empty string)** • Um nome do gerenciador de filas	O nome do gerenciador de filas que possui a fila em que mensagens publicadas sobre o tópico são enviadas.

Tabela 101. As propriedades que são específicas para um objeto Tópico (continuação)			
Nome da propriedade	**tipo**	**Valores válidos (valor padrão em negrito)**	**Descrição**
brokerVersion[1]	Sequência	• **Not set** • 1 • 2	A versão do broker que está sendo usada. O valor desta propriedade substitui o valor da propriedade brokerVersion do objeto ConnectionFactory. No entanto, se você não configurar o valor desta propriedade, o valor da propriedade brokerVersion do objeto ConnectionFactory será usado em seu lugar.

Nota:

1. Essa propriedade pode ser usada com a versão 7.0 de IBM WebSphere MQ classes for JMS , mas não afeta um aplicativo conectado a um gerenciador de filas da versão 7.0 , a menos que a propriedade providerVersion do objeto ConnectionFactory seja configurada para um número de versão menor que 7..

O exemplo a seguir mostra um conjunto de propriedades de um objeto Fila:

```
expiry:                 UNLIM
persistence:            QDEF
baseQueueManagerName:   ExampleQM
baseQueueName:          SYSTEM.JMS.TEMPQ.MODEL
```

O exemplo a seguir mostra um conjunto de propriedades de um objeto Tópico:

```
expiry:                 UNLIM
persistence:            NON
baseTopicName:          myTestTopic
```

Tarefas relacionadas

Especificando que Apenas CipherSpecs Certificados por FIPS São Usados no Tempo de Execução no Cliente de MQI

Referências relacionadas

Federal Information Processing Standards (FIPS) para UNIX, Linux e Windows

V 7.5.0.9 *Configurando a propriedade targetClientMatching para uma especificação de ativação*

Será possível configurar a propriedade **targetClientMatching** para uma especificação de ativação para que um cabeçalho MQRFH2 seja incluído em mensagens de resposta quando as mensagens de solicitação não contiverem um cabeçalho MQRFH2. Isso significa que quaisquer propriedades de mensagem que um aplicativo definir em uma mensagem de resposta serão incluídas quando a mensagem for enviada.

Sobre esta tarefa

Se um aplicativo bean acionado por mensagens (MDB) consumir mensagens que não contiverem um cabeçalho MQRFH2 por meio de uma especificação de ativação do adaptador de recursos do JCA do IBM WebSphere MQ e subsequentemente enviar mensagens de resposta para o Destino do JMS criado por meio do campo JMSReplyTo da mensagem de solicitação, as mensagens de resposta deverão incluir um cabeçalho MQRFH2, mesmo se as mensagens de solicitação não incluírem, caso contrário, qualquer propriedade de mensagem que o aplicativo tiver definido em uma mensagem de resposta será perdida.

A propriedade **targetClientMatching** define se uma mensagem de resposta enviada para a fila identificada pelo campo de cabeçalho JMSReplyTo de uma mensagem recebida terá um cabeçalho MQRFH2 apenas se a mensagem recebida tiver um cabeçalho MQRFH2. É possível configurar essa propriedade para uma especificação de ativação, no Liberty WebSphere Application Server tradicional e WebSphere Application Server .

Se você configurar o valor da propriedade **targetClientMatching** como `false`, um cabeçalho MQRFH2 poderá ser incluído em uma mensagem de resposta enviada para um Destino JMS criado por meio do cabeçalho JMSReplyTo de uma mensagem de solicitação recebida que não contém um MQRFH2. Isso é porque a propriedade **targetClient** no Destino JMS está configurada com o valor `0`, o que significa que as mensagens contêm um cabeçalho MQRFH2. A presença do cabeçalho MQRFH2 na mensagem de saída permite o armazenamento de propriedades de mensagem definidas pelo usuário na mensagem quando enviadas para a fila IBM WebSphere MQ.

Se a propriedade **targetClientMatching** for configurada como `true` e uma mensagem de solicitação não incluir um cabeçalho MQRFH2, um cabeçalho MQRFH2 não será incluído na mensagem de resposta.

Procedimento

- No WebSphere Application Server tradicional, use o console de administração para definir a propriedade **targetClientMatching** como uma propriedade customizada na especificação de ativação IBM WebSphere MQ :
  a) Na área de janela de navegação, clique em **Recursos -> JMS -> Especificações de ativação**.
  b) Selecione o nome da especificação de ativação que deseja visualizar ou alterar.
  c) Clique em **Propriedades customizadas -> Novo** e, em seguida, insira os detalhes da nova propriedade customizada.
     Configure o nome da propriedade como `targetClientMatching`, o tipo como `java.lang.Boolean` e o valor como `false`.
- No WebSphere Application Server Liberty, especifique a propriedade **targetClientMatching** na definição de uma especificação de ativação no `server.xml`.
  Por exemplo:

```
<jmsActivationSpec id="SimpleMDBApplication/SimpleEchoMDB/SimpleEchoMDB">
<properties.wmqJms destinationRef="MDBRequestQ"
queueManager="MY_QMGR" transportType="BINDINGS" targetClientMatching="false"/>
<authData password="********" user="tom"/>
</jmsActivationSpec>
```

Conceitos relacionados

“Criando destinos em um aplicativo JMS” na página 889
Em vez de recuperar destinos como objetos administrados de um namespace Java Naming and Directory Interface (JNDI), um aplicativo JMS pode usar uma sessão para criar destinos dinamicamente no tempo de execução. Um aplicativo pode usar um identificador uniforme de recursos (URI) para identificar uma fila do WebSphere MQ ou um tópico e, opcionalmente, para especificar uma ou mais propriedades de um objeto de Fila ou Tópico.

“Configurando o adaptador de recursos para comunicação de saída” na página 761
Para configurar a comunicação de saída, defina as propriedades de um objeto ConnectionFactory e um objeto de destino administrado.

Modo ASF e não ASF
O modo Application Server Facilities (ASF) é o método padrão pelo qual o serviço listener de mensagens no WebSphere Application Server processa mensagens.

O serviço listener de mensagens possui dois modos de operação, Application Server Facilities (ASF) e não Application Server Facilities (não ASF):

- O modo ASF fornece suporte simultâneo e transacional para aplicativos. Para beans de unidade de mensagem de publicação / assinatura, o modo ASF fornece melhor rendimento e simultaneidade, porque no modo não ASF, o listener é de encadeamento único
- O modo não ASF é principalmente para uso com provedores de sistemas de mensagens de terceiros que não suportam ASF JMS, que é uma extensão opcional para a especificação JMS. O modo não ASF também é transacional, mas, como o comprimento do caminho é menor do que para o modo ASF, geralmente fornece desempenho aprimorado.

Para ativar o modo de operação não ASF para todos os listeners do bean acionados por mensagem no servidor de aplicativos, configure essa propriedade para um valor diferente de zero.

Nota:

O modo não ASF não pode ser selecionado em sistemas z/OS , portanto, não se deve configurar um valor diferente de zero para essa propriedade neste caso.

Processamento de mensagens no modo ASF..

No modo ASF, as sessões e os encadeamentos do servidor são alocados apenas para trabalho quando uma mensagem adequada para o bean acionado por mensagens (MDB) é detectado. O número de encadeamentos que um MDB pode processar simultaneamente é determinado pelo valor da propriedade **Maximum Sessions** para a porta listener ou especificação de ativação.

Processamento de mensagens no modo não ASF..

No modo não ASF, os encadeamentos estão ativos a partir do momento em que a porta do listener ou a especificação de ativação é iniciada O número de encadeamentos ativos é determinado pelo valor especificado para a propriedade **Maximum Sessions** . O número de encadeamentos especificados na propriedade **Maximum Sessions** está ativo independentemente do número de mensagens que estão disponíveis para processamento. Cada encadeamento ativo é uma conexão de rede física individual

IBM WebSphere MQ Versão 7.0 ou posterior permite ter até dez encadeamentos compartilhando uma única conexão de rede física.

Conceitos relacionados

IBM WebSphere MQ classes para o JMS Application Server Facilities
Este tópico descreve como as classes WebSphere MQ para JMS implementam a classe ConnectionConsumer e a funcionalidade avançada na classe Session. Ele também resume a função de um conjunto de sessões do servidor.

Tarefas relacionadas

Configurando especificações de ativação para o modo não ASF
As especificações de ativação são a maneira padronizada de gerenciar e configurar o relacionamento entre um bean orientado a mensagens (MDB) em execução no WebSphere Application Server e um destino no IBM WebSphere MQ. Esta tarefa explica como configurar o WebSphere Application Server para usar o modo não ASF para processar mensagens

Informações relacionadas

Processamento de mensagens no modo ASF e no modo não ASF

Configurando especificações de ativação para o modo não ASF
As especificações de ativação são a maneira padronizada de gerenciar e configurar o relacionamento entre um bean orientado a mensagens (MDB) em execução no WebSphere Application Server e um destino no IBM WebSphere MQ. Esta tarefa explica como configurar o WebSphere Application Server para usar o modo não ASF para processar mensagens

Antes de começar

A maneira na qual você define as propriedades de uma especificação de ativação depende das interfaces de administração fornecidas por seu servidor de aplicativos Esta tarefa supõe que você esteja usando o WebSphere Application Server versão 7 ou posterior como seu servidor de aplicativos e o IBM WebSphere MQ como seu provedor de sistemas de mensagens.

Nota:

O modo não ASF não pode ser selecionado em sistemas z/OS ..

Sobre esta tarefa

As propriedades de uma especificação de ativação determinam como um bean da unidade de mensagens (MDB) recebe mensagens JMS de uma fila IBM WebSphere MQ . Para configurar o modo não ASF, defina as propriedades de uma ou mais especificações de ativação

Há várias configurações do IBM WebSphere MQ que podem ser usadas no modo não ASF Com as seguintes configurações, cada encadeamento usa uma conexão de rede física separada:

- Um gerenciador de filas IBM WebSphere MQ Versão 7.x , usando um connection factory que possui a propriedade da versão do Provedor configurada como 6.
- Um gerenciador de filas IBM WebSphere MQ Versão 7.x , usando um connection factory que tem a propriedade da versão do Provedor configurada como 7 ou não especificada, conectando por um canal do IBM WebSphere MQ que tem o parâmetro **SHARECNV** (conversas de compartilhamento) configurado como 0.

Para configurar não ASF, configure a propriedade ActivationSpec **NON.ASF.RECEIVE.TIMEOUT** para um número inteiro positivo, que indica que a entrega não ASF é usada. O valor é o tempo, em milissegundos, que uma solicitação get aguarda por mensagens que podem ainda não ter chegado (um get com chamada de espera). O valor padrão, 0, indica que a entrega ASF é usada. Para obter detalhes adicionais, consulte **Propriedades Customizadas do Serviço de Listener de Mensagens**

Esse parâmetro será válido apenas quando o aplicativo estiver em execução no WebSphere Application Server versão 7 ou mais recente

Procedimento

1. Inicie o console administrativo do WebSphere Application Server.
2. Exiba a página de configurações do serviço listener:
   a) Na área de janela de navegação, selecione **Servidores> Tipos de Servidor> WebSphere**
   b) Na área de janela de conteúdo, clique no nome do servidor de aplicativos
   c) Em **Comunicações**clique em **Sistema de mensagens> Serviço de listener de mensagens**.
3. Configure a propriedade customizada **NON.ASF.RECEIVE.TIMEOUT** como uma propriedade Customizada do serviço de listener de mensagens
   a) Clique **Propriedades personalizadas**.
   b) Clique em **Novo**.
   c) Insira o nome da propriedade, **NON.ASF.RECEIVE.TIMEOUT**, no campo **Nome** .
   d) Insira o valor necessário no campo **Valor** .
   e) Clique em **OK**.
4. Salve suas mudanças na configuração principal.
5. Para ativar a configuração alterada, pare e reinicie o servidor de aplicativos.

Resultados

Você configurou as propriedades do serviço de listener de mensagem para o WebSphere Application Server para usar o modo não ASF

Nota: Ao usar o modo não ASF, deve-se assegurar que você permita um período de tempo suficiente para que o processamento seja concluído antes que o tempo limite total do tempo de vida da transação seja atingido, para evitar tempos limites indesejados da transação. Para obter detalhes adicionais, consulte **NON.ASF.RECEIVE.TIMEOUT** na documentação do produto WebSphere Application Server .

Conceitos relacionados

“Modo ASF e não ASF” na página 778
O modo Application Server Facilities (ASF) é o método padrão pelo qual o serviço listener de mensagens no WebSphere Application Server processa mensagens.

Configurando o adaptador de recursos para comunicação de entrada

Para configurar a comunicação de entrada, defina as propriedades de um ou mais objetos ActivationSpec.

Informações relacionadas

Beans Orientados a Mensagens

Serviço de listener de mensagens

Processamento de mensagens no modo ASF e no modo não ASF

Como mensagens são processadas no modo não ASF

Limitações do adaptador de recursos IBM WebSphere MQ

Ao usar o adaptador de recursos do IBM WebSphere MQ, alguns recursos do IBM WebSphere MQ estão indisponíveis ou limitados.

O adaptador de recursos do IBM WebSphere MQ tem as limitações a seguir:

- O adaptador de recursos IBM WebSphere MQ é suportado em todas as plataformas IBM WebSphere MQ , exceto z/OS.
- O adaptador de recursos do IBM WebSphere MQ não suporta conexões em tempo real com um broker Ele suporta apenas conexões com um gerenciador de filas do IBM WebSphere MQ no modo de cliente ou de ligação
- O adaptador de recursos IBM WebSphere MQ não suporta programas de saída de canal gravados em linguagens diferentes de Java.
- Enquanto um servidor de aplicativos está em execução, o valor da propriedade sslFipsRequired deve ser true para todos os recursos de JCA ou false para todos os recursos de JCA. Esse é um requisito mesmo se os recursos de JCA não forem usados simultaneamente. Se a propriedade sslFipsRequired tiver valores diferentes para diferentes recursos de JCA, o IBM WebSphere MQ emite o código de razão MQRC_UNSUPPORTED_CIPHER_SUITE, mesmo se uma conexão SSL não estiver sendo usada.
- Não é possível especificar mais de um keystore para um servidor de aplicativos. Se forem feitas conexões com mais de um gerenciador de filas, todas as conexões devem usar o mesmo keystore. Essa limitação não se aplica ao WebSphere Application Server.
- Se você usar uma tabela de definição de canal de cliente (CCDT) com mais de uma definição de canal de conexão de cliente apropriada, no caso de uma falha, o adaptador de recursos pode selecionar uma definição de canal diferente e, portanto, um gerenciador de filas diferente na CCDT, o que causaria problemas para a recuperação de transação. O adaptador de recursos não toma nenhuma ação para evitar que essa configuração seja usada e é sua responsabilidade evitar configurações que possam causar problemas para a recuperação da transação.
- A funcionalidade de nova tentativa de conexão introduzida no IBM WebSphere MQ Version 7.0.1 não é suportada para conexões de saída ao executar em um contêiner JEE (EJB/Servlet). A nova tentativa de conexão não é suportada para JMS de saída quando o adaptador é usado em um contexto de contêiner JEE, independentemente da configuração de transação ou para uso não transacional.

Tarefas relacionadas

Especificando que Apenas CipherSpecs Certificados por FIPS São Usados no Tempo de Execução no Cliente de MQI

Referências relacionadas

Federal Information Processing Standards (FIPS) para UNIX, Linux e Windows

Configuração de pós-instalação para classes WebSphere MQ para aplicativos JMS

Este tópico informa quais autoridades WebSphere MQ classes para aplicativos JMS precisam para acessar os recursos de um gerenciador de filas. Ele também introduz os modos de conexão e descreve como configurar um gerenciador de filas para que os aplicativos possam se conectar no modo cliente.

Lembre-se de verificar o arquivo leia-me do WebSphere MQ Ele pode conter informações que suplantam as informações neste tópico.

Objetos usados pelo JMS que requerem autorização para usuários não privilegiados

Os usuários não privilegiados precisam de autorização concedida para acessar as filas usadas pelo JMS Cada aplicativo JMS precisa de autorização para o gerenciador de filas com o qual ele funciona

Para obter detalhes sobre o controle de acesso no IBM WebSphere MQ, consulte Configurando a segurança em Windows, UNIX and Linux sistemas

As classes do WebSphere MQ para aplicativos JMS precisam da autoridade `connect` e `inq` para o gerenciador de filas. É possível configurar autorizações apropriadas usando o comando de controle **`setmqaut`** , por exemplo:

```
setmqaut -m QM1 -t qmgr -g jmsappsgroup +connect +inq
```

Para o domínio ponto a ponto, as seguintes autoridades são necessárias:

- As filas usadas pelos objetos MessageProducer precisam da autoridade `put`.
- As filas usadas por objetos MessageConsumer e QueueBrowser precisam das autoridades `get`, `inq` e `browse`.
- O método QueueSession.createTemporaryQueue() precisa de acesso à fila de modelo especificada pela propriedade TEMPMODEL do objeto QueueConnectionFactory. Por padrão, esta fila de modelo é SYSTEM.TEMP.MODEL.QUEUE.

Se qualquer uma destas filas for filas de alias, suas filas de destino irão requerer a autoridade de consulta. Se a fila de destino for uma fila de cluster, ela também irá requerer a autoridade de procura.

Para o domínio de publicação / assinatura, as filas a seguir serão usadas se as classes do WebSphere MQ para JMS estiverem se conectando a um gerenciador de filas do IBM WebSphere MQ no modo de migração do provedor de sistemas de mensagens do IBM WebSphere MQ :

- SYSTEM.JMS.ADMIN.QUEUE
- SYSTEM.JMS.REPORT.QUEUE
- SYSTEM.JMS.MODEL.QUEUE
- SYSTEM.JMS.PS.STATUS.QUEUE
- SYSTEM.JMS.ND.SUBSCRIBER.QUEUE
- SYSTEM.JMS.D.SUBSCRIBER.QUEUE
- SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
- SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE
- SYSTEM.BROKER.CONTROL.QUEUE

Para obter informações adicionais sobre o modo de migração do provedor de sistemas de mensagens do IBM WebSphere MQ, consulte Quando usar **`PROVIDERVERSION`**

Além disso, se as classes do WebSphere MQ para JMS estiverem se conectando a um gerenciador de filas neste modo, qualquer aplicativo que publica mensagens precisará de acesso à fila de fluxo especificada pelo factory TopicConnectionou objeto do tópico. Por padrão, esta fila é SYSTEM.BROKER.DEFAULT.STREAM.

Se você usar ConnectionConsumer, IBM WebSphere MQ Resource Adapter ou o provedor de sistemas de mensagens WebSphere Application Server IBM WebSphere MQ , poderá ser necessária autorização adicional.

As filas a serem lidas pelo ConnectionConsumer devem ter as autoridades `get` , `inq`e `browse` . A fila de mensagens não entregues do sistema e qualquer fila de backout-requeue ou fila de relatório usada pelo ConnectionConsumer deve ter as autoridades `put` e `passall`.

Quando um aplicativo usa o modo normal do provedor de sistemas de mensagens do WebSphere MQ para executar o sistema de mensagens de publicação / assinatura, o aplicativo usa a funcionalidade de publicação / assinatura integrada fornecida pelo gerenciador de filas. Consulte Segurança de publicação/ assinatura para obter informações sobre como proteger os tópicos e as filas usados.

Modos de Conexão para Classes do WebSphere MQ para JMS

Uma classe WebSphere MQ para aplicativo JMS pode se conectar a um gerenciador de filas no modo cliente ou de ligação. No modo cliente, as classes WebSphere MQ para JMS se conectam ao gerenciador de filas sobre TCP/IP. No modo de ligação, as classes WebSphere MQ para JMS se conectam diretamente ao gerenciador de filas usando a Java Native Interface (JNI).

Um aplicativo em execução no WebSphere Application Server on z/OS pode se conectar a um gerenciador de fila em ligações ou no modo cliente, mas um aplicativo em execução em qualquer outro ambiente no z/OS pode se conectar a um gerenciador de filas apenas em modo de ligação Um aplicativo em execução em qualquer outra plataforma pode se conectar a um gerenciador de filas tanto no modo de ligações como de cliente.

É possível usar a versão atual ou anterior suportada de classes do WebSphere MQ para JMS com um gerenciador de filas atual e é possível usar uma versão atual ou anterior suportada do gerenciador de filas com a versão atual de classes do WebSphere MQ para JMS. Se diferentes versões forem misturadas, a função será limitada ao nível da versão anterior.

As seções a seguir descrevem cada um dos modos de conexão em mais detalhes.

Modo do cliente

Para se conectar a um gerenciador de fila no modo cliente, um WebSphere MQ classes para aplicativo JMS pode ser executado no mesmo sistema no qual o gerenciador de filas está em execução ou em um sistema diferente. Em cada caso, as classes WebSphere MQ para JMS se conectam ao gerenciador de filas sobre TCP/IP.

Modo de ligações

Para conectar-se a um gerenciador de filas no modo de ligações, um WebSphere MQ classes para aplicativo JMS deve ser executado no mesmo sistema no qual o gerenciador de filas está em execução.

As classes WebSphere MQ para JMS se conectam diretamente ao gerenciador de filas usando a Java Native Interface (JNI). Para usar o transporte de ligações, as classes WebSphere MQ para JMS devem ser executadas em um ambiente que tenha acesso às bibliotecas do WebSphere MQ Java Native Interface; consulte “Configurando as bibliotecas do Java Native Interface (JNI)” na página 733 para obter informações adicionais.

As classes do WebSphere MQ para JMS suportam os seguintes valores para *ConnectOption* :

- MQCNO_FASTPATH_BINDING
- MQCNO_STANDARD_BINDING
- MQCNO_SHARED_BINDING
- MQCNO_ISOLATED_BINDING
- MQCNO_SERIALIZE_CONN_TAG_QSG
- MQCNO_RESTRICT_CONN_TAG_QSG
- MQCNO_SERIALIZE_CONN_TAG_Q_MGR
- MQCNO_RESTRICT_CONN_TAG_Q_MGR

Para alterar as opções de conexão usadas pelas classes do WebSphere MQ para JMS, modifique a propriedade da Connection Factory CONNOPT..

Para obter informações adicionais sobre opções de conexão, consulte “Conectando-se a um gerenciador de filas usando a chamada MQCONNX” na página 212

Para usar o transporte de ligações, o Java Runtime Environment que está sendo usado deve suportar o Coded Character Set Identifier (CCSID) do gerenciador de fila ao qual as classes WebSphere MQ para JMS estão se conectando.

Detalhes sobre como determinar quais CCSIDs são suportados por um Java Runtime Environment podem ser localizados em WebSphere MQ FDC com ID de Análise 21 gerado ao usar as classes WebSphere MQ V7 para Java ou WebSphere MQ V7 classes para JMS .

Ligações, em seguida, modo cliente

Esse é o padrão. Ao conectar-se a um gerenciador de filas nesse modo, uma classe WebSphere MQ para o aplicativo JMS tentará se conectar no modo de ligações, o que requer que o gerenciador de filas resida na mesma máquina que o aplicativo Se a conexão for malsucedida, o aplicativo tentará se conectar no modo cliente, permitindo que o gerenciador de filas resida localmente na mesma máquina que o aplicativo ou remotamente.

Configurando seu gerenciador de filas para que as classes do WebSphere MQ para aplicativos JMS possam se conectar no modo cliente

Para configurar seu gerenciador de filas para que as classes do WebSphere MQ para aplicativos JMS possam se conectar no modo cliente, você deve criar uma definição de canal de conexão do servidor e iniciar um listener.

No z/OS, o recurso de Anexo do Cliente deve ser instalado

Criando uma definição de canal de conexão do servidor

Em todas as plataformas, é possível usar o comando DEFINE CHANNEL do MQSC para criar uma definição de canal de conexão do servidor. Consulte o exemplo a seguir:

```
DEFINE CHANNEL(JAVA.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP)
```

No IBM i, é possível usar o comando de CL CRTMQMCHL em vez disso, como no exemplo a seguir:

```
CRTMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN)
                                TRPTYPE(*TCP)
                                MQMNAME(QMGRNAME)
```

Nesse comando, *QMGRNAME* é o nome do gerenciador de filas.

Também é possível criar uma definição do canal de conexão do servidor usando o IBM WebSphere MQ Explorer, que é executado no Linux e no Windowsou as operações e painéis de controle no z/OS.

O nome do canal (JAVA.CHANNEL nos exemplos anteriores) deve ser o mesmo que o nome do canal especificado pela propriedade CHANNEL do connection factory que seu aplicativo usa para se conectar ao gerenciador de filas. O valor padrão da propriedade CHANNEL é SYSTEM.DEF.SVRCONN.

Iniciando um listener

Deve-se iniciar um listener para o gerenciador de filas se um ainda não estiver iniciado.

Em todas as plataformas, é possível usar o comando MQSC START LISTENER para iniciar um listener, mas, exceto no z/OS, deve-se primeiro criar um objeto do listener usando o comando MQSC DEFINE LISTENER. Consulte o exemplo a seguir:

```
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) PORT(1414)
START LISTENER(LISTENER.TCP)
```

Nos sistemas UNIX, Linux e Windows, também é possível usar o comando de controle **runmqlsr** para iniciar um listener, como no exemplo a seguir:

```
runmqlsr -t tcp -p 1414 -m QMgrName
```

Nesse comando, *QMgrName* é o nome do gerenciador de filas.

Também é possível iniciar um listener usando o WebSphere MQ Explorer, que é executado no Linux e no Windowsou nas operações e painéis de controle no z/OS.

O número da porta na qual o listener está atendendo deve ser o mesmo que o número da porta especificado pela propriedade PORT do connection factory que seu aplicativo usa para se conectar ao gerenciador de filas. O valor padrão da propriedade PORT é 1414.

O teste de verificação de instalação ponto a ponto para classes do WebSphere MQ para JMS

Um programa de teste de verificação de instalação ponto a ponto (IVT) é fornecido com classes WebSphere MQ para JMS. O programa se conecta a um gerenciador de filas no modo de ligações ou de cliente, envia uma mensagem à fila chamada SYSTEM.DEFAULT.LOCAL.QUEUE e, em seguida, recebe a mensagem da fila. O programa pode criar e configurar todos os objetos que requer dinamicamente no tempo de execução ou pode usar JNDI para recuperar objetos administrados a partir de um serviço de diretório.

Execute o teste de verificação de instalação sem antes usar JNDI porque o teste é autocontido e não requer o uso de um serviço de diretório. Para obter uma descrição de objetos administradas, consulte “tipos de objeto JMS” na página 951

O teste de verificação de instalação ponto a ponto sem usar JNDI

Neste teste, o programa IVT cria e configura todos os objetos que requer dinamicamente no tempo de execução e não usa JNDI.

Um script é fornecido para executar o programa IVT. O script é chamado IVTRun em sistemas UNIX and Linux e IVTRun.bat no Windowse está no subdiretório `bin` das classes WebSphere MQ para o diretório de instalação JMS.

Para executar o teste no modo de ligações, insira o comando a seguir:

```
IVTRun -nojndi [-m qmgr] [-v providerVersion] [-t]
```

Para executar o teste no modo cliente, primeiro configure o gerenciador de filas, conforme descrito em “Preparando e executando os programas de amostra” na página 111 Observe que o canal a ser usado é padronizado como **SYSTEM.DEF.SVRCONN** e a fila a ser usada é **SYSTEM.DEFAULT.LOCAL.QUEUE**, em seguida, insira o comando a seguir:

```
IVTRun -nojndi -client -m qmgr -host hostname [-port port] [-channel channel]
       [-v providerVersion] [-ccsid ccsid] [-t]
```

Nenhum script equivalente é fornecido em sistemas z/OS , mas é possível executar o IVT no modo de ligações chamando a classe Java diretamente, usando o comando a seguir:

```
java com.ibm.mq.jms.MQJMSIVT -nojndi [-m qmgr] [-v providerVersion] [-t]
```

O caminho de classe deve conter com.ibm.mqjms.jar.

O parâmetros nos comandos têm os significados a seguir:

-m *qmgr*
O nome do gerenciador de filas ao qual o programa IVT se conecta. Se você executar o teste no modo de ligações e omitir esse parâmetro, o programa IVT se conecta ao gerenciador de filas padrão.

-host *hostname*
O nome do host ou o endereço IP do sistema em que o gerenciador de filas está em execução.

-port *port*
O número da porta na qual o listener do gerenciador de filas está atendendo. O valor padrão é 1414.

-channel *channel*
O nome do canal MQI que o programa IVT usa para se conectar ao gerenciador de filas. O valor padrão é `SYSTEM.DEF.SVRCONN`.

-v *providerVersion*
O nível de liberação do gerenciador de filas ao qual o programa IVT espera se conectar.

Esse parâmetro é usado para configurar a propriedade PROVIDERVERSION de um objeto MQQueueConnectionFactory e tem os mesmos valores válidos que os da propriedade PROVIDERVERSION. Portanto, para obter mais informações sobre esse parâmetro, incluindo seus valores válidos, consulte a descrição da propriedade PROVIDERVERSION em Propriedades de IBM WebSphere MQ classes for JMS objetos

O valor padrão é `unspecified`.

-ccsid ***ccsid***
O identificador de conjunto de caracteres codificados (CCSID) ou página de códigos a ser usado pela conexão. O valor padrão é 819.

-t
O rastreio está ativado. Por padrão, o rastreio está desativado.

Um teste bem-sucedido produz uma saída semelhante à saída de amostra a seguir:

```
5724-H72, 5655-R36, 5724-L26, 5655-L82 (c) Copyright IBM Corp. 2008, 2024. All
Rights Reserved.
Websphere MQ classes for Java(tm) Message Service 7.0
Installation Verification Test

Creating a QueueConnectionFactory
Creating a Connection
Creating a Session
Creating a Queue
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message
  JMSMessage class: jms_text
  JMSType:          null
  JMSDeliveryMode:  2
  JMSExpiration:    0
  JMSPriority:      4
  JMSMessageID:     ID:414d5120514d5f6d627720202020202001edb14620005e03
  JMSTimestamp:     1187170264000
  JMSCorrelationID: null
  JMSDestination:   queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
  JMSReplyTo:       null
  JMSRedelivered:   false
    JMSXUserID: mwhite
    JMS_IBM_Encoding: 273
    JMS_IBM_PutApplType: 28
    JMSXAppID: WebSphere MQ Client for Java
    JMSXDeliveryCount: 1
    JMS_IBM_PutDate: 20070815
    JMS_IBM_PutTime: 09310400
    JMS_IBM_Format: MQSTR
    JMS_IBM_MsgType: 8
A simple text message from the MQJMSIVT
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
IVT completed OK
IVT finished
```

O teste de verificação de instalação ponto a ponto usando JNDI

Neste teste, o programa IVT usa JNDI para recuperar objetos administrados de um serviço de diretório.

Antes que seja possível executar o teste, deve-se configurar um serviço de diretório baseado em um servidor Lightweight Directory Access Protocol (LDAP) ou no sistema de arquivos local. Deve-se também configurar a ferramenta de administração JMS do WebSphere MQ para que ela possa usar o

serviço de diretório para armazenar objetos administrado Para obter mais informações sobre esses pré-requisitos, consulte “Pré-requisitos para classes do WebSphere MQ para JMS” na página 725. Para obter informações sobre como configurar a ferramenta de administração JMS do WebSphere MQ , consulte “Configurando a Ferramenta de Administração JMS” na página 947..

O programa IVT deve ser capaz de usar JNDI para recuperar um objeto MQQueueConnectionFactory e um objeto MQQueue do serviço de diretório. Um script é fornecido para criar esses objetos administrados para você. O script é chamado IVTSetup em sistemas UNIX and Linux e IVTSetup.bat em Windowse está no subdiretório `bin` do WebSphere MQ classes para o diretório de instalação JMS. Para executar o script, insira o comando a seguir:

```
IVTSetup
```

O script chama a ferramenta de administração JMS do WebSphere MQ para criar os objetos administrados.

O objeto de MQQueueConnectionFactory é ligado com o nome ivtQCF e é criado com os valores padrão para todas as suas propriedades, o que significa que o programa IVT executa no modo de ligações e se conecta ao gerenciador de filas padrão. Se desejar que o programa IVT seja executado no modo cliente ou se conecte a um gerenciador de filas diferente do gerenciador de filas padrão, você deve usar a ferramenta de administração do WebSphere MQ JMS ou WebSphere MQ Explorer para alterar as propriedades apropriadas do objeto do factory MQQueueConnection. Para obter informações sobre como usar a ferramenta de administração JMS WebSphere MQ , consulte “Usando a ferramenta de administração JMS do WebSphere MQ” na página 946. Para obter informações sobre como usar o WebSphere MQ Explorer, consulte a ajuda fornecida com o WebSphere MQ Explorer.

O objeto MQQueue é ligado com o nome ivtQ e é criado com os valores padrão para todas as suas propriedades, exceto para a propriedade QUEUE, que tem o valor SYSTEM.DEFAULT.LOCAL.QUEUE.

Quando tiver criado os objetos administrados, será possível executar o programa IVT. Para executar o teste usando JNDI, insira o comando a seguir:

```
IVTRun -url "providerURL" [-icf initCtxFact] [-t]
```

Os parâmetros no comando têm os significados a seguir:

-url " *providerURL* "
O localizador uniforme de recursos (URL) do serviço de diretório. A URL pode ter um dos formatos a seguir:

- `ldap://`*`hostname`*`/`*`contextName`*, para um serviço de diretório com base em um servidor LDAP
- `file:/`*`directoryPath`*, para um serviço de diretório com base no sistema de arquivos local

Observe que a URL deve ser colocada entre aspas (").

-icf *initCtxFact*
O nome da classe do factory de contexto inicial, que deve ser um dos valores a seguir:

- `com.sun.jndi.ldap.LdapCtxFactory`, para um serviço de diretório com base em um servidor LDAP. Esse é o valor-padrão.
- `com.sun.jndi.fscontext.RefFSContextFactory`, para um serviço de diretório com base no sistema de arquivos local.

-t
O rastreio está ativado. Por padrão, o rastreio está desativado.

Um teste bem-sucedido produz uma saída semelhante àquela de um teste bem-sucedido sem usar JNDI. A diferença principal é que a saída indica que o teste está usando JNDI para recuperar um objeto MQQueueConnectionFactory e um objeto MQQueue.

Embora não seja estritamente necessário, uma boa prática é limpar após o teste excluindo os objetos administrados criados pelo script IVTSetup. Um script é fornecido para esse propósito. O script é chamado IVTTidy em sistemas UNIX and Linux e IVTTidy.bat em Windowse está no subdiretório `bin` das classes WebSphere MQ para o diretório de instalação JMS.

Determinação de problema para o teste de verificação de instalação ponto a ponto

O teste de verificação de instalação pode falhar pelas razões a seguir:

- Se o programa IVT grava uma mensagem indicando que não pode localizar uma classe, verifique se o caminho de classe está configurado corretamente, conforme descrito em "Variáveis de ambiente usadas por classes IBM WebSphere MQ para JMS" na página 731.
- O teste pode falhar com a mensagem a seguir:

```
Failed to connect to queue manager 'qmgr' with connection mode 'connMode'
and host name 'hostname'
```

  e um código de razão associado de 2059. As variáveis na mensagem têm os significados a seguir:

  qmgr
  O nome do gerenciador de filas ao qual o programa IVT está tentando se conectar. Essa inserção de mensagem estará em branco se o programa IVT estiver tentando se conectar com o gerenciador de filas padrão no modo de ligações.

  connMode
  O modo de conexão, que é `Bindings` ou `Client`.

  HOSTNAME
  O nome do host ou o endereço IP do sistema em que o gerenciador de filas está em execução.

  Esta mensagem significa que o gerenciador de filas ao qual o programa IVT está tentando se conectar não está disponível. Verifique se o gerenciador de filas está em execução e, se o programa IVT estiver tentando se conectar com o gerenciador de filas padrão, certifique-se de que o gerenciador de filas esteja definido como o gerenciador de filas padrão para seu sistema.
- O teste pode falhar com a mensagem a seguir:

```
Failed to open MQ queue 'SYSTEM.DEFAULT.LOCAL.QUEUE'
```

  Esta mensagem significa que a fila SYSTEM.DEFAULT.LOCAL.QUEUE não existe no gerenciador de filas ao qual o programa IVT está conectado. Como alternativa, se a fila existir, o programa IVT não poderá abrir a fila porque não está ativado para efetuar put e get de mensagens. Verifique se a fila existe e se está ativada para efetuar out e get de mensagens.
- O teste pode falhar com a mensagem a seguir:

```
Unable to bind to object
```

  Essa mensagem significa que há uma conexão com o servidor LDAP, mas que o servidor LDAP não está configurado corretamente. O servidor LDAP não está configurado para armazenar objetos Java ou as permissões nos objetos ou sufixo não estão corretos. Para obter mais ajuda nessa situação, consulte a documentação de seu servidor LDAP.
- O teste pode falhar com a mensagem a seguir:

```
The security authentication was not valid that was supplied for
QueueManager 'qmgr' with connection mode 'Client' and host name 'hostname'
```

  Esta mensagem significa que o gerenciador de filas não é configurado corretamente para aceitar uma conexão do cliente a partir de seu sistema. Consulte "Preparando e executando os programas de amostra" na página 111 para obter detalhes.

O teste de verificação de instalação de publicação / assinatura para classes WebSphere MQ para JMS

Um programa de teste de verificação de instalação (IVT) de publicação / assinatura é fornecido com classes WebSphere MQ para JMS. O programa se conecta a um gerenciador de filas no modo de ligações ou do cliente, assina um tópico, publica uma mensagem sobre o tópico e, em seguida, recebe a mensagem que acaba de publicar. O programa pode criar e configurar todos os objetos que requer

dinamicamente no tempo de execução ou pode usar JNDI para recuperar objetos administrados a partir de um serviço de diretório.

Execute o teste de verificação de instalação sem antes usar JNDI porque o teste é autocontido e não requer o uso de um serviço de diretório. Para obter uma descrição de objetos administradas, consulte "tipos de objeto JMS" na página 951

O teste de verificação da instalação de publicar/assinar sem usar JNDI

Neste teste, o programa IVT cria e configura todos os objetos que requer dinamicamente no tempo de execução e não usa JNDI.

Um script é fornecido para executar o programa IVT. O script é chamado PSIVTRun em sistemas UNIX and Linux e PSIVTRun.bat em Windowse está no subdiretório `bin` das classes WebSphere MQ para o diretório de instalação JMS.

Para executar o teste no modo de ligações, insira o comando a seguir:

```
PSIVTRun -nojndi [-m qmgr] [-bqm brokerQmgr] [-v providerVersion] [-t]
```

Para executar o teste no modo cliente, primeiro, configure o gerenciador de filas conforme descrito em "Preparando e executando os programas de amostra" na página 111, observando que o canal a ser usado é por padrão SYSTEM.DEF.SVRCONN, em seguida, insira o comando a seguir:

```
PSIVTRun -nojndi -client -m qmgr -host hostname [-port port] [-channel channel]
         [-bqm brokerQmgr] [-v providerVersion] [-ccsid ccsid] [-t]
```

O parâmetros nos comandos têm os significados a seguir:

-m *qmgr*
O nome do gerenciador de filas ao qual o programa IVT se conecta. Se você executar o teste no modo de ligações e omitir esse parâmetro, o programa IVT se conecta ao gerenciador de filas padrão.

-host *hostname*
O nome do host ou o endereço IP do sistema em que o gerenciador de filas está em execução.

-port *port*
O número da porta na qual o listener do gerenciador de filas está atendendo. O valor padrão é 1414.

-channel *channel*
O nome do canal MQI que o programa IVT usa para se conectar ao gerenciador de filas. O valor padrão é `SYSTEM.DEF.SVRCONN`.

-bqm *brokerQmgr*
O nome do gerenciador de filas no qual o broker está em execução. O valor padrão é o nome do gerenciador de filas ao qual o programa IVT se conecta.

Esse parâmetro será relevante apenas se o parâmetro -v especificar um número de versão do gerenciador de filas menor que 7 e você estiver usando o WebSphere Event Broker ou WebSphere Message Broker como o broker de publicação / assinatura.

-v *providerVersion*
O nível de liberação do gerenciador de filas ao qual o programa IVT espera se conectar.

Esse parâmetro é usado para configurar a propriedade PROVIDERVERSION de um objeto MQTopicConnectionFactory e tem os mesmos valores válidos que os da propriedade PROVIDERVERSION. Portanto, para obter mais informações sobre esse parâmetro, incluindo seus valores válidos, consulte a descrição da propriedade PROVIDERVERSION em Propriedades de IBM WebSphere MQ classes for JMS objetos

O valor padrão é `unspecified`.

-ccsid *ccsid*
O identificador de conjunto de caracteres codificados (CCSID) ou página de códigos a ser usado pela conexão. O valor padrão é 819.

-t
O rastreio está ativado. Por padrão, o rastreio está desativado.

Um teste bem-sucedido produz uma saída semelhante à saída de amostra a seguir:

```
5724-H72, 5655-R36, 5724-L26, 5655-L82 (c) Copyright IBM Corp. 2008, 2024. All
Rights Reserved.
Websphere MQ classes for Java(tm) Message Service 7.0
Publish/Subscribe Installation Verification Test

Creating a TopicConnectionFactory
Creating a Connection
Creating a Session
Creating a Topic
Creating a TopicPublisher
Creating a TopicSubscriber
Creating a TextMessage
Adding text
Publishing the message to topic://MQJMS/PSIVT/Information
Waiting for a message to arrive [5 secs max]...

Got message:
  JMSMessage class: jms_text
  JMSType:          null
  JMSDeliveryMode:  2
  JMSExpiration:    0
  JMSPriority:      4
  JMSMessageID:     ID:414d5120514d5f6d627720202020202001edb14620006706
  JMSTimestamp:     1187182520203
  JMSCorrelationID: ID:414d5120514d5f6d627720202020202001edb14620006704
  JMSDestination:   topic://MQJMS/PSIVT/Information
  JMSReplyTo:       null
  JMSRedelivered:   false
    JMSXUserID: mwhite
    JMS_IBM_Encoding: 273
    JMS_IBM_PutApplType: 26
    JMSXAppID: QM_mbw
    JMSXDeliveryCount: 1
    JMS_IBM_PutDate: 20070815
    JMS_IBM_ConnectionID: 414D5143514D5F6D627720202020202001EDB14620006601
    JMS_IBM_PutTime: 12552020
    JMS_IBM_Format: MQSTR
    JMS_IBM_MsgType: 8
A simple text message from the MQJMSPSIVT program
Reply string equals original string
Closing TopicSubscriber
Closing TopicPublisher
Closing Session
Closing Connection
PSIVT finished
```

O teste de verificação de instalação de publicar/assinar usando JNDI

Neste teste, o programa IVT usa JNDI para recuperar objetos administrados de um serviço de diretório.

Antes que seja possível executar o teste, deve-se configurar um serviço de diretório baseado em um servidor Lightweight Directory Access Protocol (LDAP) ou no sistema de arquivos local. Deve-se também configurar a ferramenta de administração JMS do WebSphere MQ para que ela possa usar o serviço de diretório para armazenar objetos administrado Para obter mais informações sobre esses pré-requisitos, consulte “Pré-requisitos para classes do WebSphere MQ para JMS” na página 725. Para obter informações sobre como configurar a ferramenta de administração JMS do WebSphere MQ , consulte “Configurando a Ferramenta de Administração JMS” na página 947..

O programa IVT deve ser capaz de usar JNDI para recuperar um objeto MQTopicConnectionFactory e um objeto MQTopic do serviço de diretório. Um script é fornecido para criar esses objetos administrados para você. O script é chamado IVTSetup em sistemas UNIX and Linux e IVTSetup.bat em Windowse está no

subdiretório `bin` do WebSphere MQ classes para o diretório de instalação JMS. Para executar o script, insira o comando a seguir:

```
IVTSetup
```

O script chama a ferramenta de administração JMS do WebSphere MQ para criar os objetos administrados.

O objeto MQTopicConnectionFactory é ligado com o nome ivtTCF e é criado com os valores padrão para todas as suas propriedades, o que significa que o programa IVT é executado no modo de ligações, se conecta ao gerenciador de filas padrão e usa a função de publicar/assinar integrada. Se você desejar que o programa IVT seja executado no modo cliente, conecte-se a um gerenciador de filas diferente do gerenciador de filas padrão ou use o WebSphere Event Broker ou WebSphere Message Broker em vez da função de publicação / assinatura integrada, deve-se usar a ferramenta de administração JMS do WebSphere MQ ou o Explorer do WebSphere MQ para alterar as propriedades apropriadas do objeto do Factory MQTopicConnection. Para obter informações sobre como usar a ferramenta de administração JMS WebSphere MQ , consulte “Usando a ferramenta de administração JMS do WebSphere MQ” na página 946. Para obter informações sobre como usar o WebSphere MQ Explorer, consulte a ajuda fornecida com o WebSphere MQ Explorer.

O objeto MQTopic é ligado ao nome ivtT e é criado com os valores padrão para todas as suas propriedades, exceto para a propriedade TOPIC, que tem o valor MQJMS/PSIVT/Information.

Quando tiver criado os objetos administrados, será possível executar o programa IVT. Para executar o teste usando JNDI, insira o comando a seguir:

```
PSIVTRun -url "providerURL" [-icf initCtxFact] [-t]
```

Os parâmetros no comando têm os significados a seguir:

-url " *providerURL* "
O localizador uniforme de recursos (URL) do serviço de diretório. A URL pode ter um dos formatos a seguir:

- `ldap://hostname/contextName`, para um serviço de diretório com base em um servidor LDAP
- `file:/directoryPath`, para um serviço de diretório com base no sistema de arquivos local

Observe que a URL deve ser colocada entre aspas (").

-icf *initCtxFact*
O nome da classe do factory de contexto inicial, que deve ser um dos valores a seguir:

- `com.sun.jndi.ldap.LdapCtxFactory`, para um serviço de diretório com base em um servidor LDAP. Esse é o valor-padrão.
- `com.sun.jndi.fscontext.RefFSContextFactory`, para um serviço de diretório com base no sistema de arquivos local.

-t
O rastreio está ativado. Por padrão, o rastreio está desativado.

Um teste bem-sucedido produz uma saída semelhante àquela de um teste bem-sucedido sem usar JNDI. A diferença principal é que a saída indica que o teste está usando JNDI para recuperar um objeto MQTopicConnectionFactory e um objeto MQTopic.

Embora não seja estritamente necessário, uma boa prática é limpar após o teste excluindo os objetos administrados criados pelo script IVTSetup. Um script é fornecido para esse propósito. O script é chamado IVTTidy em sistemas UNIX and Linux e IVTTidy.bat em Windowse está no subdiretório `bin` das classes WebSphere MQ para o diretório de instalação JMS.

Determinação de problema para o teste de verificação de instalação de publicar/assinar

O teste de verificação de instalação pode falhar pelas razões a seguir:

- Se o programa IVT grava uma mensagem indicando que não pode localizar uma classe, verifique se o caminho de classe está configurado corretamente, conforme descrito em "Variáveis de ambiente usadas por classes IBM WebSphere MQ para JMS" na página 731.
- O teste pode falhar com a mensagem a seguir:

```
Failed to connect to queue manager 'qmgr' with
connection mode 'connMode' and host name 'hostname'
```

  e um código de razão associado de 2059. As variáveis na mensagem têm os significados a seguir:

  qmgr
  O nome do gerenciador de filas ao qual o programa IVT está tentando se conectar. Essa inserção de mensagem estará em branco se o programa IVT estiver tentando se conectar com o gerenciador de filas padrão no modo de ligações.

  connMode
  O modo de conexão, que é `Bindings` ou `Client`.

  HOSTNAME
  O nome do host ou o endereço IP do sistema em que o gerenciador de filas está em execução.

  Esta mensagem significa que o gerenciador de filas ao qual o programa IVT está tentando se conectar não está disponível. Verifique se o gerenciador de filas está em execução e, se o programa IVT estiver tentando se conectar com o gerenciador de filas padrão, certifique-se de que o gerenciador de filas esteja definido como o gerenciador de filas padrão para seu sistema.
- O teste pode falhar com a mensagem a seguir:

```
Unable to bind to object
```

  Essa mensagem significa que há uma conexão com o servidor LDAP, mas que o servidor LDAP não está configurado corretamente. O servidor LDAP não está configurado para armazenar objetos Java ou as permissões nos objetos ou sufixo não estão corretos. Para obter mais ajuda nessa situação, consulte a documentação de seu servidor LDAP.
- O teste pode falhar com a mensagem a seguir:

```
The security authentication was not valid that was supplied for
QueueManager 'qmgr' with connection mode 'Client' and host name 'hostname'
```

  Essa mensagem significa que o gerenciador de filas não está configurado corretamente para aceitar uma conexão do cliente do sistema. Para obter informações adicionais, consulte "Preparando e executando os programas de amostra" na página 111.

O programa de teste de verificação de instalação para o adaptador de recursos WebSphere MQ

O programa IVT é fornecido como um arquivo EAR. Para usar o programa, deve-se implementá-lo e definir alguns objetos como recursos JCA.

O programa de teste de verificação de instalação (IVT) é fornecido como um arquivo EAR (enterprise archive) chamado wmq.jmsra.ivt.ear. Esse arquivo é instalado com as classes WebSphere MQ para JMS no mesmo diretório que o arquivo RAR do adaptador de recursos WebSphere MQ , wmq.jmsra.rar. Para obter informações sobre onde esses arquivos estão instalados, consulte "Instalação do adaptador de recursos do WebSphere MQ" na página 743.

Deve-se implementar o programa IVT em seu servidor de aplicativos. O programa IVT inclui um servlet e um MDB que testa se uma mensagem pode ser enviada e recebida de uma fila do WebSphere MQ . Opcionalmente, é possível usar o programa IVT para verificar se o adaptador de recursos WebSphere MQ foi configurado corretamente para suportar transações distribuídas.

Antes de poder executar o programa IVT, deve-se definir um objeto ConnectionFactory , um objeto Queue e possivelmente um objeto Activation Specification como recursos JCA e assegurar que seu servidor de aplicativos crie objetos JMS a partir dessas definições e os ligue a um namespace JNDI. É possível escolher as propriedades dos objetos, mas o conjunto de propriedades a seguir é um exemplo simples:

objeto ConnectionFactory

```
channel:              SYSTEM.DEF.SVRCONN
hostName:             localhost
port:                 1414
queueManager:         ExampleQM
transportType:        CLIENT
```

Objeto Queue

```
baseQueueManagerName:  ExampleQM
baseQueueName:         TEST.QUEUE
```

Por padrão, o programa IVT espera que um objeto ConnectionFactory seja ligado no namespace JNDI com o nome jms/ivt/IVTCF e um objeto da Fila seja ligado com o nome jms/ivt/IVTQueue. É possível usar diferentes nomes, mas se fizer isso, deverá inserir os nomes dos objetos na página inicial do programa IVT e modificar o arquivo EAR conforme apropriado.

Após ter implementado o programa IVT e o servidor de aplicativos ter criado os objetos JMS e ligado-os ao namespace JNDI, é possível iniciar o programa IVT inserindo uma URL no seguinte formato em seu navegador da web:

```
http://app_server_host:port/WMQ_IVT/
```

em que *app_server_host* é o endereço IP ou nome do host do sistema no qual seu servidor de aplicativos está em execução, e *port* é o número da porta TCP na qual o servidor de aplicativos está atendendo. Aqui está um exemplo:

```
http://localhost:9080/WMQ_IVT/
```

Figura 122 na página 793 mostra a página inicial do programa IVT.

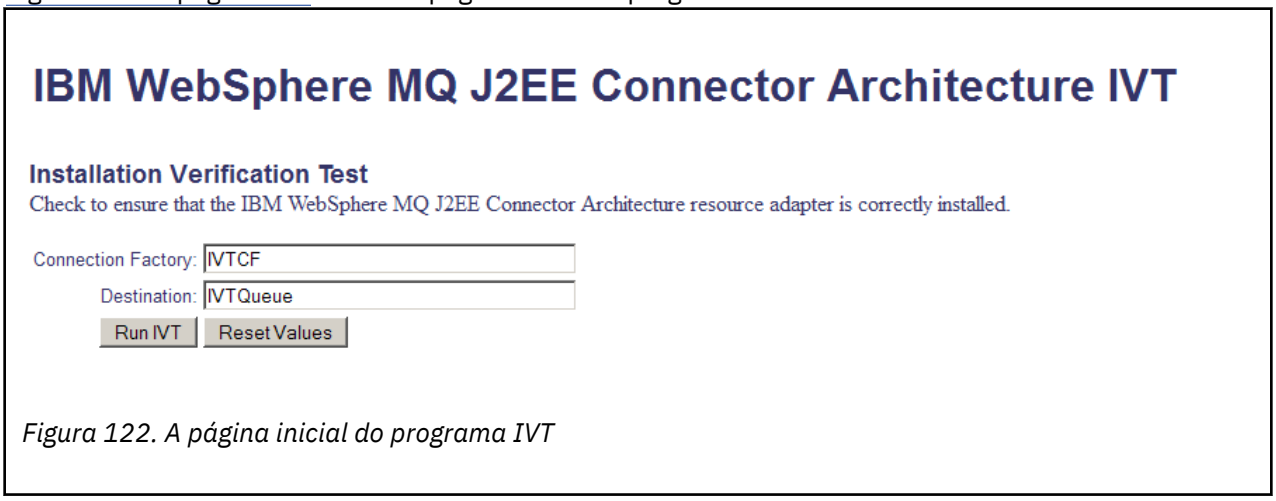


Figura 122. A página inicial do programa IVT

Para executar o teste, clique em **Run IVT**.Figura 123 na página 794 mostra a página exibida se o IVT for bem-sucedido.

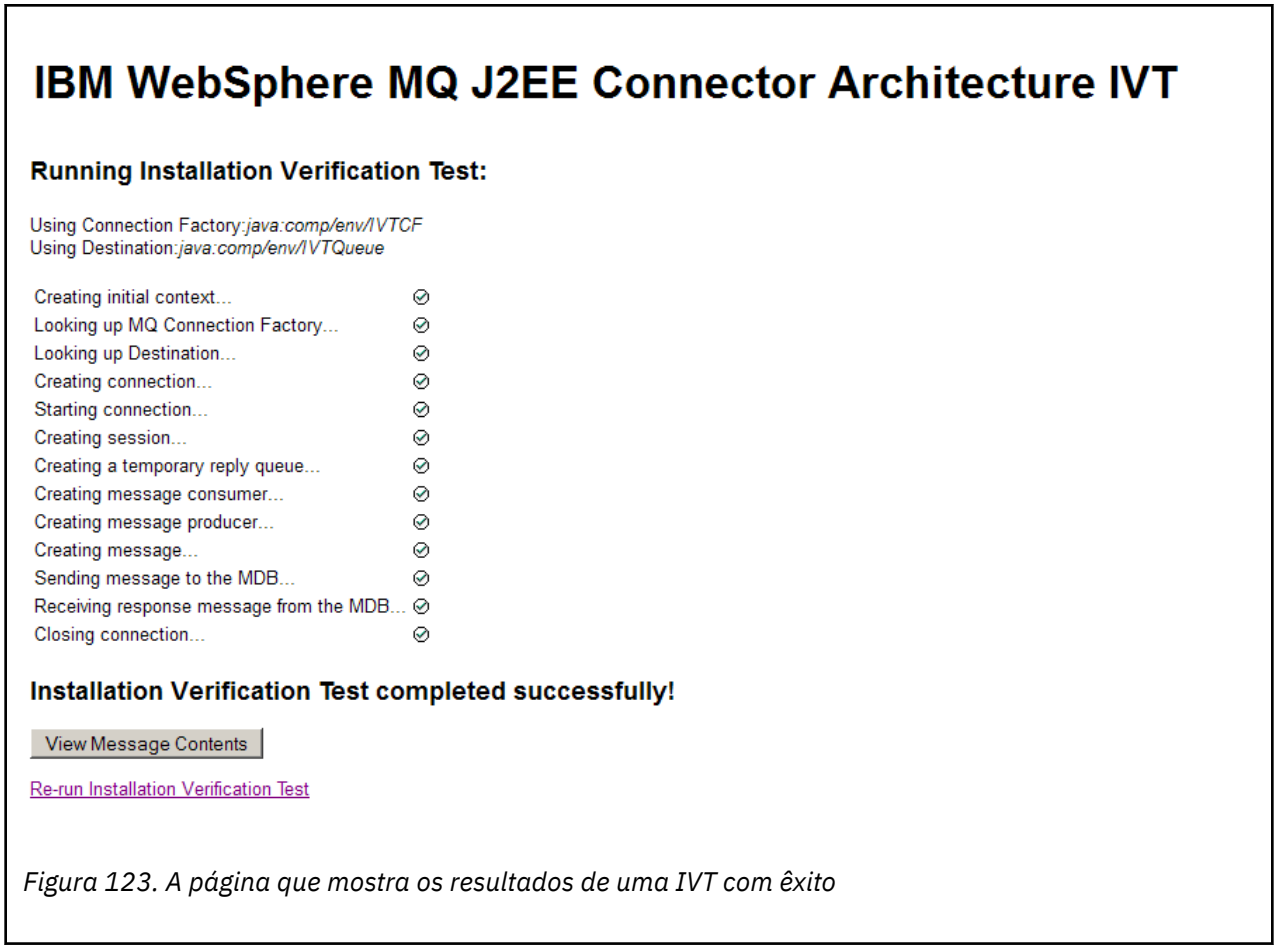


Figura 123. A página que mostra os resultados de uma IVT com êxito

Se o IVT falhar, uma página como essa mostrada em Figura 124 na página 795 será exibida. Para obter informações adicionais sobre a causa da falha, clique em **Visualizar rastreio de pilha**.

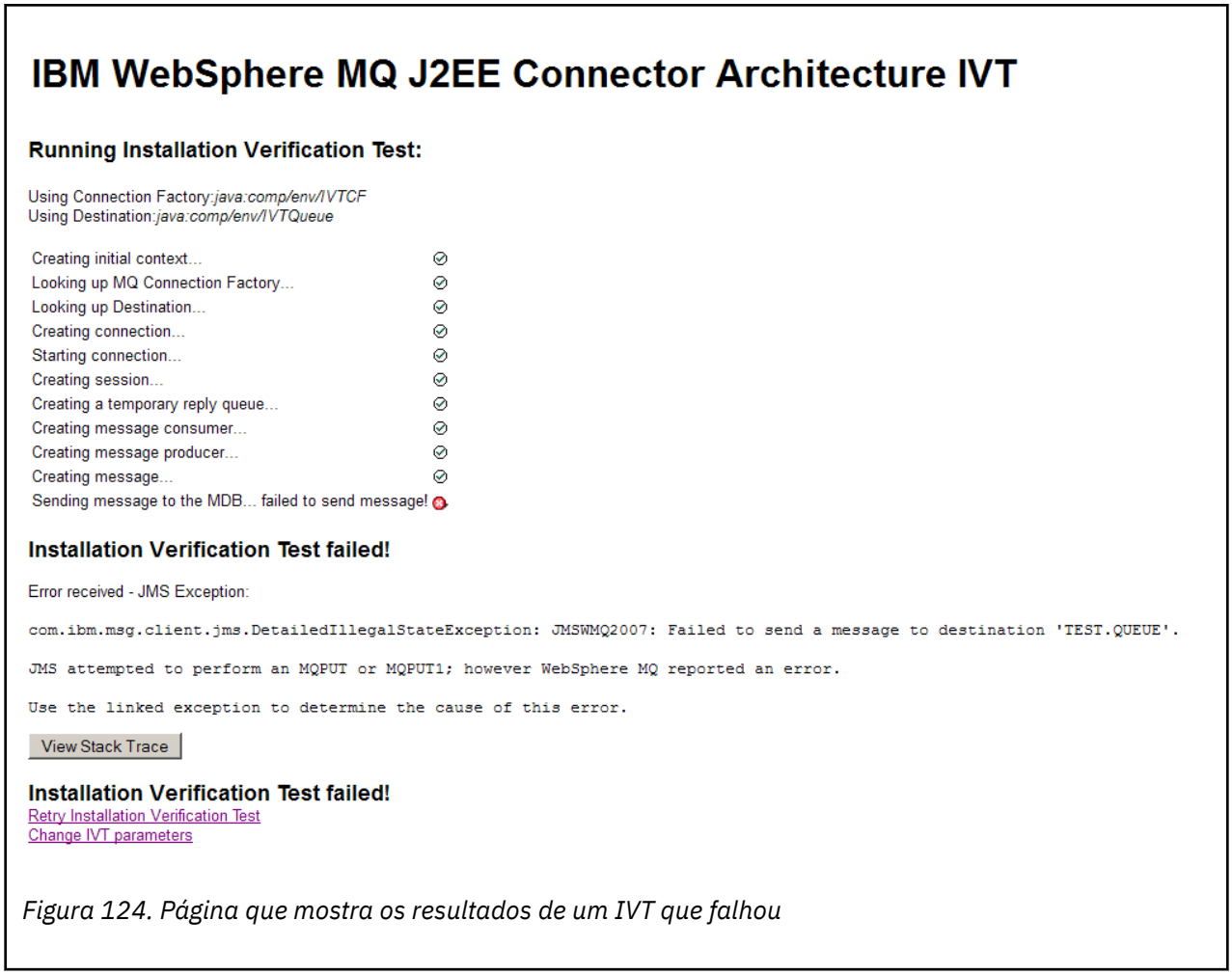


Figura 124. Página que mostra os resultados de um IVT que falhou

Para obter instruções detalhadas e informações sobre scripts de utilitário fornecidos para implementar o aplicativo IVT nos servidores de aplicativos JBoss e WAS CE, consulte:

Tarefas relacionadas

“Instalando e testando o adaptador de recursos MQ no WAS CE” na página 795
Instalando o adaptador de recursos IBM WebSphere MQ e executando o aplicativo de teste de verificação de instalação (IVT) no WebSphere Application Server CE.

“Instalando e testando o adaptador de recursos no JBoss AS 5.1 e 6” na página 798
Após instalar o adaptador de recursos IBM WebSphere MQ no JBoss AS 5.1 ou 6, é possível testar a instalação do adaptador de recursos instalando e executando o aplicativo de teste de verificação de instalação (IVT).

Instalando e testando o adaptador de recursos MQ no WAS CE

Instalando o adaptador de recursos IBM WebSphere MQ e executando o aplicativo de teste de verificação de instalação (IVT) no WebSphere Application Server CE.

Antes de começar

Esta tarefa supõe que você tenha um servidor WebSphere Application Server CE em execução e que esteja familiarizado com tarefas de administração padrão para ele. Essa tarefa também supõe que você tenha uma instalação do IBM WebSphere MQ em seu sistema local e que esteja familiarizado com as tarefas de administração padrão

Se você estiver usando o adaptador de recursos para conectar-se a um cliente IBM WebSphere MQ e precisar executar transações XA distribuídas, deverá seguir as etapas adicionais marcadas como **Somente XA do Cliente**

1. Crie um gerenciador de filas chamado `ExampleQMe` configure-o conforme descrito em "Preparando e executando os programas de amostra" na página 111 , observando que o listener deve ser iniciado na porta 1414, o canal a ser usado é chamado `SYSTEM.DEF.SVRCONN` e a fila usada pelo aplicativo IVT é nomeado `TEST.QUEUE`. A fila modelo `SYSTEM.DEFAULT.MODEL.QUEUE` também precisará receber a autoridade DSP e PUT para que esse aplicativo possa criar uma fila de resposta temporária Se você desejar usar um gerenciador de filas diferente, detalhes de conexão diferentes ou uma fila diferente, consulte "Implementando o aplicativo IVT no WAS CE com um ambiente customizado do MQ" na página 797
2. Obtenha o arquivo do adaptador de recursos (`wmq.jmsra.rar`), o aplicativo IVT (`wmq.jmsra.ivt.ear`) e o `WAS_CE_jmsra_deployment_plan.xml` e `WAS_CE_jmsra_ivt_deployment_plan.xml deployment plan files`. Para obter detalhes sobre o local desses arquivos, consulte "Instalação do adaptador de recursos do WebSphere MQ" na página 743..

Para obter uma descrição de ligações e conexões de modo de cliente, consulte "Modos de Conexão para Classes do WebSphere MQ para JMS" na página 783

Se desejar usar uma fila, gerenciador de filas, porta, host, canal ou usar o modo de ligações diferente em vez do modo cliente, consulte "Implementando o aplicativo IVT no WAS CE com um ambiente customizado do MQ" na página 797.

Procedimento

1. **Somente XA do cliente**: edite sua cópia do arquivo `WAS_CE_jmsra_deployment_plan.xml` ..
   a) Localize a definição de conexão jms / ivt/IVTCF e modifique-a de forma que o connection factory seja ativado por transação XA
      i) Comente a seção NonXA :

      ```
      <conn:xa-transaction>
      ```

      ii) Remova o comentário da seção de configuração XA:

      ```
      <conn:xa-transaction>
          <conn:transaction-caching/>
      </conn:xa-transaction>
      ```

   b) Salve as mudanças.
2. Opcional: **Somente XA do cliente**: Modifique o descritor de conjunto do MDB para requerer transações... Isso força o MDB no IVT a participar de uma transação XA, embora o aplicativo IVT ainda funcione com essa modificação.
   a) Abra o arquivo `wmq.jmsra.ivt.ear`.
   b) Abra o arquivo `WMQ_IVT_MDB.jar` dentro dele
   c) Edite o `META-INF/ejb-jar.xml`.
      i) Comente ou exclua a linha dentro do descritor de montagem:

      ```
      <trans-attribute>NotSupported</trans-attribute>
      ```

      ii) Remova o comentário da linha no descritor de montagem:

      ```
      <trans-attribute>Required</trans-attribute>
      ```

      iii) Salve suas alterações e atualize o arquivo dentro do arquivo `WMQ_IVT_MDB.jar`
      iv) Atualize o arquivo wmq.jmsra.ivt.ear com o arquivo `WMQ_IVT_MDB.jar` modificado

3. Implementar o adaptador de recursos em seu servidor usando o arquivo de plano de implementação modificado.
   a) Para fazer isso na linha de comandos, digite o seguinte comando do WAS CE:

   ```
   deploy -u system -p manager deploy wmq.jmsra.rar WAS_CE_jmsra_deployment_plan.xml
   ```

   b) Usando a interface de administração da web, acesse **Aplicativos> Implementador:**
      i) Configure o Archive como o arquivo wmq.jmsra.rar .
      ii) Configure o Plano para o arquivo WAS_CE_jmsra_deployment_plan.xml .
      iii) Assegure que 'Iniciar aplicativo após a instalação' esteja selecionado.
      iv) Clique em **Instalar**.
4. Implementar o aplicativo IVT em seu servidor usando o plano de implementação fornecido.
   a) Na linha de comandos, isso pode ser feito usando o seguinte comando WAS CE:

   ```
   deploy -u system -p manager deploy wmq.jmsra.ivt.ear WAS_CE_jmsra_ivt_deployment_plan.xml
   ```

   b) Usando a interface de administração da web, acesse **Aplicativos> Implementador:**
      i) Configure o **Archive** para o arquivo `wmq.jmsra.ivt.ear` .
      ii) Configure o **Plano** para ser o arquivo `WAS_CE_jmsra_ivt_deployment_plan.xml`
      iii) Certifique-se de que **Iniciar aplicativo após instalação** esteja selecionado.
      iv) Clique em **Instalar**.
5. Execute o aplicativo IVT. Para saber detalhes adicionais, consulte a seção “O programa de teste de verificação de instalação para o adaptador de recursos WebSphere MQ” na página 792. Para o WAS CE, a URL padrão é `http://localhost:8080/WMQ_IVT/`

Implementando o aplicativo IVT no WAS CE com um ambiente customizado do MQ

Se você desejar usar uma fila, um gerenciador de filas, uma porta, um host, um canal ou um modo de ligações diferentes em vez do modo cliente, deverá modificar o aplicativo IVT e os scripts associados no WebSphere Application Server CE antes de implementar o adaptador de recursos ou o aplicativo IVT.

Sobre esta tarefa

Se desejar implementar em uma configuração diferente daquela especificada no “Instalando e testando o adaptador de recursos MQ no WAS CE” na página 795, ou seja, se desejar usar uma fila diferente, gerenciador de filas, porta, host, canal ou modo de ligações em vez do modo cliente, execute as etapas a seguir antes de implementar o adaptador de recursos ou aplicativo IVT.

Procedimento

1. Se você desejar especificar um gerenciador de fila e uma fila diferentes para usar para o aplicativo IVT, configure valores para o gerenciador de filas e a fila em `WAS_CE_jmsra_deployment_plan.xml`, para obter detalhes, consulte “Configurando valores para o gerenciador de filas e a fila” na página 798
2. Se você desejar especificar um gerenciador de filas e uma fila diferentes na configuração para o bean acionado por mensagens (MDB), configure valores para o gerenciador de filas e a fila que você está usando no `WAS_CE_jmsra_ivt_deployment_plan.xml`, para obter detalhes, consulte “Configurando valores para a configuração do MDB” na página 798.
3. Se você estiver configurando o adaptador de recursos para conectar ao IBM WebSphere MQ no modo de ligações, assegure-se de que as bibliotecas JNI estejam no caminho do sistema ou no caminho para o WAS CE. Para obter informações adicionais, consulte “Instalando e testando o adaptador de recursos MQ no WAS CE” na página 795.

4. Se você já tiver implementado o adaptador de recursos, poderá reimplementá-lo com o plano de implementação modificado para alterar as configurações usando o comando a seguir:

   ```
   deploy --user system --password manager redeploy wmq.jmsra.rar
   WAS_CE_jmsra_deployment_plan.xml
   ```

Como proceder a seguir

Continue implementando o adaptador de recursos, conforme descrito em "Instalando e testando o adaptador de recursos MQ no WAS CE" na página 795

Configurando valores para o gerenciador de filas e a fila

Explica como configurar valores para o gerenciador de filas e a fila que você está usando no `WAS_CE_jmsra_deployment_plan.xml`

Procedimento

No `WAS_CE_jmsra_deployment_plan.xml`, configure valores para o gerenciador de fila e fila que você está usando para o aplicativo IVT.

Para a definição de conexão jms / ivt/IVTCF:

1. Configure o valor do elemento queueManager para ser o nome do gerenciador de filas.
2. Se você estiver usando uma conexão do cliente, configure o valor dos vários elementos de conexão do cliente para ser apropriado para uma conexão com seu gerenciador de filas.
3. Se você estiver usando uma conexão de ligação:
   a. Configure o valor do elemento transportType para BINDINGS.
   b. Comente ou exclua os vários elementos de conexão do cliente.
4. Para o destino da mensagem jms / ivt/IVTQueue, configure o valor do elemento de Nome baseQueuepara ser o nome da fila criada para o aplicativo IVT
5. Salve as mudanças.

Configurando valores para a configuração do MDB

Explica como configurar valores para a configuração do MDB no `WAS_CE_jmsra_deployment_plan.xml`

Procedimento

Em `WAS_CE_jmsra_ivt_deployment_plan.xml` , configure valores para o gerenciador de fila e a fila que você está usando na configuração para o MDB

Para o bean acionado por mensagens WMQ_IVT_MDB:

1. Configure o valor do elemento queueManager para ser o nome do gerenciador de filas.
2. Se você estiver usando uma conexão do cliente, configure o valor dos vários elementos de conexão do cliente para ser apropriado para uma conexão com seu gerenciador de filas.
3. Se você estiver usando uma conexão de ligação:
   a. Configure o valor do elemento transportType para BINDINGS.
   b. Comente ou exclua os vários elementos de conexão do cliente.
4. Salve as mudanças.

Instalando e testando o adaptador de recursos no JBoss AS 5.1 e 6

Após instalar o adaptador de recursos IBM WebSphere MQ no JBoss AS 5.1 ou 6, é possível testar a instalação do adaptador de recursos instalando e executando o aplicativo de teste de verificação de instalação (IVT).

Antes de começar

Importante: Essas instruções são para JBoss AS 5.1 e 6, elas não são válidas para JBoss AS 7.

Para obter informações sobre como instalar o adaptador de recursos em JBoss EAP 6.3, consulte "Instalando e testando o adaptador de recursos no JBoss EAP 6.3" na página 801.

Esta tarefa supõe que você tenha um servidor JBoss em execução e esteja familiarizado com tarefas de administração padrão para ele. Esta tarefa também presume que você tenha uma instalação do IBM WebSphere MQ em seu sistema local e que você esteja familiarizado com as tarefas de administração padrão.

Se você estiver usando o adaptador de recursos para se conectar a um cliente IBM WebSphere MQ e precisar executar transações XA distribuídas, deverá seguir as etapas adicionais marcadas como **Somente XA do Cliente**. Para obter uma descrição de ligações e conexões de modo de cliente, consulte "Modos de Conexão para Classes do WebSphere MQ para JMS" na página 783

Procedimento

1. Crie um gerenciador de filas chamado ExampleQMe configure-o conforme descrito em "Preparando e executando os programas de amostra" na página 111..

   Ao configurar o gerenciador de filas, observe os pontos a seguir:

   - O listener deve ser iniciado na porta 1414.
   - O canal a ser usado é chamado SYSTEM.DEF.SVRCONN.
   - A fila usada pelo aplicativo IVT é denominada TEST.QUEUE.

   Também é necessário conceder autoridade DSP e PUT à fila modelo SYSTEM.DEFAULT.MODEL.QUEUE para que esse aplicativo possa criar uma fila de resposta provisória.

   Se você desejar usar um gerenciador de filas diferente, detalhes de conexão diferentes ou uma fila diferente, consulte "Implementando o aplicativo IVT no WAS CE com um ambiente customizado do MQ" na página 797
2. Obtenha o arquivo do adaptador de recurso (`wmq.jmsra.rar`), o aplicativo IVT (`wmq.jmsra.ivt.ear`) e o arquivo `jboss-jmsra-ds.xml` .

   Para obter o local desses arquivos, consulte "Instalação do adaptador de recursos do WebSphere MQ" na página 743
3. **Somente XA do Cliente**: Edite o arquivo `jboss-jmsra-ds.xml` para ativar Transações XA no connection factory.
   a) Comente ou exclua a linha dentro da definição de connection factory `<local-transaction/>`.
   b) Remova o comentário da linha dentro da definição de connection factory `<xa-transaction/>`
   c) Salve as mudanças.
4. **Somente XA do cliente**: (opcional) Modifique o descritor de montagem do MDB para requerer transações... Isso força o MDB no IVT a participar de uma transação XA, embora o aplicativo IVT ainda funcione com essa modificação.
   a) Abra o arquivo `wmq.jmsra.ivt.ear`.
   b) Abra o arquivo `WMQ_IVT_MDB.jar` dentro dele
   c) Editar `META-INF/ejb-jar.xml`:
      i) Comente ou exclua a linha dentro do descritor de montagem:

      ```
      <trans-attribute>NotSupported</trans-attribute>
      ```

      ii) Remova o comentário da linha no descritor de montagem:

      ```
      <trans-attribute>Required</trans-attribute>
      ```

iii) Salve suas alterações e atualize o arquivo dentro do arquivo `WMQ_IVT_MDB.jar`

iv) Atualize o arquivo `wmq.jmsra.ivt.ear` com o `WMQ_IVT_MDB.jar`modificado

5. Implemente o adaptador de recursos para o servidor copiando o arquivo `wmq.jmsra.rar` no diretório `jboss/server/default/deploy`.
6. Crie os recursos JMS necessários para o Aplicativo IVT copiando o arquivo `jboss-jmsra-ds.xml` no diretório `jboss/server/default/deploy`.
7. Implemente o aplicativo IVT copiando o arquivo `wmq.jmsra.ivt.ear` no diretório `jboss/server/default/deploy`.
8. Execute o aplicativo IVT. Para saber detalhes adicionais, consulte a seção "O programa de teste de verificação de instalação para o adaptador de recursos WebSphere MQ" na página 792. Para JBoss, a URL padrão é `http://localhost:8080/WMQ_IVT/`..

Implementando o aplicativo IVT no JBoss com um ambiente IBM WebSphere MQ customizado

Ao instalar o adaptador de recursos do IBM WebSphere MQ no JBoss, se você desejar usar uma fila diferente, um gerenciador de filas, uma porta, um host, um canal ou um modo de ligações em vez do modo de cliente, deverá primeiro modificar o aplicativo IVT e os scripts associados no JBoss antes de implementar o adaptador de recursos ou o aplicativo IVT.

Sobre esta tarefa

Importante: Essas instruções são aplicáveis apenas para Java EE Versões 6 e 5, não para Java EE Versão 7. Portanto, o uso destas instruções para JBoss Versão 8 (WildFly) não é suportado.

Se desejar implementar em uma configuração diferente daquela especificada no "Instalando e testando o adaptador de recursos no JBoss AS 5.1 e 6" na página 798, ou seja, se desejar usar um gerenciador de filas, fila, porta, host, canal ou modo de ligações diferentes em vez do modo cliente, conclua as etapas a seguir antes de implementar o adaptador de recurso ou o aplicativo IVT.

Procedimento

1. Se você desejar especificar um gerenciador de fila e uma fila diferentes para usar para o aplicativo IVT, configure valores para o gerenciador de filas e a fila
   a) Para a definição de conexão jms / ivt/IVTCF:
      i) Configure o valor da propriedade de configuração queueManager como o nome do gerenciador de filas.
      ii) Se você estiver usando uma conexão do cliente, configure o valor dos vários elementos de conexão do cliente para ser apropriado para uma conexão com seu gerenciador de filas.
      iii) Se você estiver usando uma conexão de ligações, configure o valor do elemento transportType para BINDINGS e, em seguida, comente ou exclua os vários elementos de conexão do cliente.
   b) Para o mbean jms / ivt/IVTQueue, configure o valor do elemento de Nome baseQueuepara ser o nome da fila criada para o aplicativo IVT.
   c) Salve as mudanças.
2. Se desejar especificar um gerenciador de filas e uma fila diferentes na configuração para o bean acionado por mensagem (MDB), modifique a configuração do MDB para se conectar ao gerenciador de filas e à fila.
   a) Abra o arquivo `wmq.jmsra.ivt.ear`.
   b) Abra o `WMQ_IVT_MDB.jar` dentro dele
   c) Editar `META-INF/ejb-jar.xml`:
      i) Configure o valor da propriedade activation-config queueManager para ser o nome do gerenciador de filas.

ii) Se você estiver usando uma conexão do cliente, configure o valor das várias propriedades de configuração de ativação de conexão do cliente para ser apropriado para uma conexão com seu gerenciador de filas.

iii) Se você estiver usando uma conexão de ligações, configure o valor de transportType activation-config-property para BINDINGS e, em seguida, comente ou exclua os vários elementos de conexão do cliente.

d) Salve as alterações e atualize o arquivo dentro do arquivo `WMQ_IVT_MDB.jar`

e) Atualize o arquivo `wmq.jmsra.ivt.ear` com o `WMQ_IVT_MDB.jar`modificado

3. Se estiver configurando o adaptador de recursos para se conectar ao IBM WebSphere MQ no modo de ligações, assegure-se de que as bibliotecas JNI estejam no caminho do sistema ou no caminho para JBoss. Veja detalhes na seção "Configurando as bibliotecas do Java Native Interface (JNI)" na página 733.

Como proceder a seguir

Continue implementando o adaptador de recursos, conforme descrito em "Instalando e testando o adaptador de recursos no JBoss AS 5.1 e 6" na página 798

.

Instalando e testando o adaptador de recursos no JBoss EAP 6.3

Depois de instalar o adaptador de recurso do IBM WebSphere MQ no JBoss Enterprise Application Platform (EAP) 6.3, em um servidor independente ou em um servidor em execução em um domínio gerenciado, é possível testar a instalação do adaptador de recursos instalando e executando o aplicativo de teste de verificação de instalação (IVT).

Sobre esta tarefa

Importante: Essas instruções são apenas para JBoss EAP 6.3 Para obter informações sobre como instalar o adaptador de recursos em JBoss AS 5.1 e 6, consulte "Instalando e testando o adaptador de recursos no JBoss AS 5.1 e 6" na página 798

Esta tarefa supõe que você tenha um servidor JBoss em execução e esteja familiarizado com tarefas de administração padrão para ele. Esta tarefa também supõe que você tenha uma instalação do IBM WebSphere MQ em seu sistema local e que esteja familiarizado com a administração padrão

Procedimento

1. Crie um gerenciador de filas chamado ExampleQMe configure-o conforme descrito em "Preparando e executando os programas de amostra" na página 111..

   Ao configurar o gerenciador de filas, observe os pontos a seguir:

   - O listener deve ser iniciado na porta 1414.
   - O canal a ser usado é chamado SYSTEM.DEF.SVRCONN.
   - A fila usada pelo aplicativo IVT é denominada TEST.QUEUE.

   Também é necessário conceder autoridade DSP e PUT à fila modelo SYSTEM.DEFAULT.MODEL.QUEUE para que esse aplicativo possa criar uma fila de resposta provisória.

   Se você desejar usar um gerenciador de filas diferente, detalhes de conexão diferentes ou uma fila diferente, consulte "Implementando o aplicativo IVT no WAS CE com um ambiente customizado do MQ" na página 797
2. Obtenha o arquivo do adaptador de recursos (`wmq.jmsra.rar`) e o aplicativo IVT (`wmq.jmsra.ivt.ear`)

   Para obter o local desses arquivos, consulte "Instalação do adaptador de recursos do WebSphere MQ" na página 743

3. Instale o adaptador de recursos e, em seguida, teste a instalação executando o aplicativo de teste de verificação de instalação (IVT):
   - Se você estiver instalando o adaptador de recursos em um servidor independente, consulte “Instalando e testando em um servidor independente” na página 802
   - Se você estiver instalando o adaptador de recursos em um servidor em execução em um domínio gerenciado, consulte “Instalando e testando em um servidor em execução em um domínio gerenciado” na página 803

Instalando e testando em um servidor independente

Após instalar o adaptador de recursos do IBM WebSphere MQ no JBoss EAP 6.3 em um servidor independente, é possível testar a instalação do adaptador de recursos instalando e executando o aplicativo de teste de verificação de instalação (IVT)..

Sobre esta tarefa

As informações nesta tarefa são para instalar e testar o adaptador de recursos em um servidor independente Se você estiver instalando o adaptador de recursos em um servidor em execução em um domínio gerenciado, consulte “Instalando e testando em um servidor em execução em um domínio gerenciado” na página 803

Importante: Essas instruções são apenas para JBoss EAP 6.3 Para obter informações sobre como instalar o adaptador de recursos em JBoss AS 5.1 e 6, consulte “Instalando e testando o adaptador de recursos no JBoss AS 5.1 e 6” na página 798

Procedimento

1. Implemente o adaptador de recursos para o servidor copiando o arquivo `wmq.jmsra.rar` no diretório <*EAP_HOME*>`/standalone/deployments`.
2. Crie os recursos JMS necessários para o Aplicativo IVT incluindo as entradas a seguir na seção `<resource-adapters>` do arquivo <*EAP_HOME*>`/standalone/configuration/standalone-full.xml` :

```
<resource-adapter id="wmq.jmsra">
  <archive>
      wmq.jmsra.rar
  </archive>
  <transaction-support>NoTransaction</transaction-support>
  <connection-definitions>
     <connection-definition
          class-name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
          jndi-name="java:jboss/jms/ivt/IVTCF"
          enabled="true"
          use-java-context="true"
          pool-name="IVTCF">
       <config-property name="port">
           1414
       </config-property>
       <config-property name="hostName">
           localhost
       </config-property>
       <config-property name="channel">
           SYSTEM.DEF.SVRCONN
       </config-property>
       <config-property name="transportType">
           CLIENT
       </config-property>
       <config-property name="queueManager">
           ExampleQM
       </config-property>
     </connection-definition>
  </connection-definitions>
  <admin-objects>
    <admin-object class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
                  jndi-name="java:jboss/jms/ivt/IVTQueue"
                  pool-name="IVTQueue">
      <config-property name="baseQueueName">
          TEST.QUEUE
      </config-property>
```

```
    </admin-object>
  </admin-objects>
</resource-adapter>
```

3. Inclua as informações a seguir nos parâmetros de inicialização do servidor de aplicativos:

```
-Dcom.ibm.mq.connector.IVTMDBCFJNDIName=java:jboss/jms/ivt/IVTCF
```

4. Implementar o aplicativo IVT copiando o arquivo `wmq.jmsra.ivt.ear` no diretório *<EAP_HOME>*`/standalone/deployments`.
5. Inicie o servidor de aplicativos.
6. Execute o aplicativo IVT.

   Para obter mais informações, consulte "O programa de teste de verificação de instalação para o adaptador de recursos WebSphere MQ" na página 792. Para JBoss, a URL padrão é `http://localhost:8080/WMQ_IVT/`.

   Nota: Os nomes JNDI usados para os recursos JMS necessários para executar o aplicativo IVT são os seguintes:

```
Connection Factory : IVTCF
JNDI name          : java:jboss/jms/ivt/IVTCF

Destination        : IVTQueue
JNDI name          : java:jboss/jms/ivt/IVTQueue
```

   Quando você ativar o aplicativo IVT usando a URL especificada acima, insira os nomes JNDI desses recursos em seus respectivos campos e, em seguida, execute o aplicativo clicando em **Executar IVT**

Instalando e testando em um servidor em execução em um domínio gerenciado

Após instalar o adaptador de recursos IBM WebSphere MQ no JBoss EAP 6.3 em um servidor em execução em um domínio gerenciado, é possível testar a instalação do adaptador de recursos instalando e executando o aplicativo de teste de verificação de instalação (IVT).

Sobre esta tarefa

As informações nesta tarefa são para instalar e testar o adaptador de recursos em um servidor em execução em um domínio gerenciado. Se você estiver instalando o adaptador de recursos em um servidor independente, consulte "Instalando e testando em um servidor independente" na página 802

Importante: Essas instruções são apenas para JBoss EAP 6.3 Para obter informações sobre como instalar o adaptador de recursos em JBoss AS 5.1 e 6, consulte "Instalando e testando o adaptador de recursos no JBoss AS 5.1 e 6" na página 798

Procedimento

1. Implemente o adaptador de recursos em seu servidor usando o JBoss Management Console ou a CLI de Gerenciamento.
2. Crie os recursos JMS necessários para o Aplicativo IVT incluindo as entradas a seguir na seção `<resource-adapters>` do `<EAP_HOME>/domain/configuration/domain.xml file`:

```
<resource-adapter id="wmq.jmsra">
  <archive>
      wmq.jmsra.rar
  </archive>
  <transaction-support>NoTransaction</transaction-support>
  <connection-definitions>
    <connection-definition
         class-name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
         jndi-name="java:jboss/jms/ivt/IVTCF"
         enabled="true"
         use-java-context="true"
         pool-name="IVTCF">
      <config-property name="port">
          1414
      </config-property>
      <config-property name="hostName">
```

```
            localhost
        </config-property>
        <config-property name="channel">
            SYSTEM.DEF.SVRCONN
        </config-property>
        <config-property name="transportType">
            CLIENT
        </config-property>
        <config-property name="queueManager">
            ExampleQM
        </config-property>
      </connection-definition>
    </connection-definitions>
    <admin-objects>
      <admin-object class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
                    jndi-name="java:jboss/jms/ivt/IVTQueue"
                    pool-name="IVTQueue">
        <config-property name="baseQueueName">
            TEST.QUEUE
        </config-property>
      </admin-object>
    </admin-objects>
</resource-adapter>
```

3. Inclua as informações a seguir nos parâmetros de inicialização do servidor de aplicativos:

```
-Dcom.ibm.mq.connector.IVTMDBCFJNDIName=java:jboss/jms/ivt/IVTCF
```

4. Pare e reinicie o servidor de aplicativos.
5. Implemente o aplicativo IVT usando o JBoss Management Console ou a CLI de Gerenciamento.
6. Execute o aplicativo IVT.

   Para obter mais informações, consulte “O programa de teste de verificação de instalação para o adaptador de recursos WebSphere MQ” na página 792. Para JBoss, a URL padrão é `http://localhost:8080/WMQ_IVT/`.

   Nota: Os nomes JNDI usados para os recursos JMS necessários para executar o aplicativo IVT são os seguintes:

```
Connection Factory : IVTCF
JNDI name          : java:jboss/jms/ivt/IVTCF

Destination        : IVTQueue
JNDI name          : java:jboss/jms/ivt/IVTQueue
```

   Insira os nomes JNDI desses recursos em seus respectivos campos quando você ativar o aplicativo IVT usando a URL especificada acima e, em seguida, executar o aplicativo clicando em **Executar IVT**.

Scripts fornecidos com classes do WebSphere MQ para JMS

Vários scripts são fornecidos para ajudar com tarefas comuns que precisam ser executadas ao usar classes WebSphere MQ para JMS.

Tabela 102 na página 804 lista todos os scripts e seus usos. Os scripts estão no subdiretório `bin` das classes WebSphere MQ para o diretório de instalação JMS.

Tabela 102. Scripts fornecidos com classes do WebSphere MQ para JMS

Utilitário	**Usar**
Limpeza[1]	Esse script é mantido para compatibilidade com liberações anteriores, mas não executa nenhuma função. A limpeza manual das informações de assinatura não é mais necessária
DefaultConfiguration	Executa o aplicativo de configuração padrão em plataformas diferentes do Windows.
formatLog[1]	Esse script é mantido para compatibilidade com liberações anteriores, mas não executa nenhuma função. Agora a saída de log é produzida em texto legível.

Tabela 102. Scripts fornecidos com classes do WebSphere MQ para JMS (continuação)	
Utilitário	**Usar**
IVTRun[1] IVTSetup[1] IVTTidy[1]	Usado no teste de verificação de instalação ponto a ponto, conforme descrito em "O teste de verificação de instalação ponto a ponto para classes do WebSphere MQ para JMS" na página 785.
JMSAdmin[1]	Executa a ferramenta de administração do JMS do WebSphere MQ , conforme descrito em "Chamando a ferramenta de administração do IBM WebSphere MQ classes for JMS" na página 946
JMSAdmin.config	O arquivo de configuração da ferramenta de administração JMS do WebSphere MQ , conforme descrito em "Configurando a Ferramenta de Administração JMS" na página 947.
PSIVTRun[1]	Executa o programa de teste de verificação da instalação de publicação/assinatura, conforme descrito em "O teste de verificação de instalação de publicação / assinatura para classes WebSphere MQ para JMS" na página 788.
PSReportDump.class	Essa classe é mantida para compatibilidade com liberações anteriores, mas não executa nenhuma função.
setjmsenv	Configura as variáveis de ambiente para executar um WebSphere MQ classes para aplicativo JMS em uma Java virtual machine (JVM) de 32 bits em sistemas UNIX and Linux , conforme descrito em "Variáveis de ambiente usadas por classes IBM WebSphere MQ para JMS" na página 731.
setjmsenv64	Configura as variáveis de ambiente para executar um WebSphere MQ classes para aplicativo JMS em uma JVM de 64 bits em sistemas UNIX and Linux , conforme descrito em "Variáveis de ambiente usadas por classes IBM WebSphere MQ para JMS" na página 731.
Nota: 1. No Windows, o nome do arquivo tem a extensão de arquivo .bat.	

Suporte para OSGi

OSGi fornece uma estrutura que suporta a implementação de aplicativos como pacotes configuráveis. Nove pacotes configuráveis do OSGi são fornecidos como parte do IBM WebSphere MQ classes for JMS.

O OSGi fornece uma estrutura Java de propósito geral, segura e gerenciada, que suporta a implementação de aplicativos fornecidos na forma de pacotes configuráveis. Os dispositivos compatíveis com o OSGi poderão fazer download e instalar pacotes configuráveis, e removê-los quando não forem mais necessários. A estrutura gerencia a instalação e a atualização de pacotes configuráveis de um modo dinâmico e escalável.

O IBM WebSphere MQ classes for JMS.. inclui os seguintes pacotes configuráveis OSGi.

com.ibm.msg.client.osgi.jms<version number>.jar
A camada comum de código no IBM WebSphere MQ classes for JMS Para obter informações sobre a arquitetura em camadas das classes do WebSphere MQ para JMS, consulte "Uma arquitetura em camadas" na página 809

com.ibm.msg.client.osgi.jms.prereq_<version number>.jar
O pré-requisito do archive Java (JAR) para a camada comum.

com.ibm.msg.client.osgi.commonservices.j2se_<version number>.jar
Serviços comuns para aplicativos Java Platform, Standard Edition (Java SE).

com.ibm.msg.client.osgi.nls_<version number>.jar
Mensagens para a camada comum.

com.ibm.msg.client.osgi.wmq_<version number>.jar
O provedor de sistemas de mensagens IBM WebSphere MQ em IBM WebSphere MQ classes for JMS. Para obter informações sobre a arquitetura em camadas do IBM WebSphere MQ classes for JMS , consulte “Uma arquitetura em camadas” na página 809

com.ibm.msg.client.osgi.wmq.prereq_<version number>.jar
Os arquivos JAR de pré-requisito para o provedor de sistemas de mensagens do IBM WebSphere MQ.

com.ibm.msg.client.osgi.wmq.nls_<version number>.jar
Mensagens para o provedor de sistemas de mensagens IBM WebSphere MQ .

com.ibm.mq.osgi.directip_< version number> .jar
Os arquivos JAR para permitir que o provedor de sistemas de mensagens do IBM WebSphere MQ crie uma conexão em tempo real com um broker..

em que < version number> é o número da versão do WebSphere MQ que foi instalado..

Os pacotes configuráveis são instalados no subdiretório `java/lib/OSGi` da instalação do WebSphere MQ ou da pasta `java\lib\OSGi` no Windows.

O pacote configurável com.ibm.mq.osgi.java < version number> .jar, que também é instalado no subdiretório `java/lib/OSGi` de sua instalação do WebSphere MQ , ou a pasta `java\lib\OSGi` no Windows, faz parte das classes do WebSphere MQ para Java Esse pacote configurável não deve ser carregado em um ambiente de tempo de execução OSGi que tenha as classes WebSphere MQ para JMS carregadas.

Os pacotes configuráveis OSGi para as classes WebSphere MQ para JMS foram gravados na especificação OSGi Release 4. Eles não funcionam em um ambiente de liberação 3 do OSGi.

Deve-se configurar seu caminho de sistema ou caminho da biblioteca corretamente para que o ambiente de tempo de execução do OSGi possa localizar quaisquer arquivos da DLL ou bibliotecas compartilhadas requeridas.

Se você usar os pacotes configuráveis do OSGi para o IBM WebSphere MQ classes for JMS, os tópicos temporários não funcionarão. Além disso, as classes de saída do canal gravadas em Java não são suportadas devido a um problema inerente em classes de carregamento em um ambiente do carregador de classes múltiplo como OSGi. Um pacote configurável do usuário pode estar ciente das classes do IBM WebSphere MQ para pacotes configuráveis JMS, mas os pacotes configuráveis do IBM WebSphere MQ classes for JMS não estão cientes de nenhum pacote configurável do usuário. Como resultado, o carregador de classes usado em um pacote configurável do IBM WebSphere MQ classes for JMS não pode carregar uma classe de saída de canal que está em um pacote configurável do usuário...

Para obter mais informações sobre o OSGi, consulte o website do OSGi Alliance.

Resolvendo problemas com classes IBM WebSphere MQ para JMS

É possível investigar os problemas, executando os programas de verificação de instalação e utilizando os recursos de rastreio e de log.

Se um programa não for concluído com êxito, execute um dos programas de verificação de instalação, conforme descrito em “O teste de verificação de instalação ponto a ponto para classes do WebSphere MQ para JMS” na página 785 e “O teste de verificação de instalação de publicação / assinatura para classes WebSphere MQ para JMS” na página 788, e siga o conselho fornecido nas mensagens de diagnóstico.

Criação de log eIBM WebSphere MQ classes for JMS

Por padrão, a saída de log é enviada ao arquivo `mqjms.log`. É possível redirecioná-la a um arquivo ou diretório específico.

O recurso de log do IBM WebSphere MQ classes for JMS é fornecido para relatar problemas sérios, principalmente problemas que podem indicar erros de configuração, em vez de erros de programação. Por padrão, a saída de log é enviada ao arquivo `mqjms.log` no diretório ativo da JVM.

É possível redirecionar a saída de log para outro arquivo configurando a propriedades com.ibm.msg.client.commonservices.log.outputName. O valor dessa propriedade pode ser:

- Um nome de caminho único.
- Uma lista separada por vírgula de nomes de caminho (todos os dados são registrados para todos os arquivos).

Cada nome de caminho pode ser:

- Absoluto ou relativo.
- `stderr` ou `System.err` para representar o fluxo de erro padrão.
- `sttdout` ou `System.out` para representar o fluxo de saída padrão.

Se o valor da propriedade identifica um diretório, a saída de log será gravada no `mqjms.log` nesse diretório. Se o valor da propriedade identifica um arquivo específico, a saída de log será gravada nesse arquivo.

É possível configurar essa propriedade no arquivo de configuração do IBM WebSphere MQ classes for JMS ou como uma propriedade de sistema no comando **java**. No exemplo a seguir, a propriedade é configurada como uma propriedade de sistema e identifica um arquivo específico:

```
java -Djava.library.path=library_path
     -Dcom.ibm.msg.client.commonservices.log.outputName=/mydir/mylog.txt
      MyAppClass
```

No comando, *library_path* é o caminho para o diretório que contém as bibliotecas do IBM WebSphere MQ classes for JMS (consulte “Configurando as bibliotecas do Java Native Interface (JNI)” na página 733).

É possível desativar a saída de log configurando a propriedade com.ibm.msg.client.commonservices.log.status para OFF. O valor padrão dessa propriedade é ON.

Os valores `System.err` e `System.out` podem ser configurados para enviar a saída do log para os fluxos `System.err` e `System.out`.

Introdução às classes WebSphere MQ para JMS, para programadores

WebSphere MQ classes para JMS é o provedor JMS fornecido com o WebSphere MQ. WebSphere MQ classes para JMS implementa as interfaces definidas no pacote javax.jms e também fornece dois conjuntos de extensões para a API JMS. Os aplicativos Java Platform, Standard Edition (Java SE) e Java Platform, Enterprise Edition (Java EE) podem usar as classes WebSphere MQ para JMS.

A especificação JMS define um conjunto de interfaces que os aplicativos podem usar para executar operações do sistema de mensagens A versão mais recente da especificação é a Versão 1.1 O pacote javax.jms especifica os detalhes das interfaces JMS e um provedor JMS implementa essas interfaces para um produto de sistema de mensagens específico. WebSphere MQ classes para JMS é um provedor JMS que implementa as interfaces JMS para WebSphere MQ.

O fluxo de lógica em um aplicativo JMS começa com objetos ConnectionFactory e Destination. O aplicativo usa um objeto ConnectionFactory para criar um objeto Connection, que representa a conexão ativa do aplicativo com um servidor de sistema de mensagens. O aplicativo usa o objeto Connection para criar um objeto Session, que é um único contexto encadeado para produzir e consumir mensagens. O aplicativo pode então usar o objeto Session e um objeto Destination para criar um objeto MessageProducer, que o aplicativo usa para enviar mensagens para o destino especificado. O destino é uma fila ou um tópico no sistema de mensagens e é contido pelo objeto Destination. O aplicativo também pode usar o objeto Session e um objeto Destination para criar um objeto MessageConsumer, que o aplicativo usa para receber mensagens que foram enviadas para o destino especificado.

A especificação JMS espera que ConnectionFactory e objetos de Destino sejam objetos administrados. Um administrador cria e mantém objetos administrados em um repositório central e um aplicativo JMS recupera esses objetos usando a JNDI (Java Naming and Directory Interface) O repositório de objetos administrados pode variar de um arquivo simples a um diretório Lightweight Directory Access Protocol (LDAP).

WebSphere MQ classes para JMS suporta o uso de objetos administrados. Um aplicativo pode usar todos os recursos do WebSphere MQ que são expostos por meio de classes WebSphere MQ para JMS sem ter nenhuma informação específica do WebSphere MQcodificada no próprio aplicativo. Esse acordo fornece ao aplicativo um grau de independência da configuração subjacente do WebSphere MQ . Para atingir esta independência, o aplicativo pode usar JNDI para recuperar connection factories e destinos que estão armazenados como objetos administrados, e usar somente as interfaces definidas no pacote javax.jms para executar operações do sistema de mensagens. Um administrador pode utilizar a ferramenta de administração JMS do WebSphere MQ ou IBM WebSphere MQ Explorer para criar e manter objetos administrados em um repositório central. No entanto, um servidor de aplicativos geralmente fornece seu próprio repositório para objetos administrados e suas próprias ferramentas para criar e manter os objetos. Um aplicativo Java EE pode, portanto, usar JNDI para recuperar objetos administrados a partir do repositório do servidor de aplicativos ou de um repositório central.

WebSphere MQ classes para JMS também fornece extensões para a API JMS. Liberações anteriores de classes do WebSphere MQ para JMS contêm extensões que são implementadas em objetos MQConnectionFactory, MQQueue e MQTopic. Esses objetos possuem propriedades e métodos específicos do WebSphere MQ. Os objetos podem ser objetos administrados ou um aplicativo pode criar os objetos dinamicamente no tempo de execução. Esta liberação do WebSphere MQ classes para JMS mantém essas extensões e é possível continuar a usar, sem mudança, quaisquer aplicativos que usem essas extensões. Essas extensões são conhecidas como *WebSphere MQ extensões JMS*. Observe que, neste conjunto de documentação, os objetos que são criados dinamicamente por um aplicativo no tempo de execução *não* são considerados objetos administrados.

Além das extensões JMS do WebSphere MQ , esta liberação de classes do WebSphere MQ para JMS fornece um conjunto mais genérico de extensões para a API JMS. Essas extensões são conhecidas como *IBM extensões JMS*e têm os objetivos gerais a seguir:

- Para fornecer um nível maior de consistência nos provedores JMS do IBM
- Para facilitar a gravação de um aplicativo de ponte entre dois sistemas de mensagens IBM
- Para facilitar a porta de um aplicativo de um provedor JMS do IBM para outro

O foco principal dessas extensões é a criação e configuração de connection factories e de destinos dinamicamente no tempo de execução, mas as extensões também fornecem uma função que não está diretamente relacionada ao sistema de mensagens, como a função para determinação de problemas.

Um connection factory, fila ou objeto de tópico criado usando a interface javax.jms ou um conjunto de extensões JMS pode ser endereçado usando qualquer uma dessas APIs; ou seja, ele pode ser convertido para qualquer uma das interfaces Para manter a portabilidade do aplicativo no nível mais alto, use a API mais genérica adequada para seus requisitos.

Os aplicativos Java SE e Java EE podem usar classes WebSphere MQ para JMS. Na plataforma Java EE , as classes WebSphere MQ para JMS suportam dois tipos de comunicação entre um componente de um aplicativo e um gerenciador de filas do WebSphere MQ :

Comunicação de saída
Usando a API do JMS diretamente, um componente de aplicativo cria uma conexão com um gerenciador de filas e, em seguida, envia e recebe mensagens

Por exemplo, o componente de aplicativo pode ser um aplicativo cliente, um servlet, um JavaServer Page (JSP), um enterprise Java bean (EJB) ou um bean acionado por mensagens (MDB). Neste tipo de comunicação, o contêiner do servidor de aplicativos fornece somente funções de baixo nível para suportar operações do sistema de mensagens, como definição do conjunto de conexões e gerenciamento de encadeamentos.

Comunicação de entrada
Uma mensagem que chega em um destino é entregue a um MDB que, então, processa a mensagem.

Os aplicativos Java EE usam MDBs para processar mensagens assincronamente. Um MDB age como um listener de mensagens JMS e é implementado por um método onMessage(), que define como uma mensagem é processada. Um MDB está implementado no contêiner EJB de um servidor de aplicativos. A maneira precisa na qual um MDB é configurado depende de qual servidor de aplicativos que está sendo usado, mas as informações de configuração devem especificar a qual gerenciador de

filas conectar-se, como conectar-se ao gerenciador de filas, qual destino monitorar para mensagens e o comportamento transacional do MDB. Estas informações são então usadas pelo contêiner EJB. Quando uma mensagem que satisfaz os critérios de seleção do MDB chega ao destino especificado, o contêiner EJB usa as classes WebSphere MQ para JMS para recuperar a mensagem do gerenciador de filas e, em seguida, entrega a mensagem ao MDB chamando seu método onMessage().

Classes IBM WebSphere MQ para arquitetura JMS

As classes IBM WebSphere MQ para JMS, conforme fornecidas na IBM WebSphere MQ Versão 7.0e liberações subsequentes, contêm vários aprimoramentos em comparação com liberações anteriores. Alguns desses aprimoramentos são como resultado de mudanças na implementação de classes IBM WebSphere MQ para JMS e alguns são como resultado de classes IBM WebSphere MQ para JMS explorando mudanças na função IBM WebSphere MQ subjacente.

As seções a seguir resumem os aprimoramentos de chave

Uma arquitetura em camadas

Em liberações anteriores do WebSphere MQ, a implementação de WebSphere MQ classes para JMS foi inteiramente específica para WebSphere MQ. Outros produtos IBM que fornecem sistemas de mensagens também incluíram provedores JMS, mas esses provedores JMS têm muito pouco ou nada em comum com a implementação das classes do WebSphere MQ para JMS

Em WebSphere MQ V7.0, WebSphere MQ classes para JMS tem uma arquitetura em camadas. A camada superior do código é uma camada comum que pode ser usada por qualquer provedor JMS IBM . Quando um aplicativo chama um método JMS, qualquer processamento da chamada que não é específico para um sistema de mensagens é executado pela camada comum, que também fornece uma resposta consistente para a chamada. Qualquer processamento da chamada que seja específico a um sistema de mensagens é delegado a uma camada inferior. A Figura 125 na página 809 mostra a arquitetura em camadas

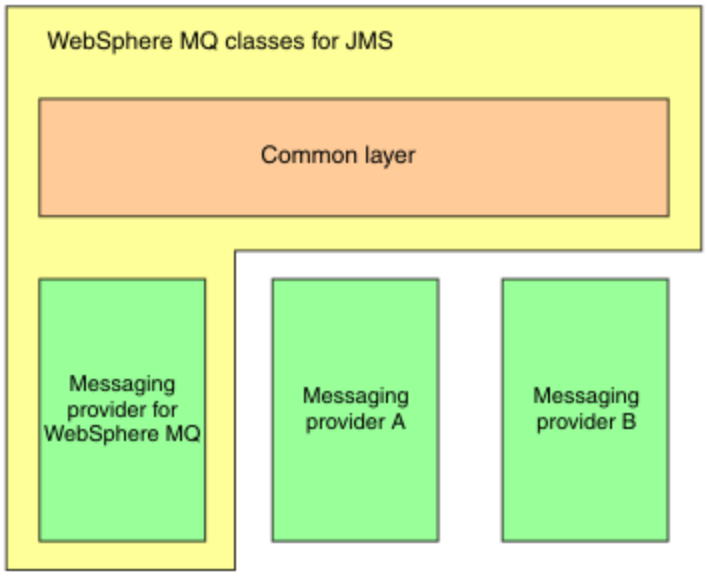


Figura 125. A arquitetura em camadas para provedores JMS IBM

A mudança para uma arquitetura em camadas tem os seguintes objetivos:

- Para melhorar a consistência do comportamento dos vários provedores JMS do IBM
- Para facilitar a gravação de um aplicativo de ponte entre dois sistemas de mensagens IBM
- Para facilitar a porta de um aplicativo de um provedor JMS do IBM para outro

Essa implementação de classes do WebSphere MQ para JMS também introduz um novo conjunto de extensões para a API JMS Essas extensões são conhecidas como *IBM extensões JMS*. O foco principal dessas extensões é criar e configurar connection factories e destinos dinamicamente no tempo de execução.

Um aplicativo usando as extensões JMS IBM inicia criando um objeto Factory JmsFactory, especificando como um parâmetro uma constante que identifica o sistema de mensagens escolhido. O aplicativo usa o objeto JmsFactoryFactory para criar connection factories e destinos que tenham classes corretas especializadas para o sistema de mensagens escolhido.

O aplicativo pode então configurar os connection factory e os destinos configurando suas propriedades. As extensões JMS IBM fornecem um conjunto de métodos para configurar propriedades. Esses métodos não dependem de qualquer sistema de mensagens. Cada tipo de dados possui seu próprio método configurando e cada propriedade é identificada por um nome, que é definido como um membro final estático da classe WMQConstants. Quando um aplicativo chama um desses métodos, um dos parâmetros na chamada é o nome da propriedade e o outro parâmetro é o valor da propriedade.

Por exemplo, se o WebSphere MQ for o sistema de mensagens, uma das propriedades de um connection factory será o nome do gerenciador de filas ao qual se conectar. Usando as extensões JMS do IBM , um aplicativo configura o nome do gerenciador de filas para JUPITER chamando o método a seguir:

```
JmsConnectionFactory myCF;
...
myCF.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "JUPITER");
```

Em contraste, um aplicativo pode executar a mesma função chamando o método a seguir:

```
MQConnectionFactory myCF;
...
myCF.setQueueManager("JUPITER");
```

Este método é uma extensão JMS do WebSphere MQ e é específico para o WebSphere MQ como o sistema de mensagens O uso desse método, portanto, torna o aplicativo potencialmente menos fácil de transportar para outro provedor JMS do IBM .

O relacionamento entre as classes do WebSphere MQ para JMS e as classes do WebSphere MQ para Java

Nas liberações do WebSphere MQ, anteriores à Versão 7.0, as classes WebSphere MQ para JMS foram implementadas quase inteiramente como uma camada de código sobre as classes WebSphere MQ para Java. Essa disposição causou alguma confusão entre os desenvolvedores de aplicativos porque a configuração de campos ou métodos de chamada na classe MQEnvironment pode causar efeitos indesejados e inesperados no comportamento de tempo de execução do código que é gravado usando classes WebSphere MQ para JMS. Além disso, a implementação das classes do WebSphere MQ para JMS tinha algumas restrições em áreas em que a API JMS não é um ajuste natural em cima das classes do WebSphere MQ para Java, e essas restrições levaram a alguns problemas relacionados ao desempenho do tempo de execução

A partir do WebSphere MQ V7.0, a implementação das classes WebSphere MQ para JMS não é mais dependente das classes WebSphere MQ para Java. WebSphere MQ classes para Java e WebSphere MQ classes para JMS são agora peers que usam uma interface Java comum para o MQI. Essa disposição permite mais escopo para otimização de desempenho e significa que a configuração de campos ou métodos de chamadas na classe MQEnvironment não tem efeito no comportamento de tempo de execução do código que é gravado usando classes WebSphere MQ para JMS. Figura 126 na página 811 mostra o relacionamento entre as classes do WebSphere MQ para JMS e WebSphere MQ para Java em liberações anteriores do WebSphere MQ e no WebSphere MQ V7.0 e liberações subsequentes.

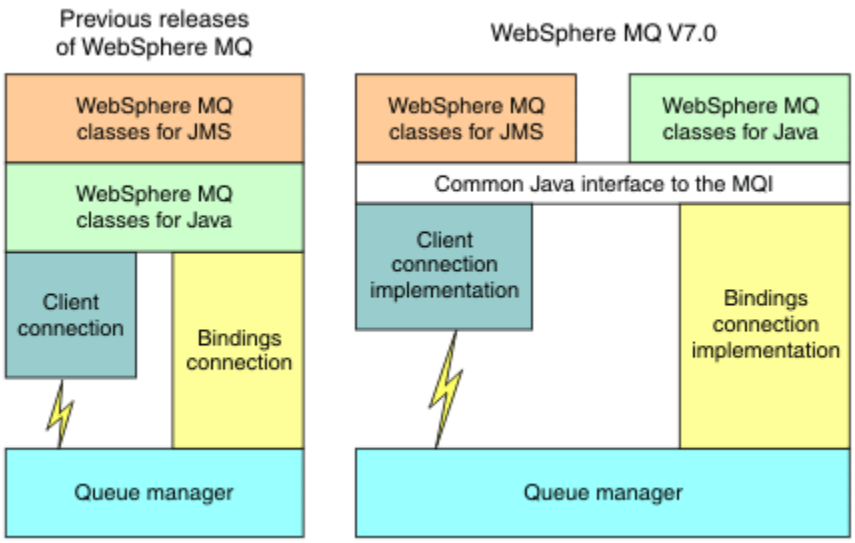


Figura 126. O relacionamento entre as classes do WebSphere MQ para JMS e as classes do WebSphere MQ para Java

Para manter a compatibilidade com liberações anteriores, as classes de saída de canal gravadas em Java ainda podem usar as classes WebSphere MQ para interfaces Java, mesmo se as classes de saída de canal forem chamadas a partir de classes WebSphere MQ para JMS. No entanto, usar as classes WebSphere MQ para interfaces Java significa que seus aplicativos ainda são dependentes das classes WebSphere MQ para o arquivo JAR Java, com.ibm.mq.jar. Se você não desejar com.ibm.mq.jar em seu caminho de classe, poderá usar o novo conjunto de interfaces no pacote com.ibm.mq.exits .

Agora é possível criar e configurar objetos administrados por JMS com o WebSphere MQ Explorer.

Sistema de Mensagens de Publicação/Assinatura

WebSphere MQ V7.0e liberações subsequentes contêm a função de publicação / assinatura integrada. Essa função substitui o WebSphere MQ Publish / Subscribe, que foi fornecido com o WebSphere MQ V6.0.

As classes do WebSphere MQ para aplicativos JMS podem usar a função de publicação / assinatura integrada e podem usá-la em vez de usar o WebSphere Event Broker ou o WebSphere Message Broker para o sistema de mensagens de publicação / assinatura com o WebSphere MQ como o transporte. Configurar as classes do WebSphere MQ para JMS para usar a nova função é mais simples do que configurar as classes do WebSphere MQ para JMS para usar o WebSphere MQ Publish / Subscribe, o WebSphere Event Broker ou o WebSphere Message Broker. Os administradores e desenvolvedores de aplicativos não precisam mais gerenciar as filas de publicação, filas de assinantes, armazenamentos de assinatura e limpeza do assinante. Além disso, os objetos ConnectionFactory e Topic possuem um número menor de propriedades.

A função integrada publicar/assinar também fornece recursos adicionais como publicações retidas e uma opção de dois esquemas curinga para especificar um intervalo de tópicos para os quais um aplicativo deseja se subscrever.

Um aplicativo ainda pode usar uma conexão em tempo real para um broker do WebSphere Event Broker ou WebSphere Message Broker para sistema de mensagens de publicação / assinatura. Este suporte permanece inalterado.

Os aplicativos que usam o WebSphere MQ Publish / Subscribe podem usar a função de publicação / assinatura integrada sem mudança quando o gerenciador de filas ao qual eles estão conectados é atualizado As propriedades configuradas por um aplicativo, mas que não são requeridas pela função publicar/assinar integrada, são ignoradas.

WebSphere MQ provedor de sistemas de mensagens

O provedor de sistemas de mensagens WebSphere MQ tem dois modos de operação:

- *WebSphere MQ modo normal do provedor de sistemas de mensagens*
- *WebSphere MQ modo de migração do provedor de sistemas de mensagens*

O modo normal do provedor de sistemas de mensagens do WebSphere MQ usa todos os recursos do WebSphere MQ Versão 7.0e gerenciadores de filas de liberação subsequentes para implementar JMS. Esse modo é usado apenas para se conectar a um gerenciador de filas do WebSphere MQ e pode se conectar ao WebSphere MQ Versão 7.0e aos gerenciadores de filas de liberação subsequentes no modo cliente ou de ligações. Este modo é otimizado para usar o novo WebSphere MQ Versão 7.0 e função de liberação subsequente.

O modo de migração do provedor de sistemas de mensagens do WebSphere MQ é baseado na função WebSphere MQ Versão 6.0 e usa apenas os recursos que estavam disponíveis no gerenciador de filas do WebSphere MQ Versão 6.0 para implementar JMS. É possível conectar a um WebSphere MQ Versão 7.0 e gerenciadores de filas de liberação subsequentes usando o modo de migração do provedor de sistemas de mensagens do WebSphere MQ , mas não é possível usar nenhuma das otimizações da Versão 7.0 . Este modo permite que conexões a uma das seguintes versões do gerenciador de filas:

1. WebSphere MQ Versão 7.0e subsequente, gerenciador de filas em ligações ou no modo cliente, mas esse modo usa apenas os recursos que estavam disponíveis para um gerenciador de filas do WebSphere MQ Versão 6.0
2. WebSphere MQ Versão 6.0 ou anterior gerenciador de filas no modo cliente

Se desejar se conectar ao WebSphere Event Broker ou ao WebSphere Message Broker usando o WebSphere MQ Enterprise Transport, use o modo de migração do provedor de sistemas de mensagens WebSphere MQ . Se você usar o WebSphere MQ Transporte em Tempo Real, o WebSphere MQ modo de migração do provedor de sistemas de mensagens será selecionado automaticamente, porque você selecionou explicitamente as propriedades no objeto do connection factory. A conexão com o WebSphere Event Broker ou WebSphere Message Broker usando o WebSphere MQ Enterprise Transport segue as regras gerais para seleção de modo descritas em Regras para selecionar o WebSphere MQ modo do provedor de sistemas de mensagens .

Consumo de Mensagem Assíncrona

WebSphere MQ V7.0 e qualquer liberação subsequente suporta consumo de mensagem assíncrona. Um aplicativo pode registrar uma função de retorno de chamada para um destino. Quando uma mensagem adequada é enviada para o destino, WebSphere MQ chama a função e transmite a mensagem como um parâmetro. A função então processa a mensagem de maneira assíncrona. Em liberações anteriores do WebSphere MQ, esse recurso estava disponível apenas ao usar classes WebSphere MQ para JMS.

As classes do WebSphere MQ para JMS foram alteradas para explorar esse novo recurso no WebSphere MQ V7.0 e qualquer liberação subsequente. A implementação de listeners de mensagens JMS agora é um ajuste mais natural com o WebSphere MQe as classes WebSphere MQ para JMS não têm mais que pesquisar um destino para verificar se uma mensagem adequada foi enviada para o destino. Como resultado, o desempenho de listeners de mensagens JMS é melhorado, particularmente quando um aplicativo usa vários listeners de mensagem em uma sessão para monitorar diversos destinos O rendimento da mensagem é aumentado e o tempo gasto para entregar uma mensagem a um listener de mensagem após ele ter chegado a um destino é reduzido

Os Message Driven Beans (MDBs) possuem melhorias de desempenho semelhantes. Além disso, devido a outro aprimoramento da função WebSphere MQ , vários MDBs que estão consumindo mensagens do mesmo destino agora experimentam contenção reduzida nas mensagens.

Seleção de Mensagem

Com exceção da seleção de mensagens por identificador de mensagens ou identificador de correlação, toda a seleção de mensagens em releases do WebSphere MQ anteriores à versão 7.0 foi feita por classes

do WebSphere MQ para JMS. No WebSphere MQ V7.0, e em qualquer liberação subsequente, toda a seleção de mensagens é feita pelo gerenciador de fila

Como resultado, o rendimento de mensagem é aumentado para aplicativos que consomem mensagens usando a seleção de mensagens. A melhoria de desempenho é maior para um aplicativo que se conecta no modo cliente porque apenas as mensagens que satisfazem os critérios de seleção são transportadas pela rede e as classes do WebSphere MQ para JMS manipulam apenas as mensagens que ele entrega ao aplicativo.

Compartilhando uma Conexão de Comunicações

Em liberações anteriores do WebSphere MQ, se um aplicativo cliente WebSphere MQ conectado a um gerenciador de filas mais de uma vez usando o mesmo canal MQI, cada instância do canal MQI requereu uma conexão TCP separada. No WebSphere MQ V7.0 e em qualquer liberação subsequente, cada conexão com o gerenciador de filas usando o mesmo canal MQI pode compartilhar uma única conexão TCP. Essa disposição significa que menos recursos de rede são necessários e o tempo total gasto para criar diversas conexões com o gerenciador de filas é reduzido, particularmente ao usar SSL porque o handshake SSL ocorre apenas uma vez no início da conexão TCP

WebSphere MQ classes para JMS explora este aprimoramento. Para um aplicativo que se conecta a um gerenciador de filas no modo cliente, as classes WebSphere MQ para JMS podem criar mais de uma conexão com um gerenciador de filas usando o canal MQI com o nome especificado como uma propriedade do objeto ConnectionFactory . Cada uma dessas conexões ao gerenciador de filas agora pode compartilhar uma única conexão TCP.

Leitura Antecipada nas Conexões do Cliente

Se um aplicativo usar uma conexão do cliente para consumir mensagens não persistentes de um destino, o destino poderá ser configurado para que as classes do WebSphere MQ para JMS usem um buffer para armazenar as mensagens de interesse antes de entregá-las ao aplicativo.. Esta otimização é denominada *leitura antecipada* e pode ser usada por aplicativos que consomem as mensagens de maneira síncrona chamando o método receive() e pelos listeners de mensagem e MDBs, que consomem as mensagens de maneira assíncrona. A leitura antecipada é especialmente efetiva para destinos com um grande número de mensagens que precisam ser consumidas rapidamente.

A leitura antecipada não se aplica a mensagens persistentes porque, se as mensagens persistentes fossem lidas em um buffer, o gerenciador de filas não conseguiria mais recuperar as mensagens seguindo uma falha. No entanto, um aplicativo que consome as mensagens a partir de um destino com uma mistura de mensagens persistentes e não persistentes ainda podem usar a leitura antecipada. A ordem das mensagens é preservada, mas o tempo de execução se beneficia da leitura antecipada apenas para mensagens não persistentes.

Ao decidir se deve usar a leitura antecipada, considere os seguintes pontos:

- Se um aplicativo estiver consumindo mensagens de um destino que esteja configurado para leitura antecipada e o aplicativo terminar por alguma razão, quaisquer mensagens não persistentes que estiverem atualmente armazenadas no buffer serão descartadas.
- Se todas as seguintes condições forem verdadeiras, as mensagens enviadas a uma fila em uma sessão podem não ser recebidas na ordem em que foram enviadas:
  - Um aplicativo usa dois consumidores de mensagens na mesma sessão para consumir as mensagens da fila.
  - Cada consumidor de mensagens usa um objeto Destino diferente para a fila.
  - Qualquer um ou ambos os objetos Destino são configurados para leitura antecipada.

Enviando mensagens

Quando um aplicativo envia mensagens para um destino, o destino pode ser configurado para que, quando o aplicativo chama send (), WebSphere MQ classes para JMS encaminha a mensagem para o gerenciador de filas e retorna o controle de volta para o aplicativo sem determinar se o gerenciador de

filas recebeu a mensagem com segurança. As classes do WebSphere MQ para JMS podem funcionar dessa maneira somente para mensagens não persistentes e para mensagens persistentes enviadas em uma sessão transacionadas

Para mensagens persistentes enviadas em uma sessão transacionada, o aplicativo finalmente determina se o gerenciador de filas recebeu as mensagens com segurança quando chama commit (). Para quaisquer mensagens enviadas em uma sessão que não sejam transacionadas, a propriedade SENDCHECKCOUNT do objeto ConnectionFactory especifica quantas mensagens devem ser enviadas antes das classes WebSphere MQ para JMS verificar se o gerenciador de filas recebeu as mensagens com segurança.

Esta otimização é mais benéfica para um aplicativo que se conecta a um gerenciador de filas no modo cliente e precisa enviar uma sequência de mensagens em sucessão rápida, mas não requer feedback imediato do gerenciador de filas para cada mensagem enviada.

Saídas do canal

Quando chamado a partir das classes do WebSphere MQ para JMS, os programas de saída do canal gravados em C ou C++ agora se comportam da mesma maneira que quando são chamados a partir de um cliente MQI do Websphere MQ . O desempenho das classes de saída de canal gravadas em Java foi melhorado e agora é possível gravar classes de saída de canal usando um novo conjunto de interface no pacote com.ibm.mq.exits em vez de usar as interfaces nas classes WebSphere MQ para Java.

Propriedades da Mensagem

Uma mensagem JMS consiste em um conjunto de campos de cabeçalho, um conjunto de propriedades e um corpo que contém dados do aplicativo. Como um mínimo, uma mensagem do WebSphere MQ consiste em um descritor de mensagens e nos dados do aplicativo

Quando um WebSphere MQ classes para aplicativo JMS envia uma mensagem JMS, WebSphere MQ classes para JMS mapeia a mensagem JMS em uma mensagem WebSphere MQ . Alguns dos campos e propriedades do cabeçalho JMS são mapeados em campos no descritor de mensagens e alguns são mapeados em campos em um cabeçalho adicional do WebSphere MQ chamado de cabeçalho MQRFH2 . Quando um WebSphere MQ classes para aplicativo JMS recebe uma mensagem JMS, o WebSphere MQ classes para JMS executa o mapeamento reverso.

Um aplicativo que está usando o MQI para receber mensagens de um WebSphere MQ classes para o aplicativo JMS deve, portanto, ser capaz de manipular um cabeçalho MQRFH2 . Se o aplicativo não puder manipular um cabeçalho MQRFH2 , a propriedade TARGCLIENT do objeto de Destino poderá ser configurada para informar às classes do WebSphere MQ para que o JMS não inclua um cabeçalho MQRFH2 nas mensagens do WebSphere MQ . No entanto, ao excluir o cabeçalho MQRFH2 , as informações mantidas em alguns dos campos e propriedades do cabeçalho JMS são perdidas.

Da mesma forma, um aplicativo que está usando o MQI para enviar mensagens para um WebSphere MQ classes para aplicativo JMS deve incluir um cabeçalho MQRFH2 em cada mensagem. Se um cabeçalho MQRFH2 não for incluído, as classes do WebSphere MQ para JMS poderão configurar apenas os campos de cabeçalho JMS e as propriedades que podem ser derivados dos campos em um descritor de mensagens.

WebSphere MQ V7.0 fornece algum suporte adicional para aplicativos que usam o MQI para receber mensagens e enviar mensagens para as classes WebSphere MQ para aplicativos JMS.

Quando um aplicativo chama MQGET para receber uma mensagem de uma classe WebSphere MQ para o aplicativo JMS, o aplicativo pode optar por receber a mensagem de uma das seguintes maneiras:

1. A mensagem é entregue com um descritor de mensagens, um cabeçalho MQRFH2 que contém dados derivados de campos e propriedades do cabeçalho JMS e os dados do aplicativo.
2. A mensagem é entregue a um descritor de mensagens, dados do aplicativo e um conjunto de propriedades de mensagem.

Na opção 2, cada propriedade de mensagem representa um campo ou propriedade de cabeçalho JMS que foi originalmente mapeado pelas classes WebSphere MQ para JMS em um campo em um cabeçalho MQRFH2 . Após a chamada MQGET, o aplicativo pode usar a chamada MQINQMP para receber os valores

das propriedades da mensagem. Usar a opção 2 em vez da opção 1 para receber uma mensagem simplifica a lógica de aplicativo das seguintes maneiras:

- O aplicativo não precisa analisar a parte da variável do cabeçalho MQRFH2 , que contém o campo do cabeçalho JMS e os dados da propriedade codificados em um formato XML.
- O aplicativo não tem que converter os dados de caractere na parte variável do cabeçalho MQRFH2.

De forma correspondente, antes de um aplicativo chamar MQPUT para enviar uma mensagem para um WebSphere MQ classes para o aplicativo JMS, o aplicativo pode usar a chamada MQSETMP para configurar os valores de propriedades de mensagem em vez de construir um cabeçalho MQRFH2 .

Capacidade de Manutenção

WebSphere MQ classes para JMS contém uma série de melhorias relacionadas à capacidade de manutenção:

- Rastreamento.

  WebSphere MQ classes para JMS contém uma classe que um aplicativo pode usar para controlar o rastreio. Um aplicativo pode iniciar e parar o rastreio, especificar o nível necessário de detalhes em um rastreio, e customizar a saída de rastreio de várias maneiras ..
- Criação de log

  WebSphere MQ classes para JMS mantém um arquivo de log, que contém mensagens sobre erros que você precisa corrigir. As mensagens são escritas em texto simples. WebSphere MQ classes para JMS contém uma classe que um aplicativo pode usar para especificar o local do arquivo de log e seu tamanho máximo.
- Tecnologia de Suporte de Primeira Falha (FFST).

  Se ocorrer uma falha grave, as classes WebSphere MQ para JMS geram um relatório FFST em um arquivo FDC. O relatório FFST contém informações que o serviço IBM pode utilizar para diagnosticar o problema mais rapidamente
- Informações sobre a versão

  WebSphere MQ classes para JMS contém uma classe que um aplicativo pode usar para consultar a versão de WebSphere MQ classes para JMS.
- Mensagens de exceção..

  Mensagens de exceção foram aprimoradas para fornecer mais informações sobre as causas de erros e as ações necessárias para corrigir erros.
- Servidores de aplicativos..

  A integração dos recursos de capacidade de manutenção das classes do WebSphere MQ para JMS com aqueles do servidor de aplicativos WebSphere foi melhorada

O MQC Foi Substituído por MQConstants

Um novo pacote, com.ibm.mq.constants, é fornecido com IBM WebSphere MQ Versão 7.0. Este pacote contém a classe MQConstants, que implementa várias interfaces. MQConstants contém definições de todas as constantes que estavam na interface do MQC e de várias constantes novas. As interfaces neste pacote seguem de perto os nomes dos arquivos de cabeçalho de constantes usados em IBM WebSphere MQ

Por exemplo, a interface CMQC contém uma constante MQOO_INPUT_SHARED; esta interface e a constante correspondem ao arquivo de cabeçalho `cmqc.h` e a constante MQOO_INPUT_SHARED.

com.ibm.mq.constants pode ser usado com classes IBM WebSphere MQ para Java e classes IBM WebSphere MQ para JMS.

O MQC ainda está presente e possui as constantes que possuía anteriormente. No entanto, para todos os novos aplicativos, deve-se usar o pacote com.ibm.mq.constants.

Gravando classes do WebSphere MQ para aplicativos JMS

Após uma breve introdução ao modelo JMS, este tópico fornece orientação detalhada sobre como gravar classes WebSphere MQ para aplicativos JMS.

O modelo de JMS

O modelo JMS define um conjunto de interfaces que os aplicativos Java podem usar para executar operações de sistema de mensagens WebSphere MQ classes para JMS, como um provedor JMS, define como os objetos JMS são relacionados aos conceitos do WebSphere MQ . A especificação JMS espera que determinados objetos JMS sejam objetos administrados..

A especificação JMS e o pacote javax.jms definem um conjunto de interfaces que os aplicativos Java podem usar para executar operações do sistema de mensagens A lista a seguir resume as interfaces JMS principais:

Destino
Um destino é para onde um aplicativo envia mensagens ou é uma origem da qual um aplicativo recebe mensagens, ou ambos.

ConnectionFactory
Um objeto ConnectionFactory contém um conjunto de propriedades de configuração para uma conexão. Um aplicativo usa um connection factory para criar uma conexão.

Conexão
Um objeto Connection contém a conexão ativa de um aplicativo com um servidor de sistema de mensagens. Um aplicativo usa uma conexão para criar sessões.

Session
Uma sessão é um único contexto encadeado para enviar e receber mensagens. Um aplicativo usa uma sessão para criar mensagens, produtores de mensagens e consumidores de mensagens. Uma sessão é transacionada ou não transacionada.

Mensagem
Um objeto Message contém uma mensagem que um aplicativo envia ou recebe.

MessageProducer
Um aplicativo usa um produtor de mensagem para enviar mensagens para um destino.

MessageConsumer
Um aplicativo usa um consumidor de mensagem para receber mensagens enviadas a um destino.

Figura 127 na página 816 mostra esses objetos e seus relacionamentos.

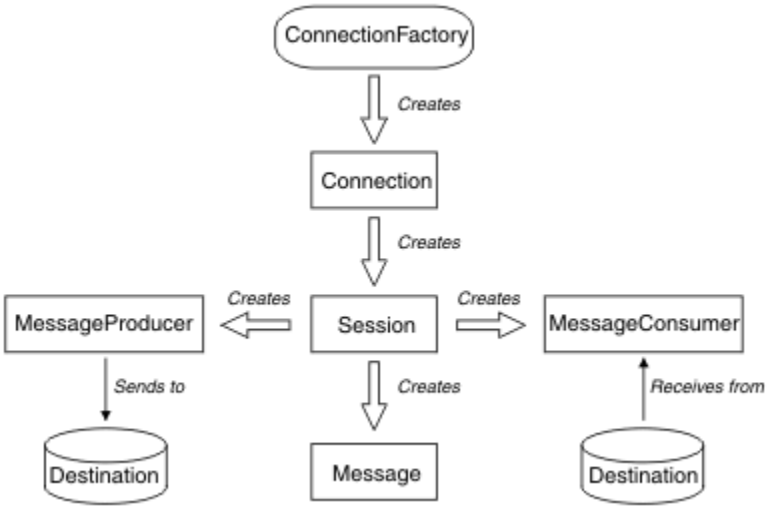


Figura 127. Objetos JMS e seus Relacionamentos

Um objeto Destination, ConnectionFactory ou Connection pode ser utilizado simultaneamente por diferentes encadeamentos de um aplicativo multiencadeado, mas um objeto Session, MessageProducer ou MessageConsumer não pode ser usado simultaneamente por diferentes encadeamentos. A maneira mais simples de garantir que um objeto Session, MessageProducer ou MessageConsumer não é usado simultaneamente é criar um objeto Session separado para cada encadeamento.

O JMS suporta dois estilos de sistema de mensagens:

- Sistema de mensagens ponto a ponto
- Sistema de Mensagens de Publicação/Assinatura

Esses estilos de sistema de mensagens também são referidos como *domínios do sistema de mensagens* e é possível combinar ambos os estilos do sistema de mensagens em um aplicativo. No domínio ponto a ponto, um destino é uma fila e, no domínio de publicar/assinar, um destino é um tópico.

Com versões do JMS antes do JMS 1.1, a programação para o domínio ponto a ponto usa um conjunto de interfaces e métodos e a programação para o domínio de publicação / assinatura usa outro conjunto. Os dois conjuntos são semelhantes, mas separados. Com JMS 1.1, é possível usar um conjunto comum de interfaces e métodos que suportam ambos os domínios do sistema de mensagens. As interfaces comuns fornecem uma visualização independente de cada domínio de sistema de mensagens. Tabela 103 na página 817 lista as interfaces independentes de domínio JMS e suas interfaces específicas de domínio correspondentes

Tabela 103. O domínio JMS independente e interfaces específicas do domínio

Interfaces independentes de domínio	**Interfaces específicas de domínio para o domínio ponto a ponto**	**Interfaces específicas de domínio para o domínio de publicar/assinar**
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
Conexão	QueueConnection	TopicConnection
Destino	Fila	Tópico
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver QueueBrowser	TopicSubscriber

O JMS 1.1 retém todas as interfaces específicas do domínio e, portanto, os aplicativos existentes ainda podem usar essas interfaces Para novos aplicativos, no entanto, considere o uso das interfaces independentes de domínio

Nas classes WebSphere MQ para JMS, os objetos JMS são relacionados aos conceitos do WebSphere MQ das seguintes maneiras:

- Um objeto Connection tem propriedades derivadas das propriedades do connection factory que foi usado para criar a conexão. Essas propriedades controlam como um aplicativo se conecta a um gerenciador de filas. Os exemplos dessas propriedades são o nome do gerenciador de filas e, para um aplicativo que se conecta ao gerenciador de filas no modo cliente, o nome do host ou o endereço IP do sistema em que o gerenciador de filas está em execução.
- Um objeto Session encapsula uma manipulação de conexões do WebSphere MQ , que, portanto, define o escopo transacional da sessão.
- Um objeto MessageProducer e um objeto MessageConsumer cada um encapsula uma manipulação de objetos WebSphere MQ .

Ao usar as classes do WebSphere MQ para JMS, todas as regras normais do WebSphere MQ se aplicam Observe, especificamente, que um aplicativo pode enviar uma mensagem a uma fila remota, mas pode

receber uma mensagem somente de uma fila de propriedade do gerenciador de filas ao qual o aplicativo está conectado.

A especificação JMS espera que ConnectionFactory e objetos de Destino sejam objetos administrados. Um administrador cria e mantém objetos administrados em um repositório central e um aplicativo JMS recupera esses objetos usando a JNDI (Java Naming and Directory Interface)

Nas classes WebSphere MQ para JMS, a implementação da interface de Destino é uma superclasse abstrata de Fila e Tópico e, portanto, uma instância de Destino é um objeto de Fila ou um objeto de Tópico. As interfaces independentes de domínio tratam uma fila ou um tópico como um destino. O domínio do sistema de mensagens para um objeto MessageProducer ou MessageConsumer é determinado por se o destino é uma fila ou um tópico.

No WebSphere MQ classes para JMS, portanto, os objetos dos seguintes tipos podem ser objetos administrados:

- ConnectionFactory
- QueueConnectionFactory
- TopicConnectionFactory
- Fila
- Tópico
- XAConnectionFactory
- XAQueueConnectionFactory
- XATopicConnectionFactory

Mensagens JMS

As mensagens JMS são compostas de um cabeçalho, propriedades e um corpo. O JMS define cinco tipos de corpo da mensagem

As mensagens JMS são compostas das seguintes partes:

Cabeçalho
Todas as mensagens suportam o mesmo conjunto de campos de cabeçalho. Os campos de cabeçalho contêm valores que são usados pelos clientes e provedores para identificar e rotear as mensagens.

Propriedades
Cada mensagem contém um recurso integrado para suportar os valores de propriedade definidos pelo aplicativo. As propriedades fornecem um mecanismo eficiente para filtrar as mensagens definidas pelo aplicativo.

Conteúdo
O JMS define vários tipos de corpo da mensagem que cobrem a maioria dos estilos de mensagens atualmente em uso.

JMS define cinco tipos de corpo de mensagem:

Fluxo
Um fluxo de valores primitivos de Java É preenchido e lido sequencialmente.

Mapear
Um conjunto de pares nome-valor, em que nomes são sequências e valores são tipos primitivos Java. As entradas podem ser acessadas sequencialmente ou aleatoriamente pelo nome. A ordem das entradas é indefinida.

text
Uma mensagem que contém um java.lang.String.

Object
Uma mensagem que contém um objeto Java serializável

bytes
Um fluxo de bytes não interpretados. Este tipo de mensagem é para a codificação literal de um corpo para corresponder ao formato de mensagem existente.

O campo de cabeçalho JMSCorrelationID é usado para vincular uma mensagem à outra. Geralmente, ele vincula uma mensagem de resposta com sua mensagem de solicitação. O JMSCorrelationID pode reter um ID de mensagem específico do provedor, um Sequência específica do aplicativo ou um valor byte[] nativo do provedor.

Seletores de mensagens no JMS..

As mensagens podem conter os valores de propriedades definidos pelo aplicativo. Um aplicativo pode usar seletores de mensagens para ter mensagens de filtro de um provedor JMS

Uma mensagem contém um recurso integrado para suportar os valores de propriedades definidos pelo aplicativo. Efetivamente, isso fornece um mecanismo para incluir campos de cabeçalho específicos do aplicativo em uma mensagem. As propriedades permitem que um aplicativo, usando seletores de mensagens, tenha um provedor JMS selecione ou filtre mensagens em seu nome, usando critérios específicos do aplicativo Propriedades definidas pelo aplicativo devem obedecer às regras a seguir:

- Os nomes de propriedades devem obedecer às regras para um identificador de seletor de mensagem.
- Os valores de propriedades podem ser Boolean, byte, short, int, long, float, double e String.
- Os prefixos de nome JMSX e JMS_ são reservados.

Os valores de propriedades são configurados antes de enviar uma mensagem. Quando um cliente recebe uma mensagem, as propriedades da mensagem são somente leitura. Se um cliente tentar configurar propriedades neste ponto, uma MessageNotWriteableException será lançada. Se clearProperties for chamado, as propriedades agora podem ser lidas e gravadas.

Um valor de propriedade pode duplicar um valor em um corpo da mensagem. O JMS não define uma política para o que pode ser feito em uma propriedade No entanto, os desenvolvedores de aplicativos devem estar cientes que os provedores JMS provavelmente manipulam dados em um corpo da mensagem de forma mais eficiente do que os dados nas propriedades da mensagem Para melhor desempenho, os aplicativos devem usar as propriedades de mensagem somente quando precisam customizar um cabeçalho de mensagem. O motivo principal para fazer isso é suportar a seleção de mensagens customizadas.

Um seletor de mensagem JMS permite que um cliente especifique as mensagens de seu interesse usando o cabeçalho da mensagem. Somente mensagens com cabeçalhos que correspondam ao seletor serão entregues.

os seletores de mensagens não podem fazer referência a valores do corpo da mensagem.

Um seletor de mensagem corresponde a uma mensagem quando o seletor é avaliado como verdadeiro quando o campo de cabeçalho da mensagem e os valores de propriedades são substituídos por seus identificadores correspondentes no seletor.

Um seletor de mensagem é uma String, com sintaxe baseada em um subconjunto da sintaxe de expressão condicional SQL92. A ordem na qual um seletor de mensagem é avaliado é da esquerda para a direita dentro de um nível de precedência. É possível usar parênteses para mudar essa ordem. Literais do seletor e nomes de operadores predefinidos estão gravados aqui em maiúsculas; no entanto, não fazem distinção entre maiúsculas e minúsculas.

Um seletor pode conter:

- Literais
  - Uma sequência literal é colocada entre aspas. Aspas duplas representam aspas simples. Os exemplos são: 'literal' e 'literal''s'. Como literais de sequência Java, eles usam a codificação de caracteres Unicode
  - Um literal numérico exato é um valor numérico sem um ponto decimal, como 57, -957 e +62. Os números no intervalo de Java longo são suportados
  - Um literal numérico aproximado é um valor numérico em notação científica, como 7E3 ou -57.9E2, ou um valor numérico com um decimal, como 7., -95,7 ou +6,2. Os números no intervalo de Java duplo são suportados
  - Os literais booleanos TRUE e FALSE.

- Identificadores:
  - Um identificador é uma sequência de comprimento ilimitado de letras e dígitos Java, o primeiro dos quais deve ser uma letra Java. Uma letra é qualquer caractere para o qual o método Character.isJavaLetter retorna true Isso inclui _ e $. Uma letra ou dígito é qualquer caractere pelo qual o método Character.isJavaLetterOrDigit retorna true.
  - Identificadores não podem ser os nomes NULL, TRUE ou FALSE.
  - Identificadores não podem ser NOT, AND, OR, BETWEEN, LIKE, IN ou IS.
  - Identificadores são referências de campo de cabeçalho ou referências de propriedade.
  - Identificadores fazem distinção entre maiúsculas e minúsculas.
  - Referências de campo de cabeçalho da mensagem são restritas a:
    - JMSDeliveryMode
    - JMSPriority
    - JMSMessageID
    - JMSTimestamp
    - JMSCorrelationID
    - JMSType

    Os valores JMSMessageID, JMSTimestamp, JMSCorrelationID e JMSType podem ser nulos e, se forem, serão tratados como um valor NULL.
  - Qualquer nome que comece com JMSX é um nome de propriedade definido pelo JMS.
  - Qualquer nome que começa com JMS_ é um nome da propriedade específico do provedor.
  - Qualquer nome que não comece com JMS é um nome de propriedade específico do aplicativo Se houver uma referência a uma propriedade que não existe em uma mensagem, seu valor é NULL. Se existir, seu valor é o valor de propriedade correspondente.
- O espaço em branco é o mesmo definido para Java: espaço, guia horizontal, avanço de formulário e terminador de linha.
- Expressões:
  - Um seletor é uma expressão condicional. Um seletor avaliado como true tem correspondência; um seletor avaliado como false ou unknown não tem correspondência.
  - As expressões aritméticas são compostas de si mesmas, operações aritméticas, identificadores (com um valor que é tratado como um literal numérico) e literais numéricos.
  - Expressões condicionais são compostas por si mesmas, por operações de comparação e por operações lógicas.
- O uso padrão de parênteses () para configurar a ordem na qual as expressões são avaliadas é suportado.
- Os operadores lógicos em ordem de precedência: NOT, AND, OR.
- Operadores de comparação: =, >, >=, <, <=, <> (diferente).
  - Somente valores do mesmo tipo podem ser comparados. Uma exceção é que a validade de comparar valores numéricos exatos e valores numéricos aproximados. (A conversão de tipo necessária é definida pelas regras de promoção numérica Java. Se houver uma tentativa de comparar tipos diferentes, o seletor será sempre false.
  - A comparação de String e Boolean é restrita a = e <>. Duas sequências serão iguais apenas se contiverem a mesma sequência de caracteres.
- Operadores aritméticos em ordem de precedência:
  - +, - unário.
  - *, /, multiplicação e divisão.
  - +, -, soma e subtração.

  - Operações aritméticas em um valor NULL não são suportadas. Se forem tentadas, o seletor completo será sempre false.
  - Operações aritméticas devem usar promoção numérica Java.
- Operador de comparação arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 and arithmetic-expr3:
  - Age BETWEEN 15 and 19 é equivalente a age >= 15 AND age <= 19.
  - A idade NÃO ENTRE 15 e 19 é equivalente à idade < 15 OU idade> 19.
  - Se qualquer uma das expressões de uma operação BETWEEN for NULL, o valor da operação será false. Se qualquer uma das expressões de uma operação NOT BETWEEN for NULL, o valor da operação será true.
- identificador [NOT] IN (string-literal1, string-literal2, ...) o operador de comparação em que o identifier tem um valor String ou NULL.
  - Country IN ('UK', 'US', 'France') é true para 'UK' e false para 'Peru'. Ele é equivalente à expressão (Country = 'UK') OR (Country = 'US') OR (Country = 'France').
  - Country NOT IN ('UK', 'US', 'France') é false para 'UK' e true para 'Peru'. Ele é equivalente à expressão NOT ((Country = 'UK') OR (Country = 'US') OR (Country = 'France')).
  - Se o identificador de uma operação IN ou NOT IN for NULL, o valor da operação será desconhecido.
- Operador de comparação identifier [NOT] LIKE pattern-value [ESCAPE escape-character], em que identifier tem um valor de sequência. pattern-value é um literal de sequência, em que _ representa qualquer caractere único e % representa qualquer sequência de caracteres (incluindo a sequência vazia). Todos os outros caracteres representam eles mesmos. O escape-character opcional é uma sequência literal de caractere único, com um caractere usado para escapar o significado especial de _ e % no pattern-value.
  - phone LIKE '12%3' é true para 123 e 12993 e false para 1234.
  - word LIKE 'l_se' é true para "lose" e false para "loose".
  - underscored LIKE '_%' ESCAPE '\' é true para "_foo" e false para "bar".
  - phone NOT LIKE '12%3' é false para 123 e 12993 e true para 1234.
  - Se o identificador de uma operação LIKE ou NOT LIKE for NULL, o valor da operação será desconhecido.
- O operador de comparação identifier IS NULL testa para um valor de campo de cabeçalho nulo ou um valor de propriedade ausente.
  - prop_name IS NULL.
- O operador de comparação identifier IS NOT NULL testa a existência de um valor de campo de cabeçalho ou um valor de propriedade não nulo.
  - prop_name IS NOT NULL.

O seletor de mensagem a seguir seleciona mensagens com um tipo de mensagem de carro, cor azul e peso maior que 2500 lbs:

```
"JMSType = 'car' AND color = 'blue' AND weight > 2500"
```

Conforme observado na lista anterior, os valores de propriedades podem ser NULL. A avaliação de expressões do seletor que contêm valores NULL é definida pela semântica de SQL 92 NULL. A lista a seguir fornece uma breve descrição dessa semântica:

- SQL trata um valor NULL como desconhecido.
- Comparação ou aritmética com um valor desconhecido sempre resulta em um valor desconhecido.
- O operador IS NULL converte um valor desconhecido em um valor TRUE.
- O operador IS NOT NULL converte um valor desconhecido em um valor FALSE.

Embora o SQL suporte comparação decimal fixa e aritmética, os seletores de mensagens JMS não. Por isso literais numéricos exatos são restritos àqueles sem um decimal. Por isso também há valores numéricos com um decimal como uma representação alternativa para um valor numérico aproximado.

Comentários SQL não são suportados.

Mapeando mensagens JMS para mensagens do WebSphere MQ

As mensagens do WebSphere MQ são compostas de um Descritor de Mensagens, um cabeçalho MQRFH2 opcional e um corpo. O conteúdo de uma mensagem JMS é parcialmente mapeado e parcialmente copiado para uma mensagem do WebSphere MQ

Este tópico descreve como a estrutura da mensagem JMS descrita na primeira parte desta seção é mapeada para uma mensagem do WebSphere MQ . É de interesse dos programadores que desejam transmitir mensagens entre o JMS e os aplicativos tradicionais do WebSphere MQ . Ele também é de interesse para pessoas que desejam manipular mensagens transmitidas entre dois aplicativos JMS, por exemplo, em uma implementação do Message Broker do WebSphere

Esta seção não se aplica se um aplicativo usar uma conexão em tempo real com um broker. Quando um aplicativo usa uma conexão em tempo real, toda a comunicação é executada diretamente sobre TCP/IP; nenhuma fila ou mensagem do WebSphere MQ está envolvida.

WebSphere MQ mensagens são compostas de três componentes:

- O Descritor de Mensagens (MQMD) do WebSphere MQ
- Um cabeçalho WebSphere MQ MQRFH2
- O corpo da mensagem.

O MQRFH2 é opcional e sua inclusão em uma mensagem de saída é controlada por uma sinalização na classe Destino JMS. É possível configurar essa sinalização usando a ferramenta de administração JMS do WebSphere MQ . Como o MQRFH2 carrega informações específicas do JMS, sempre inclua-as na mensagem quando o emissor souber que o destino de recebimento é um aplicativo JMS. Normalmente, omita MQRFH2 ao enviar uma mensagem diretamente para um aplicativo não JMS. Isso ocorre porque tal aplicativo não espera um MQRFH2 em sua WebSphere MQ mensagem.

Se uma mensagem recebida não tiver um cabeçalho MQRFH2, o objeto Fila ou Tópico do campo de cabeçalho derivado do campo de cabeçalho JMSReplyTo da mensagem, por padrão, terá este sinalizador configurado de modo que uma mensagem de resposta enviada à fila ou tópico também não tenha um cabeçalho MQRFH2. Será possível desativar esse comportamento de inclusão de um cabeçalho MQRFH2 em uma mensagem de resposta somente se a mensagem original tiver um cabeçalho MQRFH2, configurando a propriedade TARGCLIENTMATCHING do connection factory para NO.

Figura 128 na página 822 mostra como a estrutura de uma mensagem JMS é transformada em uma mensagem WebSphere MQ e novamente:

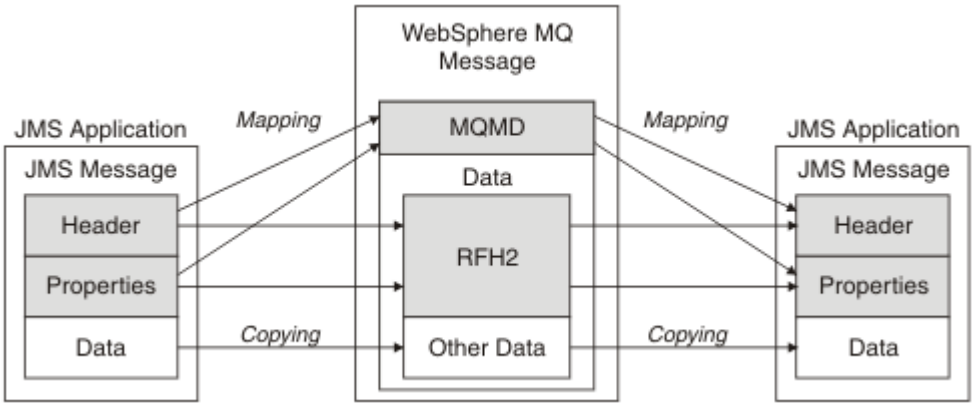


Figura 128. Como as mensagens são transformadas entre JMS e WebSphere MQ usando o cabeçalho MQRFH2

As estruturas são transformadas de duas maneiras:

Mapping
Onde o MQMD inclui um campo que é equivalente ao campo JMS, o campo JMS é mapeado no campo MQMD. Os campos MQMD adicionais são expostos como propriedades JMS, pois um aplicativo JMS pode precisar obter ou configurar esses campos ao se comunicar com um aplicativo não JMS

Copiando
Onde não houver MQMD equivalente, um campo ou propriedade do cabeçalho JMS é transmitido, possivelmente transformado, como um campo dentro do MQRFH2.

O cabeçalho MQRFH2 e o JMS
Esta coleção de tópicos descreve o cabeçalho MQRFH Versão 2, que transporta dados específicos do JMS associados ao conteúdo da mensagem. O MQRFH2 Versão 2 é um cabeçalho extensível e também pode transportar informações adicionais que não estão diretamente associadas com o JMS.. No entanto, esta seção abrange apenas seu uso pelo JMS..Para obter uma descrição completa, consulte MQRFH2 - Regras e formatação do cabeçalho 2.

Há duas partes do cabeçalho, uma parte fixa e uma parte variável.

Parte fixa
A parte fixa é modelada no padrão do cabeçalho *standard* WebSphere MQ e consiste nos seguintes campos:

StrucId (MQCHAR4)
Identificador de estruturação.

Deve ser MQRFH_STRUC_ID (valor: "RFH ") (valor inicial).

MQRFH_STRUC_ID_ARRAY (valor: "R","F","H"," ") também está definido.

Versão (MQLONG)
Número de versão da estrutura.

Deve ser MQRFH_VERSION_2 (valor: 2) (valor inicial).

StrucLength (MQLONG)
Comprimento total de MQRFH2, incluindo os campos NameValueData.

O valor configurado em StrucLength deve ser um múltiplo de 4 (os dados nos campos NameValueData podem ser preenchidos com caracteres de espaço para fazer isso).

Codificação (MQLONG)
Codificação de dados.

Codificação de quaisquer dados numéricos na parte da mensagem após MQRFH2 (o próximo cabeçalho ou os dados da mensagem após esse cabeçalho).

CodedCharSetId (MQLONG)
Identificador do conjunto de caracteres codificados.

Representação de quaisquer dados de caracteres na parte da mensagem após MQRFH2 (o próximo cabeçalho ou os dados da mensagem após esse cabeçalho).

Formato (MQCHAR8)
Nome do formato.

Nome do formato para a parte da mensagem após MQRFH2.

Sinalizadores (MQLONG)
Sinalizadores.

MQRFH_NO_FLAGS =0. Nenhum conjunto de sinalizadores.

NameValueCCSID (MQLONG)
O identificador do conjunto de caracteres codificados (CCSID) para as sequências de caracteres NameValueData contidas neste cabeçalho. O NameValueData pode ser codificado em um conjunto de caracteres que difere das outras sequências de caracteres que estão contidas no cabeçalho (StrucID e Format).

Se o NameValueCCSID for um CCSID Unicode de 2 bytes (1200, 13488 ou 17584), a ordem de bytes do Unicode será a mesma que a ordem de bytes dos campos numéricos no MQRFH2. (Por exemplo, Versão, StrucLength e NameValueCCSID em si.)

Tabela 104. Valores possíveis para o campo NameValueCCSID

Value	Significado
1200	UCS2 abertos-encerrados
1208	UTF8
13488	Subconjunto UCS2 2.0
17584	Subconjunto UCS2 2.1 (inclui o símbolo do euro)

Parte variável

A parte variável segue a parte fixa. A parte variável contém um número variável de pastas do MQRFH2. Cada pasta contém um número variável de elementos ou propriedades. Propriedades relacionadas ao grupo de pastas. Os cabeçalhos MQRFH2 criados pelo JMS podem conter qualquer uma das seguintes pastas:

A pasta <mcd>

O `mcd` contém propriedades que descrevem o formato da mensagem. Por exemplo, a propriedade `Msd` do domínio de serviço de mensagem identifica uma mensagem JMS como sendo `JMSTextMessage`, `JMSBytesMessage`, `JMSStreamMessage`, `JMSMapMessage`, `JMSObjectMessage` ou nula.

A pasta `mcd` está sempre presente em uma mensagem JMS que contém um `MQRFH2`.

Ele está sempre presente em uma mensagem que contém um `MQRFH2` enviado do WebSphere Message Broker Isso descreve o domínio, o formato, o tipo e o conjunto de mensagens de uma mensagem.

Tabela 105. mcd nome da propriedade, sinônimo, tipo de dados e pasta

Sinônimo da Propriedade	Nome da Propriedade	Tipo de Dados	Pasta (Folder)
	`mcd.Msd`	`string`	`<mcd><Msd>`*`messageDomain`*`</Msd></mcd>`
	`mcd.Set`	`string`	`<mcd><Set>`*`messageDomain`*`</Set></mcd>`
	`mcd.Type`	`string`	`<mcd><Type>`*`messageDomain`*`</Type></mcd>`
	`mcd.Fmt`	`string`	`<mcd><Fmt>`*`messageDomain`*`</Fmt></mcd>`

Não inclua suas próprias propriedades na pasta `mcd`.

A pasta <jms>

O `jms` contém campos de cabeçalho JMS, e propriedades JMSX que não são totalmente expressas no `MQMD`. A pasta `jms` está sempre presente em uma JMS `MQRFH2`.

A pasta <usr>

O `usr` contém propriedades JMS definidas por aplicativo associadas à mensagem. A pasta `usr` estará presente apenas quando um aplicativo tiver configurado uma propriedade definida pelo aplicativo.

A pasta <mqext>

`mqext` contém propriedades que são utilizadas apenas pelo WebSphere Application Server A pasta está presente apenas se o aplicativo tiver configurado pelo menos uma das propriedades definidas pela IBM

Tabela 106. `mqext` nome da propriedade, sinônimo, tipo de dados e pasta

Sinônimo da Propriedade	Nome da Propriedade	Tipo de Dados	Pasta (Folder)
`JMSArmCorrelator`	`mqext.Arm`	`string`	`<mqext><Arm>`*`armCorrelator`*`</Arm></mqext>`
`JMSRMCorrelator`	`mqext.Wrm`	`string`	`<mqext><Wrm>`*`wrmCorrelator`*`</Wrm></mqext>`

Não inclua suas próprias propriedades na pasta `mqext`.

A pasta <mqps>

O `mqps` contém propriedades usadas apenas por IBM WebSphere MQ publicar/assinar. A pasta estará presente somente se o aplicativo tiver configurado pelo menos uma das propriedades de publicação/assinatura integradas.

Tabela 107. `mqps` nome da propriedade, sinônimo, tipo de dados e pasta

Sinônimo da Propriedade	Nome da Propriedade	Tipo de Dados	Pasta (Folder)
`MQTopicString`	`mqps.Top`	`string`	`<mqps><Top>`*`topicString`*`</Top></mqps>`
`MQSubUserData`	`mqps.Sud`	`string`	`<mqps><Sud>`*`subscriberUserData`*`</Sud></mqps>`
`MQIsRetained`	`mqps.Ret`	`boolean`	`<mqps><Ret>`*`isRetained`*`</Ret></mqps>`
`MQPubOptions`	`mqps.Pub`	`i8`	`<mqps><Pub>`*`publicationOptions`*`</Pub></mqps>`
`MQPubLevel`	`mqps.Pbl`	`i8`	`<mqps><Pbl>`*`publicationLevel`*`</Pbl></mqps>`
`MQPubTime`	`mqpse.Pts`	`string`	`<mqps><Pts>`*`publicationTime`*`</Pts></mqps>`
`MQPubSeqNum`	`mqpse.Seq`	`i8`	`<mqps><Seq>`*`publicationSequenceNumber`*`</Seq></mqps>`
`MQPubStrIntData`	`mqpse.Sid`	`string`	`<mqps><Sid>`*`publicationData`*`</Sid></mqps>`
`MQPubFormat`	`mqpse.Pfmt`	`i8`	`<mqps><Pfmt>`*`messageFormat`*`</Pfmt></mqps>`

Não inclua suas próprias propriedades na pasta `mqps`.

Tabela 108 na página 826 mostra uma lista completa de nomes de propriedades.

Tabela 108. MQRFH2 pastas e propriedades usadas pelo JMS				
Nome do campo JMS	**Tipo Java**	**Nome da pasta MQRFH2**	**Nome da Propriedade**	**Tipo/valores**
JMSDestination	Destino	jms	Dst	sequência
JMSExpiration	grande	jms	Expand.	i8
JMSPriority	int	jms	Pri	i4
JMSDeliveryMode	int	jms	Dlv	i4
JMSCorrelationID	Sequência	jms	Cid	sequência
JMSReplyTo	Destino	jms	Rto	sequência
JMSTimestamp	grande	jms	Tms	i8
JMSType	Sequência	mcd	Tipo, Configuração, Fmt	sequência
JMSXGroupID	Sequência	jms	Gid	sequência
JMSXGroupSeq	int	jms	Seq.	i4
xxx (definido pelo usuário)	Qualquer	usr	xxx	qualquer
		mcd	Msd	jms_none jms_text jms_bytes jms_map jms_stream jms_object

NameValueLength (MQLONG)
Comprimento em bytes da sequência NameValueData que segue imediatamente este campo de comprimento (não inclui seu próprio comprimento).

NameValueData (MQCHARn)
Uma única sequência de caracteres, cujo comprimento em bytes é dado pelo campo NameValueLength anterior. Contém uma pasta que mantém uma sequência de propriedades. Cada propriedade é um trio de nome/tipo/valor, contido em um elemento XML cujo nome é o nome da pasta, conforme a seguir:

```
 <foldername>
triplet1 triplet2 .....   tripletn </foldername>
```

A tag de fechamento `</foldername>` pode ser seguida por espaços como caracteres de preenchimento. Cada trio é codificado usando uma sintaxe semelhante a XML:

```
   <name dt='datatype'>value</name>
```

O elemento `dt='datatype'` é opcional e é omitido para muitas propriedades, pois o tipo de dado é predefinido. Se ele for incluído, um ou mais caracteres de espaço deverão ser incluídos antes da tag `dt=`.

name
é o nome da propriedade; consulte Tabela 108 na página 826.

datatype
deve corresponder, após compactação, a um dos tipos de dados listados em Tabela 109 na página 827.

value
é uma representação de sequência do valor a ser transmitido, usando as definições em Tabela 109 na página 827.

Um valor nulo é codificado usando a seguinte sintaxe:

```
<name dt='datatype' xsi:nil='true'></name>
```

Não use `xsi:nil='false'`.

Tabela 109. Tipos de dados de propriedade

Tipo de Dados	Definição
sequência	Qualquer sequência de caracteres excluindo < e &
booleano	O caractere `0` ou `1` (`0` = falso, `1` = verdadeiro)
bin.hex	Dígitos hexadecimais que representam octetos
i1	Um número, expresso usando dígitos `0..9`, com sinal opcional (sem frações ou expoente). Deve estar no intervalo de -128 a 127, inclusive
i2	Um número, expresso usando dígitos `0..9`, com sinal opcional (sem frações ou expoente). Deve estar no intervalo de -32768 a 32767, inclusive
i4	Um número, expresso usando dígitos `0..9`, com sinal opcional (sem frações ou expoente). Deve estar no intervalo de -2147483648 a 2147483647, inclusive
i8	Um número, expresso usando dígitos `0..9`, com sinal opcional (sem frações ou expoente). Deve estar no intervalo de -9223372036854775808 a 9223372036854775807, inclusive
int	Um número, expresso usando dígitos `0..9`, com sinal opcional (sem frações ou expoente). Deve estar no mesmo intervalo que i8. Isso pode ser usado no lugar de um dos tipos i* se o emissor não desejar associar uma precisão específica à propriedade
r4	Número de vírgula flutuante, magnitude < = 3.40282347E+38,> = 1.175E-37 expresso usando dígitos `0..9`, sinal opcional, dígitos fracionários opcionais, expoente opcional
r8	Número de vírgula flutuante, magnitude < = 1.7976931348623E+308,> = 2.225E-307 expresso usando dígitos `0..9`, sinal opcional, dígitos fracionários opcionais, expoente opcional

Um valor de sequência pode conter espaços. Deve-se usar as seguintes sequências de escape em um valor de sequência:

- `&` para o caractere &
- < para o caractere <

É possível usar as seguintes sequências de escape, mas elas não são requeridas:

- `>` para o caractere >
- `&apos;` para o caractere '
- `&quot;` para o caractere "

Campos JMS e propriedades com campos MQMD correspondentes
Essas tabelas mostram os campos MQMD equivalentes a campos de cabeçalho JMS, propriedades JMS e propriedades específicas do provedor JMS.

Tabela 110 na página 828 lista os campos de cabeçalho JMS e Tabela 111 na página 828 lista as propriedades JMS mapeadas diretamente para os campos MQMD. Tabela 112 na página 828 lista as propriedades específicas do provedor e os campos MQMD para os quais eles são mapeados.

Tabela 110. Campos de cabeçalho JMS mapeados para campos MQMD

campo de cabeçalho JMS	**Tipo Java**	**Campo MQMD**	**tipo C**
JMSDeliveryMode	int	Persistence	MQLONG
JMSExpiration	grande	Expiração	MQLONG
JMSPriority	int	Priority	MQLONG
JMSMessageID	Sequência	MsgID	MQBYTE24
JMSTimestamp	grande	PutDate PutTime	MQCHAR8 MQCHAR8
JMSCorrelationID	Sequência	CorrelId	MQBYTE24

Tabela 111. Mapeamento de propriedades JMS para campos MQMD

propriedade JMS	**Tipo Java**	**Campo MQMD**	**tipo C**
JMSXUserID	Sequência	UserIdentifier	MQCHAR12
JMSXAppID	Sequência	PutApplName	MQCHAR28
JMSXDeliveryCount	int	BackoutCount	MQLONG
JMSXGroupID	Sequência	GroupId	MQBYTE24
JMSXGroupSeq	int	MsgSeqNumber	MQLONG

Tabela 112. Mapeamento de propriedades específicas do provedor JMS para campos MQMD

Propriedade Específica do Provedor JMS.	**Tipo Java**	**Campo MQMD**	**tipo C**
JMS_IBM_Report_Exception	int	Relatório	MQLONG
JMS_IBM_Report_Expiration	int	Relatório	MQLONG
JMS_IBM_Report_COA	int	Relatório	MQLONG
JMS_IBM_Report_COD	int	Relatório	MQLONG
JMS_IBM_Report_PAN	int	Relatório	MQLONG
JMS_IBM_Report_NAN	int	Relatório	MQLONG
JMS_IBM_Report_Pass_Msg_ID	int	Relatório	MQLONG
JMS_IBM_Report_Pass_Correl_ID	int	Relatório	MQLONG
JMS_IBM_Report_Discard_Msg	int	Relatório	MQLONG
JMS_IBM_MsgType	int	MsgType	MQLONG

Tabela 112. Mapeamento de propriedades específicas do provedor JMS para campos MQMD (continuação)

Propriedade Específica do Provedor JMS.	Tipo Java	Campo MQMD	tipo C
JMS_IBM_Feedback	int	Feedback	MQLONG
JMS_IBM_Format	Sequência	Formatar "1" na página 829	MQCHAR8
JMS_IBM_PutApplType	int	PutApplType	MQLONG
JMS_IBM_Encoding	int	Encoding	MQLONG
JMS_IBM_Character_Set	Sequência	CodedCharacterSetId"2" na página 829	MQLONG
JMS_IBM_PutDate	Sequência	PutDate	MQCHAR8
JMS_IBM_PutTime	Sequência	PutTime	MQCHAR8
JMS_IBM_Last_Msg_In_Group	booleano	MsgFlags	MQLONG

Nota:

1. JMS_IBM_Format representa o formato do corpo da mensagem. Isso pode ser definido pelo aplicativo configurando a propriedade JMS_IBM_Format da mensagem (observe que há um limite de 8 caracteres) ou pode ser padronizado para o formato WebSphere MQ do corpo da mensagem apropriado para o tipo de mensagem JMS. JMS_IBM_Format é mapeado para o campo Formato de MQMD apenas se a mensagem não contém seções RFH ou RFH2. Em uma mensagem típica, ele mapeia para o campo Formato do RFH2 precedendo imediatamente o corpo da mensagem.
2. O valor da propriedade JMS_IBM_Character_Set é um valor de sequência que contém o conjunto de caracteres Java equivalente para o valor numérico CodedCharacterSetId . O campo MQMD CodedCharacterSetId é um valor numérico que contém o equivalente da sequência do conjunto de caracteres Java especificado pela propriedade JMS_IBM_Character_Set.

Mapeando campos JMS para campos do WebSphere MQ (mensagens de saída)
Essas tabelas mostram como os campos de cabeçalho e de propriedade JMS são mapeados para os campos MQMD e MQRFH2 no tempo de envio () ou publicação ().

Tabela 113 na página 830 mostra como os campos de cabeçalho JMS são mapeados para campos MQMD/RFH2 no tempo de envio () ou publicação (). Tabela 114 na página 830 mostra como as propriedades JMS são mapeadas para campos MQMD/RFH2 no tempo de envio () ou publicação (). Tabela 115 na página 830 mostra como as propriedades específicas do provedor JMS são mapeadas para campos MQMD no tempo de envio () ou publicação (),

Para campos marcados como Set by Message Object, o valor transmitido é o valor mantido na mensagem JMS imediatamente antes da operação send () ou publish (). O valor na mensagem JMS é deixado inalterado pela operação.

Para campos marcados como Set by Send Method, um valor é designado quando send () ou publish () é executado (qualquer valor mantido na mensagem JMS é ignorado). O valor na mensagem JMS é atualizado para mostrar o valor utilizado.

Os campos marcados como Somente Receber não são transmitidos e são mantidos sem mudanças na mensagem por envio() ou publicação().

Tabela 113. Mapeamento do campo de mensagem de saída

Nome do campo do cabeçalho JMS	Campo MQMD usado para a transmissão	Cabeçalho	Configurado por
JMSDestination		MQRFH2	Método de envio
JMSDeliveryMode	Persistence	MQRFH2	Método de envio
JMSExpiration	Expiração	MQRFH2	Método de envio
JMSPriority	Priority	MQRFH2	Método de envio
JMSMessageID	MsgID		Método de envio
JMSTimestamp	PutDate/PutTime		Método de envio
JMSCorrelationID	CorrelId	MQRFH2	Objeto de mensagem
JMSReplyTo	ReplyToQ/ReplyToQMgr	MQRFH2	Objeto de mensagem
JMSType		MQRFH2	Objeto de mensagem
JMSRedelivered			Somente Receber

Nota:

1. O campo MQMD CodedCharacterSetId é um valor numérico que contém o equivalente da sequência do conjunto de caracteres Java especificado pela propriedade JMS_IBM_Character_Set.

Tabela 114. Mapeamento da Propriedade JMS de Mensagem de Saída

Nome da propriedade JMS	Campo MQMD usado para a transmissão	Cabeçalho	Configurado por
JMSXUserID	UserIdentifier		Método de envio
JMSXAppID	PutApplName		Método de envio
JMSXDeliveryCount			Somente Receber
JMSXGroupID	GroupId	MQRFH2	Objeto de mensagem
JMSXGroupSeq	MsgSeqNumber	MQRFH2	Objeto de mensagem

Tabela 115. Mapeamento de Propriedade Específicas do Provedor JMS de Mensagem de Saída

Nome da propriedade específica do provedor JMS	Campo MQMD usado para a transmissão	Cabeçalho	Configurado por
JMS_IBM_Report_Exception	Relatório		Objeto de mensagem
JMS_IBM_Report_Expiration	Relatório		Objeto de mensagem
JMS_IBM_Report_COA/COD	Relatório		Objeto de mensagem

Tabela 115. Mapeamento de Propriedade Específicas do Provedor JMS de Mensagem de Saída (continuação)			
Nome da propriedade específica do provedor JMS	**Campo MQMD usado para a transmissão**	**Cabeçalho**	**Configurado por**
JMS_IBM_Report_NAN/PAN	Relatório		Objeto de mensagem
JMS_IBM_Report_Pass_Msg_ID	Relatório		Objeto de mensagem
JMS_IBM_Report_Pass_Correl_ID	Relatório		Objeto de mensagem
JMS_IBM_Report_Discard_Msg	Relatório		Objeto de mensagem
JMS_IBM_MsgType	MsgType		Objeto de mensagem
JMS_IBM_Feedback	Feedback		Objeto de mensagem
JMS_IBM_Format	Formato		Objeto de mensagem
JMS_IBM_PutApplType	PutApplType		Método de envio
JMS_IBM_Encoding	Encoding		Objeto de mensagem
JMS_IBM_Character_Set	CodedCharacterSetId		Objeto de mensagem
JMS_IBM_PutDate	PutDate		Método de envio
JMS_IBM_PutTime	PutTime		Método de envio
JMS_IBM_Last_Msg_In_Group	MsgFlags		Objeto de mensagem

Mapeando campos de cabeçalho JMS em send() ou publish()
Essas notas estão relacionadas ao mapeamento de campos JMS em send () ou publish ().

JMSDestination para MQRFH2
Isso é armazenado como uma sequência que serializa as características importantes do objeto de destino, para que um JMS de recebimento possa reconstituir um objeto de destino equivalente. O campo MQRFH2 é codificado como URI (consulte “Identificadores uniformes de recursos (URIs)” na página 891 para obter detalhes da notação de URI).

JMSReplyTo para MQMD.ReplyToQ, ReplyToQMgr, MQRFH2
O nome da fila é copiado para o campo MQMD.ReplyToQ e o nome do gerenciador de filas é copiado para os campos ReplyToQMgr. As informações de extensão de destino (outros detalhes úteis mantidos no objeto de destino) serão copiadas no campo MQRFH2. O campo MQRFH2 é codificado como um URI (consulte “Identificadores uniformes de recursos (URIs)” na página 891 para obter detalhes da notação URI).

JMSDeliveryMode para MQMD.Persistence
O valor JMSDeliveryMode é configurado pelo método send() ou publish() ou MessageProducer, a menos que o objeto de destino substitua-o. O valor JMSDeliveryMode é mapeado para o campo MQMD.Persistence, conforme a seguir:

- O valor JMS PERSISTENT é equivalente a MQPER_PERSISTENT
- O valor de JMS NON_PERSISTENT é equivalente a MQPER_NOT_PERSISTENT

Se a propriedade de persistência MQQueue não estiver configurada como WMQConstants.WMQ_PER_QDEF, o valor do modo de entrega também será codificado no MQRFH2.

JMSExpiration para/de MQMD.Expiry, MQRFH2
JMSExpiration armazena a hora para expirar (a soma da hora atual e o tempo de vida), enquanto que MQMD armazena o tempo de vida. Além disso, JMSExpiration está em milissegundos, mas MQMD.Expiry está em décimos de um segundo.

- Se o método send() configura um tempo de vida ilimitado, MQMD.Expiry é configurado como MQEI_UNLIMITED e nenhum JMSExpiration é codificado no MQRFH2.
- Se o método send() configurar um tempo de vida menor que 214.748.364,7 segundos (aproximadamente 7 anos), o tempo de vida será armazenado em MQMD.Expiry e o tempo de expiração (em milissegundos) será codificada como um valor i8 no MQRFH2.
- Se o método send() configurar um tempo de vida superior a 214.748.364,7 segundos, MQMD.Expiry será configurado como MQEI_UNLIMITED. O prazo de expiração de true em milissegundos é codificado como um valor i8 no MQRFH2.

JMSPriority para MQMD.Priority
Mapeie diretamente o valor JMSPriority (0-9) para o valor de prioridade MQMD (0-9). Se JMSPriority está configurado para um valor não padrão, o nível de prioridade também é codificado no MQRFH2.

JMSMessageID de MQMD.MessageID
Todas as mensagens enviadas do JMS possuem identificadores de mensagens exclusivos designados pelo WebSphere MQ. O valor designado é retornado no campo MQMD.MessageId após a chamada MQPUT e transmitido de volta para o aplicativo no campo JMSMessageID. O WebSphere MQ messageId é um valor binário de 24 bytes, enquanto o JMSMessageID é uma sequência. O JMSMessageID é composto do valor messageId binário convertido para uma sequência de 48 caracteres hexadecimais, prefixados com o ID de caracteres: O JMS fornece uma sugestão que pode ser configurada para desativar a produção de identificadores de mensagens Essa sugestão é ignorada e um identificador exclusivo designado em todos os casos. Qualquer valor que é configurado para o campo JMSMessageID antes de um send() é sobrescrito.

Se você precisar da capacidade de especificar o MQMD MQMD.MessageID, é possível fazer isso com uma das extensões JMS do WebSphere MQ descritas em "Lendo e Gravando o Descritor de Mensagens a partir de Classes WebSphere MQ para Aplicativo JMS" na página 907.

JMSTimestamp para MQRFH2
Durante um envio, o campo JMSTimestamp é configurado de acordo com o relógio da JVM. Esse valor é configurado no MQRFH2. Qualquer valor que é configurado para o campo JMSTimestamp antes de um send() é sobrescrito. Consulte também as propriedades JMS_IBM_PutDate e JMS_IBM_PutTime.

JMSType para MQRFH2
Esta sequência é configurada no campo MQRFH2 mcd.Type. Se ela estiver em formato URI, também poderá afetar os campos mcd.Set e mcd.Fmt. Consulte também "Usando uma conexão em tempo real com um broker do WebSphere Event Broker ou do WebSphere Message Broker" na página 935.

JMSCorrelationID para MQMD.CorrelId, MQRFH2
O JMSCorrelationID pode conter um dos seguintes:

Um ID de mensagem específico do provedor
Esse é um identificador de mensagem de uma mensagem enviada ou recebida anteriormente e, por isso, deve ser uma sequência de 48 dígitos hexadecimais minúsculos prefixados com *ID:* O prefixo é removido, os caracteres restantes são convertidos em binário e, em seguida, configurados no campo MQMD.CorrelId. Nenhum valor CorrelId é codificado no MQRFH2.

Um valor provider-native byte[]
O valor é copiado no campo MQMD.CorrelId preenchido com nulos ou truncado para 24 bytes, se necessário. Nenhum valor CorrelId é codificado no MQRFH2.

Uma sequência específica do aplicativo
O valor é copiado no MQRFH2. Os primeiros 24 bytes da sequência, no formato UTF8, são gravados no MQMD.CorrelID.

Mapeando campos de propriedade do JMS
Estas notas referem-se ao mapeamento de campos de propriedade JMS em mensagens WebSphere MQ .

JMSXUserID de MQMD UserIdentifier
JMSXUserID é configurado no retorno da chamada de envio.

JMSXAppID de MQMD PutApplName
JSMXAppID está configurado no retorno a partir da chamada de envio.

JMSXGroupID para MQRFH2 (ponto a ponto)
Para mensagens ponto a ponto, o JMSXGroupID é copiado no campo GroupID do MQMD. Se o JMSXGroupID começar com o ID de prefixo, ele será convertido em binário. Caso contrário, ele será codificado como uma sequência UTF8. O valor será preenchido ou truncado, se necessário, para um comprimento de 24 bytes. O sinalizador MQMF_MSG_IN_GROUP está configurado.

JMSXGroupID para MQRFH2 (publicar/assinar)
Para mensagens de publicação/assinatura, o JMSXGroupID é copiado no MQRFH2 como uma sequência.

JMSXGroupSeq MQMD MsgSeqNumber (ponto a ponto)
Para mensagens ponto a ponto, o JMSXGroupSeq é copiado para o campo MsgSeqNumber do MQMD. O sinalizador MQMF_MSG_IN_GROUP está configurado.

JMSXGroupSeq MQMD MsgSeqNumber (publicar/assinar)
Para mensagens de publicação/assinatura, o JMSXGroupSeq é copiado no MQRFH2 como um i4.

Mapeando campos específicos do provedor JMS
As notas a seguir referem-se ao mapeamento de campos específicos do provedor JMS para mensagens IBM WebSphere MQ .

JMS_IBM_Report_<name> para MQMD Report
Um aplicativo JMS pode configurar as opções de Relatório MQMD, usando as seguintes propriedades JMS_IBM_Report_XXX. O único MQMD é mapeado para várias propriedades JMS_IBM_Report_XXX. O aplicativo deve configurar o valor dessas propriedades para as constantes padrão do IBM WebSphere MQ MQRO_ (incluído no com.ibm.mq.MQC). Então, por exemplo, para solicitar COD com dados completos, o aplicativo deve configurar JMS_IBM_Report_COD para o valor CMQC.MQRO_COD_WITH_FULL_DATA.

JMS_IBM_Report_Exception

MQRO_EXCEPTION ou
MQRO_EXCEPTION_WITH_DATA ou
MQRO_EXCEPTION_WITH_FULL_DATA

JMS_IBM_Report_Expiration

MQRO_EXPIRATION ou
MQRO_EXPIRATION_WITH_DATA ou
MQRO_EXPIRATION_WITH_FULL_DATA

JMS_IBM_Report_COA

MQRO_COA ou
MQRO_COA_WITH_DATA ou
MQRO_COA_WITH_FULL_DATA

JMS_IBM_Report_COD

MQRO_COD ou
MQRO_COD_WITH_DATA ou
MQRO_COD_WITH_FULL_DATA

JMS_IBM_Report_PAN
MQRO_PAN

JMS_IBM_Report_NAN
MQRO_NAN

JMS_IBM_Report_Pass_Msg_ID
MQRO_PASS_MSG_ID

JMS_IBM_Report_Pass_Correl_ID
MQRO_PASS_CORREL_ID

JMS_IBM_Report_Discard_Msg
MQRO_DISCARD_MSG

JMS_IBM_MsgType para MQMD MsgType
O valor é mapeado diretamente no MQMD MsgType. Se o aplicativo não tiver configurado um valor explícito do JMS_IBM_MsgType, um valor padrão será usado. Esse valor padrão é determinado conforme a seguir:

- Se JMSReplyTo é configurado para um destino de fila do IBM WebSphere MQ, MSGType é configurado para o valor MQMT_REQUEST
- Se JMSReplyTo não for configurado ou for configurado para algo diferente de um destino de fila do IBM WebSphere MQ, MsgType será configurado para o valor MQMT_DATAGRAM

JMS_IBM_Feedback para MQMD Feedback
O valor é mapeado diretamente no MQMD Feedback.

JMS_IBM_Format para MQMD Format
O valor é mapeado diretamente no Formato MQMD.

JMS_IBM_Encoding para MQMD Encoding
Se configurada, esta propriedade substitui a codificação numérica da Fila de Destino ou Tópico.

JMS_IBM_Character_Set para MQMD CodedCharacterSetId
Se configurada, esta propriedade substitui a propriedade de conjunto de caracteres codificados da Fila de Destino ou Tópico.

JMS_IBM_PutDate de MQMD PutDate
O valor desta propriedade é configurado, durante o envio, diretamente do campo PutDate no MQMD. Qualquer valor que é configurado para a propriedade JMS_IBM_PutDate antes de um envio é sobrescrito. Este campo é uma Sequência de oito caracteres, no formato de Data IBM WebSphere MQ AAAAMMDD. Essa propriedade pode ser usada com a propriedade JMS_IBM_PutTime para determinar o horário em que a mensagem foi colocada, de acordo com o gerenciador de filas.

JMS_IBM_PutTime de MQMD PutTime
O valor dessa propriedade é configurado, durante o envio, diretamente do campo PutTime no MQMD. Qualquer valor que é configurado para a propriedade JMS_IBM_PutTime antes de um envio é sobrescrito. Este campo é uma Sequência de oito caracteres, no formato de Tempo IBM WebSphere MQ HHMMSSTH. Essa propriedade pode ser usada com a propriedade JMS_IBM_PutDate para determinar o horário em que a mensagem foi colocada, de acordo com o gerenciador de filas.

JMS_IBM_Last_Msg_In_Group para MQMD MsgFlags
Para sistema de mensagens ponto a ponto, esse valor booleano é mapeado para a sinalização MQMF_LAST_MSG_IN_GROUP no campo MQMD MsgFlags. Normalmente, ela é usada com as propriedades JMSXGroupID e JMSXGroupSeq para indicar a um aplicativo IBM WebSphere MQ legado que essa mensagem é a última em um grupo. Essa propriedade é ignorada para o sistema de mensagens de publicação/assinatura.

Mapeando os campos do WebSphere MQ em campos JMS (mensagens recebidas)
Essas tabelas mostram como o cabeçalho JMS e os campos de propriedade são mapeados para os campos MQMD e MQRFH2 no tempo de get () ou receive ().

Tabela 116 na página 835 mostra como os campos de cabeçalho JMS são mapeados para os campos MQMD/MQRFH2 no horário get () ou receive (). Tabela 117 na página 835 mostra como os campos de propriedade JMS são mapeados para os campos MQMD/MQRFH2 no tempo de get () ou receive. Tabela 118 na página 836 mostra como as propriedades específicas do provedor JMS são mapeadas.

Tabela 116. Mapeamento do campo do cabeçalho JMS da mensagem recebida

Nome do campo do cabeçalho JMS	Campo MQMD recuperado	Campo MQRFH2 recuperado
JMSDestination		jms.Dst ou mqps.Top "1" na página 835
JMSDeliveryMode	Persistência "2" na página 835	jms.Dlv "2" na página 835
JMSExpiration		jms.Exp
JMSPriority	Priority	
JMSMessageID	MsgID	
JMSTimestamp	PutDate "2" na página 835 PutTime "2" na página 835	jms.Tms "2" na página 835
JMSCorrelationID	CorrelId "2" na página 835	jms.Cid "2" na página 835
JMSReplyTo	ReplyToQ "2" na página 835 ReplyToQMgr "2" na página 835	jms.Rto "2" na página 835
JMSType		mcd.Type, mcd.Set, mcd.Fmt
JMSRedelivered	BackoutCount	

Nota:

1. Se jms.Dst e mqps.Top estão definidos, o valor em jms.Dst é usado.
2. Para propriedades que podem ter valores recuperados de MQRFH2 ou MQMD, se ambos estiverem disponíveis, a configuração no MQRFH2 é usada.
3. O valor da propriedade JMS_IBM_Character_Set é um valor de sequência que contém o conjunto de caracteres Java equivalente para o valor numérico CodedCharacterSetId .

Tabela 117. Mapeamento de propriedade da mensagem de entrada

Nome da propriedade JMS	Campo MQMD recuperado	Campo MQRFH2 recuperado
JMSXUserID	UserIdentifier	
JMSXAppID	PutApplName	
JMSXDeliveryCount	BackoutCount	
JMSXGroupID	GroupId "1" na página 835	jms.Gid "1" na página 835
JMSXGroupSeq	MsgSeqNumber "1" na página 835	jms.Seq "1" na página 835

Nota:

1. Para propriedades que podem ter valores recuperados de MQRFH2 ou MQMD, se ambos estiverem disponíveis, a configuração no MQRFH2 é usada. As propriedades são configuradas a partir dos valores MQMD somente se os sinalizadores de mensagens MQMF_MSG_IN_GROUP ou MQMF_LAST_MSG_IN_GROUP estiverem configurados.

Tabela 118. Mapeamento de Propriedade JMS Específicas do Provedor de Mensagens de Entrada		
Nome da propriedade JMS	**Campo MQMD recuperado**	**Campo MQRFH2 recuperado**
JMS_IBM_Report_Exception	Relatório	
JMS_IBM_Report_Expiration	Relatório	
JMS_IBM_Report_COA	Relatório	
JMS_IBM_Report_COD	Relatório	
JMS_IBM_Report_PAN	Relatório	
JMS_IBM_Report_NAN	Relatório	
JMS_IBM_Report_Pass_Msg_ID	Relatório	
JMS_IBM_Report_Pass_Correl_ID	Relatório	
JMS_IBM_Report_Discard_Msg	Relatório	
JMS_IBM_MsgType	MsgType	
JMS_IBM_Feedback	Feedback	
JMS_IBM_Format	Formato	
JMS_IBM_PutApplType	PutApplType	
JMS_IBM_Encoding "1" na página 836	Encoding	
JMS_IBM_Character_Set "1" na página 836	CodedCharacterSetId	
JMS_IBM_PutDate	PutDate	
JMS_IBM_PutTime	PutTime	
JMS_IBM_Last_Msg_In_Group	MsgFlags	
1. Somente configure se a mensagem de entrada for uma Mensagem de bytes.		

Trocando mensagens entre um aplicativo JMS e um aplicativo WebSphere MQ tradicional
Este tópico descreve o que acontece quando um aplicativo JMS troca mensagens com um aplicativo WebSphere MQ tradicional que não pode processar o cabeçalho MQRFH2 .

. Figura 129 na página 837 mostra o mapeamento..

O administrador indica que o aplicativo JMS está se comunicando com um aplicativo WebSphere MQ tradicional configurando a propriedade TARGCLIENT do destino para *MQ*. Isso indica que nenhum cabeçalho MQRFH2 deve ser produzido. Se isso não for feito, o aplicativo de recebimento deverá ser capaz de manipular o cabeçalho MQRFH2.

O mapeamento de JMS para MQMD destinado a um aplicativo WebSphere MQ tradicional é o mesmo que o mapeamento de JMS para MQMD destinado a um aplicativo JMS. Se as classes do WebSphere MQ para JMS receberem uma mensagem do WebSphere MQ com o campo MQMD *Format* configurado para algo diferente de MQFMT_RFH2, os dados serão recebidos de um aplicativo não JMS. Se o formato for MQFMT_STRING, a mensagem será recebida como uma mensagem de texto JMS.. Caso contrário, ele será recebido como uma mensagem de bytes do JMS Como não há MQRFH2, apenas as propriedades JMS que são transmitidas no MQMD podem ser restauradas.

Se as classes do WebSphere MQ para JMS receberem uma mensagem que não tenha um cabeçalho MQRFH2 , a propriedade TARGCLIENT do objeto Queue ou Topic derivado do campo do cabeçalho JMSReplyTo da mensagem será configurado como MQ por padrão.. Isso significa que uma mensagem de resposta enviada para a fila ou tópico também não tem um cabeçalho MQRFH2. Será possível desativar

esse comportamento de inclusão de um cabeçalho MQRFH2 em uma mensagem de resposta somente se a mensagem original tiver um cabeçalho MQRFH2, configurando a propriedade TARGCLIENTMATCHING do connection factory para NO.

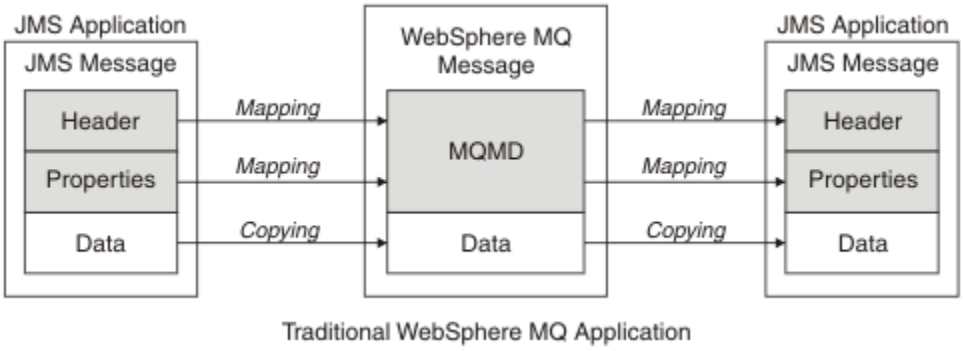


Figura 129. Como as mensagens JMS são transformadas em mensagens do WebSphere MQ sem cabeçalho MQRFH2

O corpo da mensagem JMS
Este tópico contém informações sobre a codificação do corpo da mensagem em si. A codificação depende do tipo de mensagem JMS..

ObjectMessage
Um ObjectMessage é um objeto serializado pelo Java Runtime da maneira normal

TextMessage
TextMessage é uma sequência codificada. Para uma mensagem de saída, a sequência está codificada no conjunto de caracteres fornecido pelo objeto de destino. O padrão usado é a codificação UTF8 (a codificação UTF8 começa com o primeiro caractere da mensagem; não há campo de comprimento no início). É possível, no entanto, especificar qualquer outro conjunto de caracteres suportado pelas classes do WebSphere MQ para JMS Esses conjuntos de caracteres são usados principalmente quando você envia uma mensagem para um aplicativo não JMS

Se o conjunto de caracteres for um conjunto de byte duplo (incluindo UTF16), a especificação de codificação de número inteiro do objeto de destino determina a ordem dos bytes.

Uma mensagem recebida é interpretada usando o conjunto de caracteres e a codificação especificados na própria mensagem. Essas especificações estão no último cabeçalho WebSphere MQ (ou MQMD se não houver cabeçalhos). Para mensagens JMS, o último cabeçalho é geralmente o MQRFH2.

BytesMessage
Um BytesMessage é, por padrão, uma seqüência de bytes conforme definido pela especificação JMS 1.0.2 e documentação Java associada.

Para uma mensagem não enviada que foi montada pelo próprio aplicativo, a propriedade de codificação do objeto de destino pode ser usada para substituir as codificações de número inteiro e os campos de vírgula flutuante na mensagem. Por exemplo, é possível solicitar que os valores de ponto flutuante sejam armazenados no S/390 em vez do formato IEEE).

Uma mensagem recebida é interpretada usando a codificação numérica especificada na própria mensagem. Esta especificação está no último cabeçalho WebSphere MQ (ou MQMD se não houver cabeçalhos). Para mensagens JMS, o último cabeçalho é geralmente o MQRFH2.

Se um BytesMessage for recebido e reenviado sem modificação, seu corpo será transmitido byte por byte, como foi recebido. A propriedade de codificação do objeto de destino não tem efeito no corpo. A única entidade semelhante a uma sequência que pode ser enviada explicitamente em um BytesMessage é uma sequência UTF8. Isso é codificado no formato Java UTF8 e começa com um campo de comprimento de 2 bytes. A propriedade do conjunto de caracteres do objeto de destino não tem efeito na codificação de um BytesMessage de saída. O valor do conjunto de caracteres em uma

mensagem recebida do WebSphere MQ não tem efeito na interpretação dessa mensagem como um JMS BytesMessage.

É improvável que aplicativos não Java reconheçam a codificação Java UTF8 . Portanto, para um aplicativo JMS para enviar um BytesMessage que contém dados de texto, o próprio aplicativo deve converter suas sequências em matrizes de bytes e gravar essas matrizes de bytes em BytesMessage

MapMessage
Um MapMessage é uma sequência que contém os trios nome/tipo/valor XML codificados como:

```
<map>
  <elt name="elementname1" dt="datatype1">value1</elt>
  <elt name="elementname2" dt="datatype2">value2</elt>
  ...
</map>
```

em que `datatype` é um dos tipos de dados listados em Tabela 109 na página 827. O tipo de dados padrão é `string`, portanto, o atributo `dt="string"` é omitido para elementos de sequência.

O conjunto de caracteres usado para codificar ou interpretar a sequência XML que forma o corpo de uma mensagem de mapa é determinado de acordo com as regras que se aplicam a uma mensagem de texto.

Versões de classes do WebSphere MQ para JMS anteriores à versão 5.3 codificaram o corpo de uma mensagem de mapa no seguinte formato:

```
<map>
  <elementname1 dt="datatype1">value1</elementname1>
  <elementname2 dt="datatype2">value2</elementname2>
  ...
</map>
```

A versão 5.3 e as versões mais recentes das classes do WebSphere MQ para JMS podem interpretar qualquer formato, mas as versões das classes do WebSphere MQ para JMS anteriores à versão 5.3 não podem interpretar o formato atual.

Se um aplicativo precisar enviar mensagens de mapa para outro aplicativo que está usando uma versão das classes do WebSphere MQ para JMS anterior a Versão 5.3, o aplicativo de envio deverá chamar o método do connection factory `setMapNameStyle(WMQConstants.WMQ_MAP_NAME_STYLE_COMPATIBLE)` para especificar que as mensagens de mapa sejam enviadas no formato anterior Por padrão, todas as mensagens do mapa são enviadas no formato atual.

StreamMessage
Um StreamMessage é semelhante a uma mensagem do mapa, mas sem nomes de elementos:

```
<stream>
  <elt dt="datatype1">value1</elt>
  <elt dt="datatype2">value2</elt>
  ...
</stream>
```

em que `datatype` é um dos tipos de dados listados em Tabela 109 na página 827. O tipo de dados padrão é `string`, portanto, o atributo `dt="string"` é omitido para elementos de sequência.

O conjunto de caractere usado para codificar ou interpretar a sequência XML que compõe o corpo do StreamMessage é determinado segundo as regras que se aplicam a um TextMessage.

O campo MQRFH2.format é configurado da seguinte forma:

MQFMT_NONE
para ObjectMessage, BytesMessage ou mensagens sem corpo.

MQFMT_STRING
para TextMessage, StreamMessage ou MapMessage.

Conversão de Mensagens JMS

A conversão de dados da mensagem no JMS é executada ao enviar e receber mensagens WebSphere MQ executa a maioria da conversão de dados automaticamente. Ele converte dados de texto e numéricos ao transferir uma mensagem entre aplicativos JMS O texto é convertido ao trocar um `JMSTextMessage` entre um aplicativo JMS e um aplicativo WebSphere MQ .

Se você estiver planejando fazer trocas de mensagens mais complexas, os tópicos a seguir serão de seu interesse. As trocas de mensagens complexas incluem:

- Transferindo mensagens não de texto entre um aplicativo WebSphere MQ e um aplicativo JMS.
- Trocando dados de texto no formato de byte.
- Convertendo o texto em seu aplicativo.

Dados da mensagem JMS

A conversão de dados é necessária para trocar dados de texto e numéricos entre aplicativos, mesmo entre dois aplicativos JMS. A representação interna de texto e números deve ser codificada para que possam ser transferidas em uma mensagem. A codificação força uma decisão sobre como números e texto são representados. WebSphere MQ gerencia a codificação de texto e números em mensagens JMS, exceto `JMSObjectMessage`, consulte “JMSObjectMessage” na página 846. Ele usa três atributos de mensagem. Os três atributos são `CodedCharacterSetId`, `Encoding` e `Format`.

Esses três atributos de mensagem são normalmente armazenados nos campos do cabeçalho JMS, `MQRFH2`, de uma mensagem JMS Se o tipo de mensagem for um tipo de mensagem `MQ`, em vez de `JMS` , os atributos serão armazenados no descritor de mensagem, `MQMD` Os atributos são usados para converter os dados da mensagem JMS.. Os dados da mensagem JMS são transferidos na parte de dados da mensagem de uma mensagem do WebSphere MQ

Propriedades de mensagem JMS

As propriedades de mensagens JMS, como `JMS_IBM_CHARACTER_SET`, são trocadas na parte do cabeçalho `MQRFH2` de uma mensagem JMS, a menos que a mensagem tenha sido enviada sem um `MQRFH2` Apenas `JMSTextMessage` e `JMSBytesMessage` podem ser enviados sem um `MQRFH2`. Se uma propriedade JMS for armazenada como uma propriedade de mensagens do WebSphere MQ no descritor de mensagens, `MQMD`, ela será convertida como parte da conversão `MQMD` Se uma propriedade JMS for armazenada no `MQRFH2`, ela será armazenada no conjunto de caracteres especificado por `MQRFH2.NameValueCCSID` Quando uma mensagem é enviada ou recebida, as propriedades da mensagem são convertidas para e a partir de sua representação interna na JVM. A conversão é para e a partir do conjunto de caracteres do descritor de mensagens ou `MQRFH2.NameValueCCSID`. Os dados numéricos são convertidos para texto.

Conversão de Mensagens JMS

Os tópicos a seguir contêm exemplos e tarefas que são úteis se você planeja trocar mensagens mais complexas que requerem conversão.

Abordagens de conversão de mensagens JMS

Várias abordagens de conversão de dados estão abertas para os designers de aplicativos JMS Essas abordagens não são exclusivas; alguns aplicativos devem usar uma combinação dessas abordagens. Se seu aplicativo estiver trocando apenas texto ou estiver trocando mensagens apenas com outros aplicativos JMS, normalmente você não considera a conversão de dados A conversão de dados é executada automaticamente para você pelo WebSphere MQ.

É possível fazer uma série de perguntas sobre como abordar a conversão de mensagens:

É necessário pensar sobre a conversão de mensagens mesmo?

Em alguns casos, como transferências de mensagens JMS para JMS e troca de mensagens de texto com IBM WebSphere MQ programas, IBM WebSphere MQ executa as conversões necessárias para você, automaticamente. Você pode desejar controlar a conversão de dados por motivos de

desempenho ou trocar mensagens complexas que possuem um formato predefinido. Em casos como esses, deve-se entender a conversão de mensagens e ler os seguintes tópicos.

Quais tipos de conversão existem?

Há quatro tipos principais de conversão, que são explicadas nas seções a seguir:

1. “Conversão de dados do cliente JMS” na página 840
2. “Conversão de Dados do Aplicativo” na página 841
3. “Conversão de dados do gerenciador de filas” na página 841
4. “Conversão de dados do canal de mensagens” na página 842

Onde a conversão deve ser executada?

A seção, “Escolha uma abordagem para a conversão de mensagem: receiver makes good” na página 842, descreve a abordagem comum de "receiver makes good". "Receptor faz bom" também se aplica à conversão de dados JMS

Conversão de dados do cliente JMS

Cliente JMS[4]conversão de dados é a conversão de primitivas e objetos Java em bytes em uma mensagem JMS como ela é enviada para um destino e conversão de volta, quando ela é recebida. A conversão de dados do cliente JMS usa os métodos das classes `JMSMessage` Os métodos estão listados pelo tipo de classe `JMSMessage` em Tabela 119 na página 843.

A conversão de e para a representação interna da JVM de números e de texto é executada para os métodos read, get, set e write. A conversão será executada quando a mensagem for enviada e quando qualquer um dos métodos read ou get for chamado em uma mensagem que tenha sido recebida.

A página de códigos e a codificação numérica usadas para gravar ou configurar o conteúdo de uma mensagem são definidas como atributos do destino. A página de códigos de destino e a codificação numérica podem ser mudadas de forma administrativa. Um aplicativo também pode substituir a página de códigos de destino e a codificação configurando as propriedades de mensagens que controlam o conteúdo da mensagem de configuração ou gravação.

Se você deseja converter a codificação do número quando uma mensagem `JMSBytesMessage` é enviada a um destino que não é definido como codificação `Native`, deve-se configurar a propriedade de mensagem `JMS_IBM_ENCODING` antes de enviar a mensagem. Se estiver seguindo o padrão "receiver makes good" ou se estiver trocando mensagens entre aplicativos JMS, o aplicativo não precisará configurar `JMS_IBM_ENCODING`. Na maioria dos casos, é possível deixar a propriedade `Encoding` como `Native`.

Para mensagens `JMSStreamMessage`, `JMSMapMessage` e `JMSTextMessage`, as propriedades com o identificador do conjunto de caracteres do destino são usadas. A codificação é ignorada em enviar como os números são gravados no formato de texto. O programa de aplicativo cliente JMS não precisa configurar `JMS_IBM_CHARACTER_SET` antes de enviar a mensagem se a propriedade do conjunto de caracteres de destino for aplicada.

Para obter os dados em uma mensagem, um aplicativo chama os métodos read ou get da mensagem JMS. Os métodos referem-se à página de códigos e codificação definida no cabeçalho da mensagem anterior para criar as primitivas e objetos Java corretamente.

A conversão de dados do cliente JMS atende às necessidades da maioria de aplicativos JMS que estão trocando mensagens entre um cliente JMS e outro. Não codifique qualquer conversão de dados explícitos. Não use a classe `java.nio.charset.Charset`, que geralmente é usada ao gravar texto em um arquivo. Os métodos `writeString` e `setString` fazem a conversão para você.

Para obter mais detalhes sobre a conversão de dados do cliente JMS, consulte “Conversão de mensagens do cliente JMS e codificação” na página 853

[4] "JMS Client" refere-se às classes WebSphere MQ para JMS que implementam a interface JMS, que é executada no modo de cliente ou de ligação

Conversão de Dados do Aplicativo

Um aplicativo cliente JMS pode executar a conversão explícita de dados de caracteres usando a classe `java.nio.charset.Charset` ; consulte os exemplos em Figura 132 na página 845 e Figura 133 na página 845 . Os dados da sequência são convertidos em bytes, usando o método `getBytes` e enviados como bytes. Os bytes são convertidos novamente em texto usando um construtor `String` que usa uma matriz de bytes e uma `Charset`. Os dados de caracteres são convertidos usando os métodos `encode` e `decode` `Charset`. Geralmente, a mensagem é enviada ou recebida como `JMSBytesMessage`, porque a parte da mensagem de um `JMSBytesMessage` não contém nada além dos dados gravados pelo aplicativo[5]. Também é possível enviar e receber bytes usando `JMSStreamMessage`, `JMSMapMessage` ou `JMSObjectMessage`.

Não há métodos Java para codificar e decodificar bytes que contenham dados numéricos representados em diferentes formatos de codificação Os dados numéricos são codificados e decodificados automaticamente usando os métodos numéricos read e write `JMSMessage`. Os métodos read e write usam o valor do atributo `JMS_IBM_ENCODING` dos dados da mensagem.

Um uso típico para conversão de dados do aplicativo é se um cliente JMS enviar ou receber uma mensagem formatada de um aplicativo não JMS. Uma mensagem formatada contém dados numéricos, de texto e de bytes organizados pelo comprimento dos campos de dados. A menos que o aplicativo não JMS tenha especificado o formato da mensagem como "MQSTR ", a mensagem será construída como `JMSBytesMessage`. Para receber dados da mensagem formatada em uma `JMSBytesMessage`, deve-se chamar uma sequência de métodos. Os métodos devem ser chamados na mesma ordem que os campos foram gravados na mensagem. Se os campos forem numéricos, deverá saber a codificação e o comprimento dos dados numéricos. Se algum dos campos contiver dados de byte ou de texto, deverá conhecer o comprimento de quaisquer dados de byte na mensagem. Há duas maneiras de converter uma mensagem formatado em um objeto Java fácil de usar.

1. Construa uma classe Java correspondente ao registro para encapsular a leitura e gravação da mensagem. O acesso aos dados no registro é com os métodos get e set da classe.
2. Construa uma classe Java correspondente ao registro, estendendo a classe `com.ibm.mq.headers` O acesso aos dados na classe é com os acessadores específicos de tipo no formato `getStringValue(`*`fieldname`*`);`

Consulte o “Trocando um registro formatado com um aplicativo não JMS” na página 860.

Conversão de dados do gerenciador de filas

No WebSphere MQ V7.0, a conversão da página de códigos pode ser executada por um gerenciador de filas quando um programa cliente JMS obtém uma mensagem A conversão é a mesma que a executada para um programa C. Um programa C configura `MQGMO_CONVERT` como uma opção de parâmetro `MQGET GetMsgOpts`; consulte Figura 131 na página 845. Um gerenciador de filas executa conversão para um programa cliente JMS que está recebendo uma mensagem, se a propriedade de destino `WMQ_RECEIVE_CONVERSION` estiver configurada como `WMQ_RECEIVE_CONVERSION_QMGR`, o programa cliente JMS também poderá configurar a propriedade de destino; consulte Figura 130 na página 842.

Antes da V7.0, as conversões eram sempre executadas por um cliente JMS A conversão de dados do cliente JMS é restrita para converter sequências de números e texto do tipo e comprimento conhecidos para o cliente JMS. Não é possível converter as estruturas de dados; consulte “Trocando um registro formatado com um aplicativo não JMS” na página 860. Em V7.0, até que o fix pack 7.0.1.5, se a conversão puder ser executada pelo gerenciador de filas, ela será sempre executada por ele. De 7.0.1.5 em diante, o comportamento de conversão padrão é revertido para o mesmo que V6.0e todas as conversões são executadas pelo cliente JMS. A partir de 7.0.1.5 ou 7.0.1.4 com APAR IC72897, é possível configurar uma nova opção de destino, `WMQ_RECEIVE_CONVERSION`, para controlar onde a conversão será executada e `WMQ_RECEIVE_CCSID`, para configurar a página de códigos de destino; consulte Figura 130 na página 842.

[5] Uma exceção: dados gravados usando `writeUTF` começam com um campo de 2 bytes de comprimento

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Ou

```
((MQDestination)destination).setReceiveConversion
             (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Figura 130. Ativar conversão de dados do gerenciador de filas

O principal benefício da conversão do gerenciador de filas é fornecido ao trocar mensagens com aplicativos não JMS Se o campo `Format` na mensagem for definido e o conjunto de caracteres de destino, ou codificação, for diferente para a mensagem, o gerenciador de filas executará a conversão de dados para o aplicativo de destino, se o aplicativo solicitá-la. O gerenciador de filas converte os dados da mensagem formatados de acordo com um dos tipos de mensagens predefinidos do WebSphere MQ , como um cabeçalho da ponte CICS (`MQCIH`). Se o campo `Format` for definido pelo usuário, o gerenciador de filas procurará uma saída de conversão de dados com o nome fornecido no campo `Format`

A conversão de dados do gerenciador de filas é usada para melhorar o efeito com o design padrão "receiver makes good". Um cliente JMS de envio não precisa executar a conversão. Um programa de recebimento não JMS depende da saída de conversão para assegurar que a mensagem seja entregue na página de códigos e codificação necessárias. Com um cliente JMS de envio e um receptor não JMS, o exemplo se aplica a IBM WebSphere MQ pré e post-V7.0. Com o IBM WebSphere MQ V7.0, a saída de conversão também pode ser chamada para um programa JMS de recebimento.

É possível criar uma saída de conversão de dados, usando o utilitário de saída de conversão de dados, **`crtmqcvx`**, para permitir que o gerenciador de filas converta seus próprios dados formatados de registro. É possível construir seu próprio formato de registro, usar o `com.ibm.mq.headers` para acessá-lo como uma classe Java e usar sua própria saída de conversão para convertê-lo. No z/OS o utilitário é chamado **`CSQUCVX`**e no IBM i, **`CVTMQMDTA`**. Consulte o “Trocando um registro formatado com um aplicativo não JMS” na página 860.

Conversão de dados do canal de mensagens

WebSphere MQ Os canais Emissor, Servidor, Receptor de Cluster e Emissor de Cluster possuem uma opção de conversão de mensagem, `CONVERT`. O conteúdo de uma mensagem poderá, opcionalmente, ser convertido quando uma mensagem for enviada. A conversão ocorre na extremidade de envio do canal. A definição do cluster-receiver é usada para definir automaticamente o canal cluster-sender correspondente.

A conversão de dados por canais de mensagem será geralmente usada se não for possível usar outras formas de conversão.

Escolha uma abordagem para a conversão de mensagem: "receiver makes good"

A abordagem usual no design do aplicativo WebSphere MQ para conversão de código é "receiver makes good". "Receiver makes good" reduz o número de conversões de mensagens. Ele também evitará o problema de erros do canal inesperados se a conversão de mensagem falhar em algum gerenciador de filas intermediário durante a transferência de mensagens. A regra "receiver makes good" só será interrompida se houver algum motivo pelo qual não é possível efetuar o receiver makes good. A plataforma de recebimento pode não ter o conjunto de caracteres à direita, por exemplo.

"O receptor se torna bom" também é uma boa orientação geral para aplicativos clientes JMS Mas em casos específicos, a conversão para o conjunto de caracteres correto na origem pode ser mais eficiente. A conversão a partir da representação interna de JVM deverá ocorrer quando uma mensagem que contém tipos numéricos ou texto for enviada. A conversão para o conjunto de caracteres requerido pelo receptor, se o receptor não for um cliente JMS, poderá remover a necessidade de o destinatário não JMS executar

a conversão Se o destinatário for um cliente JMS, ele será convertido novamente, de qualquer maneira, para decodificar os dados da mensagem e criar primitivas e objetos Java de criação.

A diferença entre aplicativos clientes JMS e aplicativos gravados em uma linguagem como C, é que Java deve executar a conversão de dados. Um aplicativo Java deve converter números e texto de sua representação interna para um formato codificado usado em mensagens..

Configurando propriedades de destino ou de mensagem, é possível configurar o conjunto de caracteres e a codificação usados pelo WebSphere MQ para codificar números e texto em mensagens. Normalmente, você deixaria o conjunto de caracteres como `1208` e a codificação como `Native`.

WebSphere MQ não converte matrizes de bytes. Para codificar sequências e matrizes de caracteres em matrizes de bytes use o pacote `java.nio.charset`. `Charset` especifica o conjunto de caracteres usado para converter uma sequência ou matriz de caracteres em uma matriz de bytes. Também é possível decodificar uma matriz de bytes em uma matriz de sequência ou caracteres usando um `Charset`. Não é uma boa prática depender de `java.nio.charset.Charset.defaultCodePage`, ao codificar sequências e matrizes de caracteres. O padrão `Charset` geralmente é `windows-1252` no Windows, e UTF-8 no UNIX `windows-1252` é um conjunto de caracteres de byte único e UTF-8 é um conjunto de caracteres multibyte.

Geralmente, deixe o conjunto de caracteres de destino e as propriedades de codificação em seus valores padrão de UTF-8 e `Native` ao trocar mensagens com outros aplicativos JMS Se você estiver trocando mensagens contendo números ou texto com um aplicativo JMS, escolha um dos tipos de mensagens `JMSTextMessage`, `JMSStreamMessage`, `JMSMapMessage`ou `JMSObjectMessage` que se ajuste ao seu propósito. Não há outras tarefas conversão a serem feitas.

Se você estiver trocando mensagens com aplicativos não JMS que usam um formato de registro, será mais complicado. A menos que todo o registro contenha texto e possa ser transferido como uma `JMSTextMessage`, deve-se codificar e decodificar o texto no aplicativo. Configure o tipo de mensagem de destino como `MQ`e use `JMSBytesMessage` para evitar que as classes IBM WebSphere MQ para JMS incluam informações adicionais de cabeçalho e de identificação para os dados da mensagem Use os métodos `JMSBytesMessage` para gravar números e bytes, e a classe `Charset` converte texto em matrizes de bytes explicitamente. Um número de fatores podem influenciar na sua escolha do conjunto de caracteres:

- Desempenho: é possível reduzir o número de conversões transformando texto em um conjunto de caracteres usado no maior número de servidores?
- Uniformidade: transfira todas as mensagens no mesmo conjunto de caracteres.
- Abundância: quais conjuntos de caracteres têm todos os pontos de código que os aplicativos devem usar?
- Simplicidade: os conjuntos de caracteres de byte único são mais simples de usar do que o comprimento variável e os conjuntos de caracteres multibyte.

Consulte o “Trocando um registro formatado com um aplicativo não JMS” na página 860. para exemplos de conversão de mensagens trocadas com aplicativos não JMS.

Examples

Tabela de tipos de mensagens e tipos de conversões

Tabela 119. Tipos de mensagens e tipos de conversão

	Tipo de conversão			
Tipo de Mensagem	**text**	**Numérica**	**Outro**	**Nenhum**
`JMSObjectMessage`				`getObject` `setObject`

Tabela 119. Tipos de mensagens e tipos de conversão (continuação)

	Tipo de conversão			
Tipo de Mensagem	**text**	**Numérica**	**Outro**	**Nenhum**
JMSTextMessage	getText setText			
JMSBytesMessage	readUTF writeUTF	readDouble readFloat readInt readLong readShort readUnsignedShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readUnsignedByte readBytes readChar writeByte writeBytes writeChar
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

```
gmo.Options = MQGMO_WAIT         /* wait for new messages           */
            | MQGMO_NO_SYNCPOINT /* no transaction                  */
            | MQGMO_CONVERT;     /* convert if necessary            */

     while (CompCode != MQCC_FAILED) {
     buflen = sizeof(buffer) - 1; /* buffer size available for GET   */
     memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
     memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
     md.Encoding       = MQENC_NATIVE;
     md.CodedCharSetId = MQCCSI_Q_MGR;

           MQGET(Hcon,          /* connection handle               */
           Hobj,                /* object handle                   */
           &md,                 /* message descriptor              */
           &gmo,                /* get message options             */
           buflen,              /* buffer length                   */
           buffer,              /* message buffer                  */
           &messlen,            /* message length                  */
           &CompCode,           /* completion code                 */
           &Reason);            /* reason code                     */
```

Figura 131. Fragmento de código de amqsget0.c

Enviando e recebendo texto em uma JMSBytesMessage

O código em Figura 132 na página 845 envia uma sequência em uma BytesMessage. Para simplicidade, o exemplo envia uma sequência única, para a qual uma JMSTextMessage é mais apropriada. Para receber uma sequência de texto em mensagem de bytes contendo uma mistura de tipos, deve-se saber o comprimento da sequência em bytes, chamado *TEXT_LENGTH* em Figura 133 na página 845. Mesmo para uma sequência com um número fixo de caracteres, o comprimento da representação de bytes pode ser maior.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                  .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Figura 132. Enviando uma String em um JMSBytesMessage

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Figura 133. Recebendo um String de um JMSBytesMessage

Conceitos relacionados

Conversão de mensagens do cliente JMS e codificação
Os métodos usados para executar a conversão e a codificação da mensagem do cliente JMS são listados, com exemplos de código de cada tipo de conversão.

Conversão de dados do gerenciador de filas
A conversão de dados do gerenciador de filas sempre esteve disponível para aplicativos não JMS que recebem mensagens de clientes JMS Desde a V7.0, os clientes JMS que recebem mensagens também usam a conversão de dados do gerenciador de filas A partir da 7.0.1.5 ou 7.0.1.4 com a APAR IC72897, a conversão de dados do gerenciador de filas é opcional.

Tarefas relacionadas

Trocando um registro formatado com um aplicativo não JMS
Siga as etapas sugeridas nesta tarefa para projetar e construir uma saída de conversão de dados e um aplicativo cliente JMS que possa trocar mensagens com um aplicativo não JMS usando o JMSBytesMessage A troca de uma mensagem formatada com um aplicativo não JMS pode ocorrer com ou sem chamar uma saída de conversão de dados.

Referências relacionadas

Tipos de mensagens JMS e conversão
A opção de tipo de mensagem afeta sua abordagem de conversão de mensagens. A interação da conversão de mensagem com o tipo de mensagem é descrita para os tipos de mensagens JMS, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessagee JMSBytesMessage

Tipos de mensagens JMS e conversão
A opção de tipo de mensagem afeta sua abordagem de conversão de mensagens. A interação da conversão de mensagem com o tipo de mensagem é descrita para os tipos de mensagens JMS, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessagee JMSBytesMessage

JMSObjectMessage

JMSObjectMessage contém um objeto e quaisquer objetos que ele faz referência, serializado em um fluxo de bytes pela JVM. O texto é serializado em UTF-8 e limitado para sequências ou matrizes de caracteres que não ultrapasse 65.534 bytes. Uma vantagem de JMSObjectMessage é que os aplicativos não estejam envolvidos em quaisquer problemas de conversão de dados, contanto que eles usem apenas os métodos e os atributos do objeto. JMSObjectMessage fornece a conversão de dados para objetos complexos sem o programador de aplicativos considerando como codificar um objeto em uma mensagem. A desvantagem de usar o JMSObjectMessage é que ele pode ser trocado apenas com outros aplicativos JMS Ao escolher um dos outros tipos de mensagens JMS, é possível trocar mensagens JMS com aplicativos não JMS

"Enviando e recebendo uma JMSObjectMessage" na página 849 mostra um objeto String que está sendo trocado em uma mensagem.

Um aplicativo cliente JMS pode receber um JMSObjectMessage apenas em uma mensagem que tenha um corpo de estilo JMS O destino deve especificar um corpo de estilo JMS..

JMSTextMessage

JMSTextMessage contém uma sequência de texto única. Quando uma mensagem de texto é enviada, o texto Format é configurado para "MQSTR ", WMQConstants.MQFMT_STRING O CodedCharacterSetId do texto é configurado para o identificador do conjunto de caracteres codificados definido para seu destino. O texto é codificado no CodedCharacterSetId pelo WebSphere MQ.. Os campos CodedCharacterSetId e Format são configurados no descritor de mensagem, MQMD, ou nos campos JMS em MQRFH2. Se a mensagem for definida como tendo um estilo do corpo da mensagem WMQ_MESSAGE_BODY_MQ ou o estilo de corpo não estiver especificado, mas o destino for WMQ_TARGET_DEST_MQ então os campos do descritor de mensagens serão configurados. Caso contrário, a mensagem terá um JMS RFH2 e os campos serão configurados na parte fixa de MQRFH2

Um aplicativo pode substituir o identificador do conjunto de caracteres codificados definidos para um destino. Ele deve configurar a propriedade de mensagem JMS_IBM_CHARACTER_SET para um identificador do conjunto de caracteres codificados; consulte o exemplo em "Enviando e recebendo um JMSTextmessage" na página 849.

Quando o cliente JMS chama a conversão do gerenciador de filas do método consumer.receive , é opcional A conversão do gerenciador de filas é ativado configurando a propriedade de destino WMQ_RECEIVE_CONVERSION para WMQ_RECEIVE_CONVERSION_QMGR. O gerenciador de filas converte a mensagem de texto do JMS_IBM_CHARACTER_SET especificado para a mensagem antes de transferir a mensagem para o cliente JMS. O conjunto de caracteres da mensagem convertida é 1208, UTF-8, a menos que o destino tenha um WMQ_RECEIVE_CCSID diferente. O CodedCharacterSetId na mensagem que se refere à JMSTextMessage é atualizado para o ID do conjunto de caracteres de

destino. O texto é decodificado a partir do conjunto de caracteres de destino em Unicode pelo método getText; consulte o exemplo em "Enviando e recebendo um JMSTextmessage" na página 849.

Um JMSTextMessage pode ser enviado em um corpo da mensagem no estilo MQ, sem um cabeçalho JMS MQRFH2 O valor dos atributos de destino, WMQ_MESSAGE_BODY e WMQ_TARGET_DEST determinam o estilo de corpo da mensagem, a menos que substituído pelo aplicativo. O aplicativo pode substituir os valores configurados no destino, chamando destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ) ou destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ).

Se você enviar uma JMSTextMessage com um corpo de estilo MQ enviando-a para um destino com WMQ_MESSAGE_BODY configurado como WMQ_MESSAGE_BODY_MQ, não será possível recebê-la, como uma JMSTextMessage a partir do mesmo destino. Todas as mensagens recebidas a partir de um destino com WMQ_MESSAGE_BODY configurado como WMQ_MESSAGE_BODY_MQ são recebidas como uma JMSBytesMessage. Se você tentar receber a mensagem como um JMSTextMessage , isso causará uma exceção, ClassCastException: com.ibm.jms.JMSBytesMessage cannot be cast to javax.jms.TextMessage

Nota: O texto em uma JMSBytesMessage não é convertido pelo cliente JMS. O cliente pode receber apenas o texto na mensagem como uma matriz de bytes. Se a conversão do gerenciador de filas estiver ativada, o texto será convertido por ele, mas o cliente JMS ainda deverá recebê-lo como uma matriz de bytes em um JMSBytesMessage.

Geralmente é melhor usar a propriedade WMQ_TARGET_DEST para controlar se uma JMSTextMessage é enviada com um estilo de corpo MQ ou JMS. É possível, então, receber a mensagem de um destino que tenha WMQ_TARGET_DEST configurado como WMQ_TARGET_DEST_MQ ou WMQ_TARGET_DEST_JMS. WMQ_TARGET_DEST não tem efeito no receptor.

JMSMapMessage e JMSStreamMessage

Esses dois tipos de mensagens JMS são semelhantes É possível ler e gravar os tipos primitivos para as mensagens usando métodos baseados nas interfaces DataInputStream e DataOutputStream; consulte "Tabela de tipos de mensagens e tipos de conversões" na página 851. Os detalhes são descritos em "Conversão de mensagens do cliente JMS e codificação" na página 853. Cada primitivo é identificado; consulte "O corpo da mensagem JMS" na página 837.

Os dados numéricos são lidos e gravados na mensagem codificada como texto XML. Nenhuma referência é feita para a propriedade de destino, JMS_IBM_ENCODING. Os dados de texto são tratados da mesma forma que o texto em uma JMSTextMessage. Se você fosse observar os conteúdos da mensagem criada pelo exemplo em Figura 138 na página 850, todos os dados da mensagem seria em EBCDIC, conforme ela foi enviada com um valor do conjunto de caracteres de 37.

É possível enviar vários itens em uma JMSMapMessage ou JMSStreamMessage.

É possível recuperar os itens de dados individuais pelo nome a partir de uma JMSMapMessage ou pela posição a partir de uma JMSStreamMessage. Cada item será decodificado quando um método get ou read for chamado usando o valor CodedCharacterSetId armazenado na mensagem. Se o método usado para recuperar o item retornar um tipo diferente para o tipo que foi enviado, o tipo será convertido. Se o tipo não puder ser convertido, uma exceção será lançada. Consulte Classe JMSStreamMessage para obter detalhes. O exemplo em "Enviando dados em uma JMSStreamMessage e JMSMapMessage" na página 850 ilustra a conversão de tipo e obtenção do conteúdo JMSMapMessage fora de sequÊncia.

O campo MQRFH2.format para o JMSMapMessage e JMSStreamMessage é configurado como "MQSTR ". Se a propriedade de destino WMQ_RECEIVE_CONVERSION for configurada como WMQ_RECEIVE_CONVERSION_QMGRos dados da mensagem serão convertidos pelo gerenciador de filas antes de serem enviados para o cliente JMS. O MQRFH2.CodedCharacterSetId da mensagem é o WMQ_RECEIVE_CCSID do destino. O MQRFH2.Encoding é Native. Se WMQ_RECEIVE_CONVERSION for WMQ_RECEIVE_CONVERSION_CLIENT_MSG, o CodedCharacterSetId e Encoding do MQRFH2 será o valor configurado pelo emissor.

Um aplicativo cliente JMS pode receber um JMSMapMessage ou JMSStreamMessage apenas em uma mensagem que tenha um corpo de estilo JMS e de um destino que não especifique um corpo de estilo MQ .

JMSBytesMessage

Uma JMSBytesMessage pode conter vários tipos primitivos. É possível ler e gravar os tipos primitivos para as mensagens usando métodos baseados nas interfaces DataInputStream e DataOutputStream; consulte "Tabela de tipos de mensagens e tipos de conversões" na página 851. Os detalhes são descritos em "Tipos de mensagens JMS e conversão" na página 846.

A codificação dos dados numéricos na mensagem é controlada pelo valor de JMS_IBM_ENCODING, que é configurado antes de gravar dados numéricos para o JMSBytesMessage. Um aplicativo pode substituir a codificação padrão Native definida para JMSBytesMessage configurando a propriedade de mensagem JMS_IBM_ENCODING.

Os dados de texto podem ser lidos e gravados em UTF-8 usando o readUTF e writeUTF ou em Unicode usando os métodos readChar e writeChar. Não há métodos que usem o CodedCharacterSetId. Como alternativa, o cliente JMS pode codificar e decodificar o texto em bytes usando a classe Charset Ele transfere os bytes entre a JVM e a mensagem sem as classes WebSphere MQ para JMS que executam qualquer conversão; consulte "Enviando e recebendo texto em uma JMSBytesMessage" na página 850.

Um JMSBytesMessage enviado para um aplicativo MQ geralmente é enviado em um corpo de mensagem no estilo MQ, sem um cabeçalho JMS MQRFH2 Se ele for enviado para um aplicativo JMS, o estilo de corpo da mensagem será geralmente JMS O valor dos atributos de destino, WMQ_MESSAGE_BODY e WMQ_TARGET_DEST determinam o estilo de corpo da mensagem, a menos que substituído pelo aplicativo. O aplicativo pode substituir os valores configurados no destino, chamando destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ) ou destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ).

Se você enviar um JMSBytesMessage com um corpo de estilo MQ , poderá receber a mensagem de um destino que define um estilo de corpo de mensagem MQ ou JMS. Se você enviar um JMSBytesMessage com um corpo de estilo JMS, deverá receber a mensagem de um destino que defina um estilo de corpo de mensagem JMS. Se você não enviar, o MQRFH2 será tratado como parte dos dados da mensagem do usuário, o que pode não ser o que você está esperando.

Se uma mensagem tiver um estilo de corpo MQ ou JMS, a maneira como ela é recebida não será afetada pela configuração WMQ_TARGET_DEST.

A mensagem poderá ser transformada posteriormente, pelo gerenciador de filas, se um Format for fornecido para os dados da mensagem e a conversão de dados do gerenciador de filas estiver ativada. Não use o campo de formato para nada além de especificar o formato dos dados da mensagem ou deixe-o em branco, MQConstants.MQFMT_NONE

É possível enviar vários itens em uma JMSBytesMessage. Cada item numérico será convertido quando a mensagem for enviada usando a codificação definida para a mensagem.

É possível recuperar os itens de dados individuais a partir de JMSBytesMessage. Chame os métodos read na mesma ordem que os métodos write foram chamados para criar a mensagem. Cada item numérico será convertido quando a mensagem for chamada usando o valor Encoding armazenado na mensagem.

Diferentemente de JMSMapMessage e JMSStreamMessage, JMSBytesMessage contém somente dados gravados pelo aplicativo. Nenhum dado adicional é armazenado nos dados da mensagem, como as identificações XML usadas para definir os itens em uma JMSMapMessage e JMSStreamMessage. Por esse motivo, use JMSBytesMessage para transferir mensagens formatadas para outros aplicativos.

A conversão entre JMSBytesMessage e DataInputStream e DataOutputStream é útil em alguns aplicativos. Código baseado no exemplo, "Lendo e gravando mensagens usando DataInputStream e DataOutputStream" na página 850, é necessário para usar o pacote com.ibm.mq.header com JMS.

Examples

Enviando e recebendo uma JMSObjectMessage

```
ObjectMessage omo = session.createObjectMessage();
omo.setObject(new String("A string"));
producer.send(omo);
...
ObjectMessage omi = (ObjectMessage)consumer.receive();
System.out.println((String)omi.getObject());
...
A string
```

Figura 134. Enviando e recebendo uma JMSObjectMessage

Enviando e recebendo um JMSTextmessage

Uma mensagem de texto não pode conter texto em conjuntos de caracteres diferentes. O exemplo mostra texto em conjuntos de caracteres diferentes, enviado em duas mensagens diferentes.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Figura 135. Enviar mensagem de texto no conjunto de caracteres definido pelo destino

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Figura 136. Enviar mensagem de texto em ccsid 37

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Figura 137. Receber mensagem de texto

Enviando dados em uma JMSStreamMessage e JMSMapMessage

```
StreamMessage smo = session.createStreamMessage();
smo.writeString("256");
smo.writeInt(512);
smo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
producer.send(smo);
...
MapMessage mmo = session.createMapMessage();
mmo.setString("First", "256");
mmo.setInt("Second", 512);
mmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
producer.send(mmo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println("Stream: First as float "   + smi.readFloat() +
                         " Second as String " + smi.readString());
...
Stream: First as float: 256.0, Second as String: 512
...
MapMessage mmi = (MapMessage)consumer.receive();
System.out.println("Map: Second as String " + mmi.getString("Second") +
                      " First as double "  + mmi.getDouble("First"));
...
Map: Second as String: 512, First as double: 256.0
```

Figura 138. Envie dados em JMSStreamMessage e JMSMapMessage

Enviando e recebendo texto em uma JMSBytesMessage

O código em Figura 139 na página 850 envia uma sequência em uma BytesMessage. Para simplicidade, o exemplo envia uma sequência única, para a qual uma JMSTextMessage é mais apropriada. Para receber uma sequência de texto em mensagem de bytes contendo uma mistura de tipos, deve-se saber o comprimento da sequência em bytes, chamado *TEXT_LENGTH* em Figura 140 na página 850. Mesmo para uma sequência com um número fixo de caracteres, o comprimento da representação de bytes pode ser maior.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                  .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Figura 139. Enviando uma String em um JMSBytesMessage

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Figura 140. Recebendo um String de um JMSBytesMessage

Lendo e gravando mensagens usando DataInputStream e DataOutputStream

O código em Figura 141 na página 851 cria um JMSBytesMessage usando um DataOutputStream.

```
ByteArrayOutputStream bout = new ByteArrayOutputStream();
DataOutputStream dout = new DataOutputStream(bout);
BytesMessage messageOut = prod.session.createBytesMessage();
// messageOut.setIntProperty(WMQConstants.JMS_IBM_ENCODING,
//                           ((MQDestination) (prod.destination)).getIntProperty
//                           (WMQConstants.WMQ_ENCODING));
int ccsidOut = (((MQDestination)prod.destination).getIntProperty(WMQConstants.WMQ_CCSID));
String codePageOut = CCSID.getCodepage(ccsidOut);
dout.writeInt(ccsidOut);
dout.write(codePageOut.getBytes(codePageOut));
messageOut.writeBytes(bout.toByteArray());
producer.send(messageOut);
```

Figura 141. Envie uma JMSBytesMessage usando um DataOutputStream

A instrução que configura a propriedade JMS_IBM_ENCODING é comentada. A instrução é válida, se estiver gravando diretamente em um JMSBytesMessage, mas não tem efeito ao gravar em DataOutputStream Os números gravados no DataOutputStream são codificados na codificação Native. Configurar JMS_IBM_ENCODING não tem efeito.

O código em Figura 142 na página 851 recebe uma JMSBytesMessage usando um DataInputStream.

```
static final int ccsidIn_SIZE = (Integer.SIZE)/8;
...
connection.start();
BytesMessage messageIn = (BytesMessage) consumer.receive();
int messageLength = new Long(messageIn.getBodyLength()).intValue();
byte [] bin = new byte[messageLength];
messageIn.readBytes(bin, messageLength);
DataInputStream din = new DataInputStream(new ByteArrayInputStream(bin));
int ccsidIn = din.readInt();
byte [] codePageByte = new byte[messageLength - ccsidIn_SIZE];
din.read(codePageByte, 0, codePageByte.length);
System.out.println("CCSID " + ccsidIn + " code page " + new String(codePageByte,
                    messageIn.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET)));
```

Figura 142. Receba uma JMSBytesMessage usando um DataInputStream

A página de códigos é impressa usando a propriedade de página de códigos dos dados da mensagem de entrada, JMS_IBM_CHARACTER_SET. Na entrada JMS_IBM_CHARACTER_SET é uma página de códigos Java e não um identificador de conjunto de caracteres codificados numéricos.

Tabela de tipos de mensagens e tipos de conversões

Tabela 120. Tipos de mensagens e tipos de conversão

	Tipo de conversão			
Tipo de Mensagem	**text**	**Numérica**	**Outro**	**Nenhum**
JMSObjectMessage				getObject setObject
JMSTextMessage	getText setText			

Tabela 120. Tipos de mensagens e tipos de conversão (continuação)				
	Tipo de conversão			
Tipo de Mensagem	**text**	**Numérica**	**Outro**	**Nenhum**
JMSBytesMessage	readUTF writeUTF	readDouble readFloat readInt readLong readShort readUnsignedShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readUnsignedByte readBytes readChar writeByte writeBytes writeChar
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

Conceitos relacionados

Abordagens de conversão de mensagens JMS
Várias abordagens de conversão de dados estão abertas para os designers de aplicativos JMS Essas abordagens não são exclusivas; alguns aplicativos devem usar uma combinação dessas abordagens. Se seu aplicativo estiver trocando apenas texto ou estiver trocando mensagens apenas com outros aplicativos JMS, normalmente você não considera a conversão de dados A conversão de dados é executada automaticamente para você pelo WebSphere MQ.

Conversão de mensagens do cliente JMS e codificação
Os métodos usados para executar a conversão e a codificação da mensagem do cliente JMS são listados, com exemplos de código de cada tipo de conversão.

Conversão de dados do gerenciador de filas
A conversão de dados do gerenciador de filas sempre esteve disponível para aplicativos não JMS que recebem mensagens de clientes JMS Desde a V7.0, os clientes JMS que recebem mensagens também usam a conversão de dados do gerenciador de filas A partir da 7.0.1.5 ou 7.0.1.4 com a APAR IC72897, a conversão de dados do gerenciador de filas é opcional.

Tarefas relacionadas

Trocando um registro formatado com um aplicativo não JMS
Siga as etapas sugeridas nesta tarefa para projetar e construir uma saída de conversão de dados e um aplicativo cliente JMS que possa trocar mensagens com um aplicativo não JMS usando o `JMSBytesMessage` A troca de uma mensagem formatada com um aplicativo não JMS pode ocorrer com ou sem chamar uma saída de conversão de dados.

Conversão de mensagens do cliente JMS e codificação
Os métodos usados para executar a conversão e a codificação da mensagem do cliente JMS são listados, com exemplos de código de cada tipo de conversão.

A conversão e a codificação ocorrem quando primitivas ou objetos Java são lidos ou gravados em e a partir de mensagens JMS. A conversão é chamada de conversão de dados do cliente JMS para distingui-la da conversão de dados do gerenciador de fila e da conversão de dados do aplicativo A conversão ocorre estritamente quando os dados são lidos ou gravados em uma mensagem JMS. O texto é convertido em/de uma representação Unicode interna de 16 bits[6]para o conjunto de caracteres usado para texto em mensagens. Dados numéricos são convertidos em tipos numéricos primitivos Java para a codificação definida para a mensagem. Se a conversão é executada e qual tipo de conversão é executada, depende do tipo de mensagens JMS e da operação de leitura ou gravação.

Tabela 121 na página 853 categoriza os métodos de leitura e de gravação para diferentes tipos de mensagens JMS pelo tipo de conversão executada Os tipos de conversões estão descritos no texto após a tabela.

Tabela 121. Tipos de mensagens e tipos de conversão

	Tipo de conversão			
Tipo de Mensagem	**text**	**Numérica**	**Outro**	**Nenhum**
`JMSObjectMessage`				`getObject` `setObject`
`JMSTextMessage`	`getText` `setText`			

`JMSBytesMessage`	`readUTF`
	`writeUTF`
		`readDouble`
		`readFloat`
		`readInt`
		`readLong`
		`readShort`
		`readUnsignedShort`
		`writeDouble`
		`writeFloat`
		`writeInt`
		`writeLong`
		`writeShort`
			`readBoolean`
			`readObject`
			`writeBoolean`
			`writeObject`
				`readByte`
				`readUnsignedByte`
				`readBytes`
				`readChar`
				`writeByte`
				`writeBytes`
				`writeChar`

[6] Algumas representações Unicode requerem mais de 16 bits. Consulte uma referência do Java SE

Tabela 121. Tipos de mensagens e tipos de conversão (continuação)				
	Tipo de conversão			
Tipo de Mensagem	**text**	**Numérica**	**Outro**	**Nenhum**
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

text

O CodedCharacterSetId padrão para um destino é 1208, UTF-8. Por padrão, o texto é convertido de Unicode e enviado como uma sequência de texto UTF-8. No recebimento, o texto é convertido do conjunto de caracteres codificados na mensagem recebida pelo cliente para Unicode.

Os métodos setText e writeString convertem texto de Unicode no conjunto de caracteres definido para o destino. Um aplicativo pode substituir o conjunto de caracteres de destino configurando a propriedade de mensagem JMS_IBM_CHARACTER_SET. Ao enviar uma mensagem, JMS_IBM_CHARACTER_SET deve ser um identificador de conjunto de caracteres codificados numéricos[7].

Os segmentos de código em "Enviando e recebendo um JMSTextmessage" na página 856 enviam duas mensagens. Uma é enviada no conjunto de caracteres definido para o destino e a outra no conjunto de caracteres 37 definido pelo aplicativo.

Os métodos getText e readString convertem o texto na mensagem do conjunto de caracteres definido na mensagem para Unicode. Os métodos usam a página de códigos definida na propriedade de mensagem, JMS_IBM_CHARACTER_SET. A página de códigos é mapeada de MQRFH2.CodedCharacterSetId a menos que a mensagem seja uma mensagem do tipo MQ e não tenha nenhum MQRFH2. Se a mensagem for uma mensagem do tipo MQ, sem nenhum MQRFH2, a página de códigos será mapeada de MQMD.CodedCharacterSetId.

O fragmento de código em Figura 147 na página 857 recebe a mensagem que foi enviada para o destino. O texto na mensagem é convertido da página de códigos IBM037 novamente para Unicode.

Nota: Uma maneira simples de verificar se o texto é convertido em conjunto de caracteres codificados 37 é usar o WebSphere MQ Explorer. Procure a fila e mostre as propriedades da mensagem antes que ela seja recuperada.

[7] Ao receber uma mensagem JMS_IBM_CHARACTER_SET é um nome da página de códigos Java Charset .

Compare o fragmento de código em Figura 146 na página 857 ao fragmento de código incorreto em Figura 143 na página 855. No fragmento incorreto, a cadeia de texto é convertida duas vezes, uma pelo aplicativo e novamente pelo WebSphere MQ.

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText(new String("Sent in EBCDIC character set 37".getBytes(CCSID.getCodepage(37))));
producer.send(tmo);
```

Figura 143. Conversão de página de códigos incorreta

O método `writeUTF` converte texto de Unicode para 1208, UTF-8. A sequência de texto tem como prefácio um comprimento de 2 bytes. O comprimento máximo da sequência de texto é 65534 bytes. O método `readUTF` lê um item em uma mensagem gravada pelo método `writeUTF`. Ele lê exatamente o número de bytes gravados pelo método `writeUTF`.

Numérica

A codificação numérica padrão para um destino é `Native`. A constante de codificação `Native` para Java tem o valor 273, `x'00000111'`, que é o mesmo para todas as plataformas. No recebimento, os números na mensagem são transformados corretamente em primitivas numéricas Java. A transformação usa a codificação definida na mensagem e o tipo retornado pelo método de leitura.

O método de envio converte números que são incluídos em uma mensagem por `set` e `write` para a codificação numérica definida para o destino. A codificação de destino pode ser substituída para uma mensagem por um aplicativo que configura a propriedade de mensagem, `JMS_IBM_ENCODING`; por exemplo:

```
message.setIntProperty(WMQConstants.JMS_IBM_ENCODING,
WMQConstants.WMQ_ENCODING_INTEGER_REVERSED);
```

Os métodos numéricos `get` e `read` convertem números na mensagem da codificação numérica definida na mensagem. Eles convertem os números para o tipo especificado pelo método `read` ou `get` ; consulte A propriedade ENCODING . Os métodos usam a codificação definida em `JMS_IBM_ENCODING`. A codificação é mapeada de `MQRFH2.Encoding`, a menos que a mensagem seja uma mensagem do tipo MQ e não tenha nenhum `MQRFH2`. Se a mensagem for uma mensagem do tipo MQ, sem nenhum `MQRFH2`, então, os métodos usam a codificação definida em `MQMD.Encoding`.

O exemplo em Figura 148 na página 857 mostra um aplicativo codificando um número no formato de destino e enviando-o em um `JMSStreamMessage`. Compare o exemplo em Figura 148 na página 857 ao exemplo em Figura 149 na página 857. A diferença é que `JMS_IBM_ENCODING` deve ser configurado em um `JMSBytesMessage`.

Nota: Uma maneira simples de verificar se o número está codificado corretamente é usar o WebSphere MQ Explorer. Procure na fila e mostre as propriedades da mensagem antes de ser consumida.

Outro

Os métodos `boolean` codificam `true` e `false` como `x'01'` e `x'00'` em um `JMSByteMessage`, `JMSStreamMessage` e `JMSMapMessage`.

Os métodos UTF codificam e decodificam Unicode para sequências de texto UTF-8. As sequências são limitadas a menos de 65536 caracteres e são precedidas pelo campo de comprimento de 2 bytes.

Os métodos Object agrupam tipos primitivos como objetos. Tipos numéricos e de texto são codificados ou convertidos como se os tipos primitivos tivessem sido lidos ou gravados usando os métodos numérico e de texto.

Nenhum

Os métodos `readByte`, `readBytes`, `readUnsignedByte`, `writeByte` e `writeBytes` efetuam get ou put de bytes únicos, ou matrizes de bytes, entre o aplicativo e a mensagem sem conversão. Os

métodos readChar e writeChar efetuam get e put caracteres Unicode de 2 bytes entre o aplicativo e a mensagem sem conversão.

Usando os métodos readBytes e writeBytes, o aplicativo pode executar sua própria conversão de ponto de código, como em "Enviando e recebendo texto em uma JMSBytesMessage" na página 857.

WebSphere MQ não executa nenhuma conversão de página de códigos no cliente porque a mensagem é um JMSBytesMessagee porque os métodos readBytes e writeBytes são usados. Contudo, se os bytes representarem texto, certifique-se de que a página de códigos usada pelo aplicativo corresponda ao conjunto de caracteres codificados do destino. A mensagem pode ser convertida novamente por uma saída de conversão do gerenciador de filas. Outra possibilidade é que o programa cliente JMS de recebimento possa seguir a convenção de conversão de quaisquer matrizes de bytes que representam texto na mensagem em sequências ou caracteres usando a propriedade JMS_IBM_CHARACTER_SET na mensagem.

Neste exemplo, o cliente usa o conjunto de caracteres codificados de destino para sua conversão:

```
bytes.writeBytes("In the destination code page".getBytes(
      CCSID.getCodepage(((MQDestination) destination)
           .getIntProperty(WMQConstants.WMQ_CCSID))));
```

Como alternativa, o cliente pode ter escolhido uma página de códigos e, em seguida, configurado o conjunto de caracteres codificados correspondente na propriedade JMS_IBM_CHARACTER_SET da mensagem. As classes do WebSphere MQ para Java usam JMS_IBM_CHARACTER_SET para configurar o campo CodedCharacterSetId nas propriedades JMS no MQRFH2ou no descritor de mensagem, MQMD:

```
String codePage = CCSID.getCodepage(37);
message.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage);[8]
```

Se uma matriz de bytes for gravada em um JMSStringMessage ou JMSMapMessage, WebSphere MQ classes para JMS não executará a conversão de dados, pois os bytes são digitados como dados hexadecimais e não como texto no JMSStringMessage e JMSMapMessage.

Se os bytes representarem caracteres em seu aplicativo, deve-se levar em consideração quais pontos de código ler e gravar na mensagem. O código em Figura 144 na página 856 segue a convenção de usar o conjunto de caracteres codificados de destino. Se você criar a sequência usando o conjunto de caracteres padrão para a JVM, os conteúdos de bytes dependem da plataforma. Uma JVM no Windows geralmente tem um padrão Charset de windows-1252e UNIX, UTF-8 A troca entre Windows e UNIX requer que você selecione uma página de códigos explícita para trocar texto como bytes.

```
StreamMessage smo = producer.session.createStreamMessage();
smo.writeBytes("123".getBytes(CCSID.getCodepage(((MQDestination) destination)
           .getIntProperty(WMQConstants.WMQ_CCSID))));
```

Figura 144. Gravando bytes que representam uma sequência em uma JMSStreamMessage usando o conjunto de caracteres de destino

---

Examples

Enviando e recebendo um JMSTextmessage

Uma mensagem de texto não pode conter texto em conjuntos de caracteres diferentes. O exemplo mostra texto em conjuntos de caracteres diferentes, enviado em duas mensagens diferentes.

[8] SetStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage) currently accepts only numeric character set identifiers.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Figura 145. Enviar mensagem de texto no conjunto de caracteres definido pelo destino

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Figura 146. Enviar mensagem de texto em `ccsid 37`

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Figura 147. Receber mensagem de texto

Exemplos de codificação

Exemplos mostrando um número que está sendo enviado na codificação definida para um destino. Observe que se deve configurar a propriedade `JMS_IBM_ENCODING` de um `JMSBytesMessage` para o valor especificado para o destino.

```
StreamMessage smo = session.createStreamMessage();
smo.writeInt(256);
producer.send(smo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println(smi.readInt());
...
256
```

Figura 148. Enviando um número usando a codificação de destino em um `JMSStreamMessage`

```
BytesMessage bmo = session.createBytesMessage();
bmo.writeInt(256);
int encoding = ((MQDestination) (destination)).getIntProperty
    (WMQConstants.WMQ_ENCODING)
bmo.setIntProperty(WMQConstants.JMS_IBM_ENCODING, encoding);
producer.send(bmo);
...
BytesMessage bmi = (BytesMessage)consumer.receive();
System.out.println(bmi.readInt());
...
256
```

Figura 149. Enviando um número usando a codificação de destino em um `JMSBytesMessage`

Enviando e recebendo texto em uma `JMSBytesMessage`

O código em Figura 150 na página 858 envia uma sequência em uma BytesMessage. Para simplicidade, o exemplo envia uma sequência única, para a qual uma `JMSTextMessage` é mais apropriada. Para receber uma sequência de texto em mensagem de bytes contendo uma mistura de tipos, deve-se saber

o comprimento da sequência em bytes, chamado *TEXT_LENGTH* em Figura 151 na página 858. Mesmo para uma sequência com um número fixo de caracteres, o comprimento da representação de bytes pode ser maior.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                  .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Figura 150. Enviando uma `String` em um `JMSBytesMessage`

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Figura 151. Recebendo um `String` de um `JMSBytesMessage`

Conceitos relacionados

Abordagens de conversão de mensagens JMS
Várias abordagens de conversão de dados estão abertas para os designers de aplicativos JMS Essas abordagens não são exclusivas; alguns aplicativos devem usar uma combinação dessas abordagens. Se seu aplicativo estiver trocando apenas texto ou estiver trocando mensagens apenas com outros aplicativos JMS, normalmente você não considera a conversão de dados A conversão de dados é executada automaticamente para você pelo WebSphere MQ.

Conversão de dados do gerenciador de filas
A conversão de dados do gerenciador de filas sempre esteve disponível para aplicativos não JMS que recebem mensagens de clientes JMS Desde a V7.0, os clientes JMS que recebem mensagens também usam a conversão de dados do gerenciador de filas A partir da 7.0.1.5 ou 7.0.1.4 com a APAR IC72897, a conversão de dados do gerenciador de filas é opcional.

Tarefas relacionadas

Trocando um registro formatado com um aplicativo não JMS
Siga as etapas sugeridas nesta tarefa para projetar e construir uma saída de conversão de dados e um aplicativo cliente JMS que possa trocar mensagens com um aplicativo não JMS usando o `JMSBytesMessage` A troca de uma mensagem formatada com um aplicativo não JMS pode ocorrer com ou sem chamar uma saída de conversão de dados.

Referências relacionadas

Tipos de mensagens JMS e conversão
A opção de tipo de mensagem afeta sua abordagem de conversão de mensagens. A interação da conversão de mensagem com o tipo de mensagem é descrita para os tipos de mensagens JMS, `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessagee` `JMSBytesMessage`

Conversão de dados do gerenciador de filas
A conversão de dados do gerenciador de filas sempre esteve disponível para aplicativos não JMS que recebem mensagens de clientes JMS Desde a V7.0, os clientes JMS que recebem mensagens também usam a conversão de dados do gerenciador de filas A partir da 7.0.1.5 ou 7.0.1.4 com a APAR IC72897, a conversão de dados do gerenciador de filas é opcional.

O gerenciador de filas pode converter dados de caracteres e numéricos em dados da mensagem usando os valores de `CodedCharacterSetId`, `Encoding` e `Format` configurados para os dados da mensagem. Para aplicativos não JMS, o recurso de conversão sempre esteve disponível configurando a `GetMessage`, `GMO_CONVERT` O recurso de conversão do gerenciador de fila não estava disponível para um aplicativo JMS recebendo uma mensagem até V7.0

É possível usar a conversão do gerenciador de fila, antes da V7.0, com um aplicativo cliente JMS que envia uma mensagem O cliente JMS cria um registro formatado, configura os atributos `CodedCharacterSetId`, `Encoding`e `Format` correspondentes aos dados colocados na mensagem. Um aplicativo de recebimento não JMS lê a mensagem usando `GMO_CONVERT`e faz com que uma saída de conversão de dados gravados pelo usuário seja chamada. A saída de conversão de dados é uma biblioteca compartilhada que tem o nome configurado no campo `Format`.

Desde o V7.0, o gerenciador de filas é capaz de converter mensagens que são enviadas para clientes JMS De 7.0.0.0 para 7.0.1.4 inclusivo, a conversão do gerenciador de filas é sempre chamada para clientes JMS Na 7.0.1.5 ou na 7.0.1.4 com o APAR IC72897 aplicado, a conversão do gerenciador de filas é controlada configurando a propriedade de destino `WMQ_RECEIVE_CONVERSION` para `WMQ_RECEIVE_CONVERSION_QMGR` ou `WMQ_RECEIVE_CONVERSION_CLIENT_MSG`. `WMQ_RECEIVE_CONVERSION_CLIENT_MSG` é a configuração padrão, correspondendo ao comportamento do WebSphere MQ V6.0, que não suportou a conversão de dados do gerenciador de filas para clientes JMS O aplicativo pode mudar a configuração de destino:

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Ou

```
((MQDestination)destination).setReceiveConversion
             (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Figura 152. Ativar conversão de dados do gerenciador de filas

A conversão de dados do gerenciador de filas para um cliente JMS ocorre quando o cliente chama um método `consumer.receive` .. Dados de texto são transformados em UTF-8 (1208) por padrão. Métodos de leitura e obtenção subsequentes decodificam texto nos dados recebidos de UTF-8, criando primitivas de texto Java em sua codificação Unicode interna. UTF-8 não é o único conjunto de caracteres de destino de conversão de dados do gerenciador de filas. É possível escolher um CCSID diferente configurando a propriedade de destino `WMQ_RECEIVE_CCSID`.

Um aplicativo também pode mudar a configuração de destino, por exemplo, configurando-o para 437, DOS-US:

```
((MQDestination)destination).setIntProperty
             (WMQConstants.WMQ_RECEIVE_CCSID, 437);
```

Ou

```
((MQDestination)destination).setReceiveCCSID(437);
```

Figura 153. Configurar o conjunto de caracteres codificado de destino para a conversão do gerenciador de filas

A razão para mudar `WMQ_RECEIVE_CCSID` é especializada; o CCSID escolhido não faz diferença para os objetos de texto criados na JVM. No entanto, algumas JVMs, em algumas plataformas, podem não ser capazes de manipular a conversão do CCSID de texto na mensagem para Unicode. A opção fornece uma opção de CCSID para qualquer texto entregue ao cliente na mensagem. Algumas plataformas do cliente JMS tiveram problemas com o texto da mensagem sendo entregue em UTF-8.

O código JMS é equivalente ao texto em negrito no código C em Figura 154 na página 860,

```
gmo.Options = MQGMO_WAIT         /* wait for new messages           */
            | MQGMO_NO_SYNCPOINT /* no transaction                  */
            | MQGMO_CONVERT;     /* convert if necessary            */

     while (CompCode != MQCC_FAILED) {
     buflen = sizeof(buffer) - 1; /* buffer size available for GET   */
     memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
     memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
     md.Encoding       = MQENC_NATIVE;
     md.CodedCharSetId = MQCCSI_Q_MGR;

           MQGET(Hcon,          /* connection handle               */
           Hobj,                /* object handle                   */
           &md,                 /* message descriptor              */
           &gmo,                /* get message options             */
           buflen,              /* buffer length                   */
           buffer,              /* message buffer                  */
           &messlen,            /* message length                  */
           &CompCode,           /* completion code                 */
           &Reason);            /* reason code                     */
```

Figura 154. Fragmento de código de amqsget0.c

Nota:

A conversão do gerenciador de filas é executada apenas nos dados da mensagem que possuem um formato conhecido do WebSphere MQ MQSTR, ou MQCIH são exemplos de formatos conhecidos predefinidos. Um formato conhecido também pode ser um formato definido pelo usuário, contanto que você tenha fornecido uma saída de conversão de dados.

As mensagens construídas como JMSTextMessage, JMSMapMessage e JMSStreamMessage têm um formato MQSTR e podem ser convertidas pelo gerenciador de filas.

Conceitos relacionados

Abordagens de conversão de mensagens JMS
Várias abordagens de conversão de dados estão abertas para os designers de aplicativos JMS Essas abordagens não são exclusivas; alguns aplicativos devem usar uma combinação dessas abordagens. Se seu aplicativo estiver trocando apenas texto ou estiver trocando mensagens apenas com outros aplicativos JMS, normalmente você não considera a conversão de dados A conversão de dados é executada automaticamente para você pelo WebSphere MQ.

Conversão de mensagens do cliente JMS e codificação
Os métodos usados para executar a conversão e a codificação da mensagem do cliente JMS são listados, com exemplos de código de cada tipo de conversão.

"Chamando a saída de conversão de dados" na página 422
Uma saída de conversão de dados é uma saída escrita pelo usuário que recebe controle durante o processamento de uma chamada MQGET.

Tarefas relacionadas

Trocando um registro formatado com um aplicativo não JMS
Siga as etapas sugeridas nesta tarefa para projetar e construir uma saída de conversão de dados e um aplicativo cliente JMS que possa trocar mensagens com um aplicativo não JMS usando o JMSBytesMessage A troca de uma mensagem formatada com um aplicativo não JMS pode ocorrer com ou sem chamar uma saída de conversão de dados.

Referências relacionadas

Tipos de mensagens JMS e conversão
A opção de tipo de mensagem afeta sua abordagem de conversão de mensagens. A interação da conversão de mensagem com o tipo de mensagem é descrita para os tipos de mensagens JMS, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessagee JMSBytesMessage

Trocando um registro formatado com um aplicativo não JMS
Siga as etapas sugeridas nesta tarefa para projetar e construir uma saída de conversão de dados e um aplicativo cliente JMS que possa trocar mensagens com um aplicativo não JMS usando o

JMSBytesMessage A troca de uma mensagem formatada com um aplicativo não JMS pode ocorrer com ou sem chamar uma saída de conversão de dados.

Antes de começar

Você pode ser capaz de projetar uma solução mais simples para trocar mensagens com um aplicativo não JMS usando um JMSTextMessage. Elimine que possibilidade antes de seguir as etapas nesta tarefa.

Sobre esta tarefa

Um cliente JMS é mais fácil de gravar se ele não estiver envolvido nos detalhes de formatação de mensagens JMS trocadas com outros clientes JMS Contanto que o tipo de mensagem seja JMSTextMessage, JMSMapMessage, JMSStreamMessageou JMSObjectMessage, o WebSphere MQ procurará os detalhes da formatação da mensagem O WebSphere MQ lida com diferenças em páginas de códigos e codificação numérica em diferentes plataformas.

É possível usar esses tipos de mensagens para trocar mensagens com aplicativos não JMS. Para isso, você deve entender como essas mensagens são construídas pelas classes do WebSphere MQ para JMS. É possível modificar o aplicativo não JMS para interpretar as mensagens; consulte “Mapeando mensagens JMS para mensagens do WebSphere MQ” na página 822.

Uma vantagem de usar um desses tipos de mensagens é que a programação do cliente JMS não depende do tipo de aplicativo com o qual está trocando mensagens. Uma desvantagem é que ela pode requerer uma modificação para outro programa e você pode não ser capaz de mudar o outro programa.

Uma abordagem alternativa é gravar um aplicativo cliente JMS que possa lidar com os formatos de mensagens existentes Geralmente as mensagens existentes são formatos fixos e contêm uma mistura de dados não formatados, textos e números. Use as etapas nesta tarefa e o cliente JMS de exemplo no “Gravando classes para encapsular um layout de registro em uma JMSBytesMessage” na página 864, como um ponto de início para construir um cliente JMS que possa trocar registros formatado com aplicativos não JMS

Procedimento

1. Defina o layout do registro, ou use uma das classes de cabeçalho predefinidas do WebSphere MQ

   Para manipular cabeçalhos predefinidos do WebSphere MQ , consulte Manipulação de cabeçalhos de mensagens do WebSphere MQ ..

   Figura 155 na página 862 é um exemplo de um usuário definido, layout de registro de comprimento fixo, que pode ser processado pelo utilitário de conversão de dados.
2. Crie a saída de conversão de dados.

   Siga as instruções em Gravando um programa de saída de conversão de dados para gravar uma saída de conversão de dados.

   Para tentar o exemplo em “Gravando classes para encapsular um layout de registro em uma JMSBytesMessage” na página 864, nomeie a saída de conversão de dados MYRECORD.
3. Grave classes Java para encapsular o layout de registro e enviar e receber registro. Duas abordagens que você pode executar são:

   - Grave uma classe para que leia e grave o JMSBytesMessage que contém o registro; consulte “Gravando classes para encapsular um layout de registro em uma JMSBytesMessage” na página 864.
   - Grave uma classe estendendo com.ibm.mq.header.Header para definir a estrutura de dados do registro; consulte Criando classes para novos tipos de cabeçalho.
4. Decida qual conjunto de caracteres codificados ao qual trocar mensagens.

   Consulte o “Escolha uma abordagem para a conversão de mensagem: receiver makes good” na página 842.
5. Configure o destino para trocar mensagens do tipo MQsem um cabeçalho JMS MQRFH2 .

O destino de envio e de recebimento devem estar configurados para trocar mensagens do MQ-type. É possível usar o mesmo destino para o envio e o recebimento.

O aplicativo pode substituir a propriedade do corpo da mensagem de destino:

```
((MQDestination)destination).setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

O exemplo na "Gravando classes para encapsular um layout de registro em uma JMSBytesMessage" na página 864 substitui a propriedade do corpo da mensagem de destino, assegurando que uma mensagem estilo MQ seja enviada.

6. Teste a solução com aplicativos JMS e não JMS

   As ferramentas úteis para testar uma saída de conversão de dados são:

   - O programa de amostra `amqsgetc0.c` é útil para testar o recebimento de uma mensagem enviada por um cliente JMS.. Consulte as modificações sugeridas para usar o cabeçalho de exemplo, `RECORD.h`, em Figura 156 na página 863. Com as modificações, o `amqsgetc0.c` recebe uma mensagem enviada pelo cliente JMS de exemplo, `TryMyRecord.java`; consulte "Gravando classes para encapsular um layout de registro em uma JMSBytesMessage" na página 864
   - O programa de navegação do WebSphere MQ `amqsbcg0.c`de amostra é útil para inspecionar o conteúdo do cabeçalho da mensagem, o cabeçalho JMS, `MQRFH2`e o conteúdo da mensagem.
   - O programa **`rfhutil`** , anteriormente disponível em SupportPac IH03, permite que as mensagens de teste sejam capturadas e armazenadas em arquivos e, em seguida, usadas para conduzir Fluxos de mensagens As mensagens de saída também podem ser lidas e exibidas em uma variedade de formatos. Os formatos incluem dois tipos de XML, bem como correspondências com relação a um copybook COBOL. Os dados podem estar em EBCDIC ou ASCII. Um cabeçalho RFH2 pode ser incluído na mensagem antes de a mensagem ser enviada.

   Se você tentar receber mensagens usando o programa de amostra `amqsgetc0.c` modificado e obter um erro com o código de razão `2080`, verifique se a mensagem possui um `MQRFH2`. As modificações presumem que a mensagem foi enviada para um destino que não especifica nenhum `MQRFH2`.

Examples

```
struct RECORD { MQCHAR StrucID[4];
                MQLONG Version;
                MQLONG StructLength;
                MQLONG Encoding;
                MQLONG CodeCharSetId;
                MQCHAR Format[8];
                MQLONG Flags;
                MQCHAR RecordData[32];
    };
```

Figura 155. RECORD.h

- Declare a estrutura de dados RECORD.h

```
struct tagRECORD {
   MQCHAR4    StrucId;
   MQLONG     Version;
   MQLONG     StrucLength;
   MQLONG     Encoding;
   MQLONG     CCSID;
   MQCHAR8    Format;
   MQLONG     Flags;
   MQCHAR32   RecordData;
 };
typedef struct tagRECORD RECORD;
   typedef RECORD MQPOINTER PRECORD;
   RECORD record;
   PRECORD pRecord = &(record);
```

- Modifique a chamada MQGET para usar o RECORD,

  1. Antes da modificação:

```
    MQGET(Hcon,            /* connection handle     */
          Hobj,            /* object handle         */
          &md,             /* message descriptor    */
          &gmo,            /* get message options   */
          buflen,          /* buffer length         */
          buffer,          /* message buffer        */
          &messlen,        /* message length        */
          &CompCode,       /* completion code       */
          &Reason);        /* reason code           */
```

  2. Após a modificação:

```
    MQGET(Hcon,            /* connection handle     */
          Hobj,            /* object handle         */
          &md,             /* message descriptor    */
          &gmo,            /* get message options   */
          sizeof(RECORD),  /* buffer length         */
          pRecord,         /* message buffer        */
          &messlen,        /* message length        */
          &CompCode,       /* completion code       */
          &Reason);        /* reason code           */
```

- Mude a instrução de impressão,

  1. De:

```
buffer[messlen] = '\0';            /* add terminator          */
printf("message <%s>\n", buffer);
```

  2. Para:

```
/* buffer[messlen] = '\0';            add terminator          */
printf("ccsid <%d>, flags <%d>, message <%32.32s>\n \0",
        md.CodedCharSetId, record.Flags, record.RecordData);
```

Figura 156. Modifique amqsget0.c

Conceitos relacionados

Abordagens de conversão de mensagens JMS

Várias abordagens de conversão de dados estão abertas para os designers de aplicativos JMS Essas abordagens não são exclusivas; alguns aplicativos devem usar uma combinação dessas abordagens. Se seu aplicativo estiver trocando apenas texto ou estiver trocando mensagens apenas com outros aplicativos JMS, normalmente você não considera a conversão de dados A conversão de dados é executada automaticamente para você pelo WebSphere MQ.

Conversão de mensagens do cliente JMS e codificação

Os métodos usados para executar a conversão e a codificação da mensagem do cliente JMS são listados, com exemplos de código de cada tipo de conversão.

Conversão de dados do gerenciador de filas
A conversão de dados do gerenciador de filas sempre esteve disponível para aplicativos não JMS que recebem mensagens de clientes JMS Desde a V7.0, os clientes JMS que recebem mensagens também usam a conversão de dados do gerenciador de filas A partir da 7.0.1.5 ou 7.0.1.4 com a APAR IC72897, a conversão de dados do gerenciador de filas é opcional.

Referências relacionadas

Tipos de mensagens JMS e conversão
A opção de tipo de mensagem afeta sua abordagem de conversão de mensagens. A interação da conversão de mensagem com o tipo de mensagem é descrita para os tipos de mensagens JMS, `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessagee` `JMSBytesMessage`

Utilitário para criação de código de saída de conversão

Gravando classes para encapsular um layout de registro em uma `JMSBytesMessage`
A finalidade desta tarefa é explorar, por exemplo, como combinar a conversão de dados e um layout de registro fixo em um `JMSBytesMessage`. Na tarefa, você cria algumas classes Java para trocar uma estrutura de registro de exemplo em um `JMSBytesMessage` É possível modificar o exemplo para gravar classes para trocar outras estruturas de registro.

Um `JMSBytesMessage` é a melhor opção de tipo de mensagem JMS para trocar registros de tipo de dados combinados com programas não JMS. Não há dados adicionais inseridos no corpo da mensagem pelo provedor JMS. É, portanto, a melhor opção de tipo de mensagem a ser usada se um programa cliente JMS interoperar com um programa IBM WebSphere MQ existente O principal desafio em usar um `JMSBytesMessage` está em corresponder à codificação e ao conjunto de caracteres esperados pelo outro programa. Uma solução é criar uma classe que contenha o registro. Uma classe que encapsula a leitura e gravação de um `JMSBytesMessage`, para um tipo de registro específico, facilita o envio e recebimento de registros de formato fixo em um programa JMS. Ao capturar os aspectos genéricos da interface em uma classe abstrata, grande parte da solução poderá ser reutilizada para formatos de registros diferentes. Os formatos de registros diferentes podem ser implementados em classes que estendem a classe genérica abstrata.

Uma abordagem alternativa é estender a classe `com.ibm.mq.headers.Header`. A classe `Header` tem métodos, como `addMQLONG`, para construir um formato de registro de uma maneira mais declarativa. A desvantagem de usar a classe `Header` é obter e configurar atributos usando uma interface interpretativa mais complicada. As duas abordagens resultam na mesma quantidade do código do aplicativo.

Um `JMSBytesMessage` pode conter apenas um único formato, além de um `MQRFH2`, em uma mensagem, a menos que cada registro use o mesmo formato, conjunto de caracteres codificado e codificação. O formato, a codificação e o conjunto de caracteres de um `JMSBytesMessage` são propriedades de todas as mensagens a seguir no `MQRFH2`. O exemplo é escrito na suposição de que um `JMSBytesMessage` contém apenas um registro do usuário.

Antes de começar

1. Seu nível de habilidade: você deve estar familiarizado com programação Java e JMS. Nenhuma instrução é fornecida sobre a configuração do ambiente de desenvolvimento Java É vantajoso ter gravado um programa para trocar uma `JMSTextMessage`, `JMSStreamMessage` ou `JMSMapMessage`. É possível ver as diferenças em troca de uma mensagem usando um `JMSBytesMessage`.
2. O exemplo requer IBM WebSphere MQ V7.0.
3. O exemplo foi criado usando a perspectiva Java do ambiente de trabalho Eclipse . Ele requer o JRE 6.0 ou superior. É possível usar a perspectiva Java no IBM WebSphere MQ Explorer para desenvolver e executar as classes Java Como alternativa, use o seu próprio ambiente de desenvolvimento Java
4. Usando o IBM WebSphere MQ Explorer torna a configuração do ambiente de teste e depuração, mais simples usando os utilitários de linha de comandos.

Sobre esta tarefa

Você é guiado através da criação de duas classes: RECORD e MyRecord. Juntos, essas duas classes contêm um registro de formato fixo. Eles têm métodos para obter e configurar os atributos. O método get lê o registro a partir de um JMSBytesMessage e o método put grava um registro em um JMSBytesMessage.

O propósito da tarefa é não criar uma classe de qualidade de produção que possa ser reutilizada. Você pode escolher usar os exemplos na tarefa para iniciar em suas próprias classes. A propósito da tarefa é fornecer notas de orientação, principalmente sobre o uso de conjuntos de caracteres, formatos e codificação, ao usar um JMSBytesMessage. Cada etapa da criação das classes é explicada e os aspectos de uso do JMSBytesMessage, que às vezes são omitidos, são descritos.

A classe RECORD é abstrata e define alguns campos comuns para um registro do usuário. Os campos comuns são modelados no layout de cabeçalho padrão do IBM WebSphere MQ que tem um destaque, uma versão e um campo de comprimento. A codificação, o conjunto de caracteres e os campos de formato, localizados em muitos cabeçalhos do IBM WebSphere MQ, são omitidos. Outro cabeçalho não pode seguir um formato definido pelo usuário. A classe MyRecord, que estende a classe RECORD, o faz literalmente, estendendo o registro com campos de usuário adicionais. Uma JMSBytesMessage, criada pelas classes, pode ser processada pela saída de conversão de dados do gerenciador de filas.

"Classes usadas para executar o exemplo" na página 871 inclui uma listagem completa de RECORD e MyRecord. Ele também inclui listagens de classes "andaime" extras para testar o RECORD e MyRecord. As classes extras são:

TryMyRecord
O programa principal para testar RECORD e MyRecord.

EndPoint
Uma classe abstrata que contém a conexão, o destino e a sessão JMS em uma única classe. Sua interface simplesmente atende às necessidades de teste das classes RECORD e MyRecord. Ele não é um padrão de design estabelecido para gravar aplicativos JMS.

Nota: A classe Endpoint incluirá esta linha de código após criar um destino:

```
((MQDestination)destination).setReceiveConversion
                            (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Em V7.0, a partir de V7.0.1.5, é necessário para ativar a conversão do gerenciador de filas. Ele está desativado por padrão. Em V7.0, até V7.0.1.4, a conversão do gerenciador de filas é ativada por padrão e essa linha de códigos causa um erro.

MyProducer e MyConsumer
As classes que estendem EndPoint e criam um MessageConsumer e MessageProducer, conectadas e prontas para aceitar as solicitações.

Juntas todas as classes formam um aplicativo completo o qual é possível construir e testar, para entender como usar a conversão de dados em um JMSBytesMessage.

Procedimento

1. Crie uma classe abstrata para conter os campos padrão em um cabeçalho IBM WebSphere MQ, com um construtor padrão. Posteriormente, estenda a classe para customizar o cabeçalho para seus requisitos.

```
public abstract class RECORD implements Serializable {
    private static final long serialVersionUID = -1616617232750561712L;
    protected final static int UTF8 = 1208;
    protected final static int MQLONG_LENGTH = 4;
    protected final static int RECORD_STRUCT_ID_LENGTH = 4;
    protected final static int RECORD_VERSION_1 = 1;
    protected final String RECORD_STRUCT_ID = "BLNK";
    protected final String RECORD_TYPE = "BLANK   ";
    private String structID = RECORD_STRUCT_ID;
    private int version = RECORD_VERSION_1;
    private int structLength = RECORD_STRUCT_ID_LENGTH + MQLONG_LENGTH * 2;
```

```
    private int headerEncoding = WMQConstants.WMQ_ENCODING_NATIVE;
    private String headerCharset = "UTF-8";
    private String headerFormat = RECORD_TYPE;

    public RECORD() {
        super();
    }
```

Nota:

a. Os atributos, `structID` para `nextFormat`, são listados na ordem em que são definidos em um cabeçalho da mensagem padrão do IBM WebSphere MQ.

b. Os atributos, `format`, `messageEncoding` e `messageCharset`, descrevem o próprio cabeçalho e não fazem parte do cabeçalho.

c. Deve-se decidir se armazenar o identificador do conjunto de caracteres codificados ou o conjunto de caracteres do registro. Java usa conjuntos de caracteres e IBM WebSphere MQ mensagens usam identificadores de conjunto de caracteres codificado... O código de exemplo usa conjuntos de caracteres.

d. `int` é serializado para `MQLONG` pelo IBM WebSphere MQ. `MQLONG` é de 4 bytes.

2. Crie os getters e setters para os atributos privados.

   a) Crie ou gere os getters:

```
    public String getHeaderFormat() { return headerFormat; }
    public int getHeaderEncoding() { return headerEncoding;    }
    public String getMessageCharset() { return headerCharset; }
    public int getMessageEncoding() { return headerEncoding; }
    public String getStructID() { return structID; }
    public int getStructLength() { return structLength; }
    public int getVersion() { return version; }
```

   b) Crie ou gere os setters:

```
    public void setHeaderCharset(String charset) {
        this.headerCharset = charset; }
    public void setHeaderEncoding(int encoding) {
        this.headerEncoding = encoding; }
    public void setHeaderFormat(String headerFormat) {
        this.headerFormat = headerFormat; }
    public void setStructID(String structID) {
        this.structID = structID; }
    public void setStructLength(int structLength) {
        this.structLength = structLength; }
    public void setVersion(int version) {
        this.version = version; }
}
```

3. Crie um construtor para criar uma instância RECORD a partir de um `JMSBytesMessage`.

```
    public RECORD(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super();
        setHeaderCharset(message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET));
        setHeaderEncoding(message.getIntProperty(WMQConstants.JMS_IBM_ENCODING));
        byte[] structID = new byte[RECORD_STRUCT_ID_LENGTH];
        message.readBytes(structID, RECORD_STRUCT_ID_LENGTH);
        setStructID(new String(structID, getMessageCharset()));
        setVersion(message.readInt());
        setStructLength(message.readInt());
    }
```

Nota:

a. O `messageCharset` e `messageEncoding` são capturados a partir das propriedades da mensagem, pois eles substituem o conjunto de valores para o destino. `format` não é atualizado. O exemplo não faz verificação de erros. Se o construtor `Record(BytesMessage)` for chamado, será presumido que o `JMSBytesMessage` é um tipo de mensagem RECORD. A linha "`setStructID(new String(structID, getMessageCharset()))`" configura o destaque.

b. As linhas de códigos que concluem os campos do método deserialize na mensagem, em ordem, atualizando os valores padrão configurados na instância RECORD.

4. Crie um método put para gravar os campos de cabeçalho em um JMSBytesMessage.

```
protected BytesMessage put(MyProducer myProducer) throws IOException,
        JMSException, UnsupportedEncodingException {
    setHeaderEncoding(myProducer.getEncoding());
    setHeaderCharset(myProducer.getCharset());
    myProducer.setMQClient(true);
    BytesMessage bytes = myProducer.session.createBytesMessage();
    bytes.setStringProperty(WMQConstants.JMS_IBM_FORMAT, getHeaderFormat());
    bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getHeaderEncoding());
    bytes.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET,
                         myProducer.getCCSID());
    bytes.writeBytes(String.format("%1$-" + RECORD_STRUCT_ID_LENGTH + "."
                    + RECORD_STRUCT_ID_LENGTH + "s", getStructID())
                    .getBytes(getMessageCharset()), 0, RECORD_STRUCT_ID_LENGTH);
    bytes.writeInt(getVersion());
    bytes.writeInt(getStructLength());
    return bytes;
}
```

Nota:

a. MyProducer encapsula o JMS Connection, Destination, Session, e MessageProducer em uma única classe MyConsumer, usado posteriormente, encapsula o JMS Connection, Destination, Sessione MessageConsumer em uma única classe..

b. Para um JMSBytesMessage, se a codificação for diferente de Native, a codificação deverá ser configurada na mensagem. A codificação de destino é copiada para o atributo de codificação de mensagem, JMS_IBM_CHARACTER_SET e salvo como um atributo da classe RECORD.

i) "setMessageEncoding(myProducer.getEncoding());" chama "(((MQDestination) destination).getIntProperty(WMQConstants.WMQ_ENCODING));" para obter a codificação de destino..

ii) O "Bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getMessageEncoding());" configura a codificação da mensagem

c. O conjunto de caracteres usado para transformar o texto em bytes é obtido a partir do destino e salvo como um atributo da classe RECORD. Ele não é configurado na mensagem, porque não é usado pelas classes IBM WebSphere MQ para JMS ao gravar um JMSBytesMessage

"messageCharset = myProducer.getCharset();" chamadas

```
public String getCharset() throws UnsupportedEncodingException,
        JMSException {
    return CCSID.getCodepage(getCCSID());
}
```

Ele obtém o conjunto de caractere Java de um identificador de conjunto de caracteres codificados.

" CCSID.getCodepage(ccsid) " está no pacote com.ibm.mq.headers. O ccsid é obtido a partir de um outro método no MyProducer, que consulta o destino:

```
public int getCCSID() throws JMSException {
    return (((MQDestination) destination)
            .getIntProperty(WMQConstants.WMQ_CCSID));
}
```

d. O "myProducer.setMQClient(true);" substitui a configuração de destino para o tipo de cliente, forçando-o a um cliente MQI do IBM WebSphere MQ Você pode preferir omitir esta linha de código, conforme ela oculta um erro de configuração administrativa.

"myProducer.setMQClient(true);" chama:

```
((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ); }
if (!getMQDest()) setMQBody();
```

O código tem o efeito colateral de configurar o estilo de corpo IBM WebSphere MQ para não especificado, se ele precisar substituir uma configuração de JMS.

Nota:

As classes IBM WebSphere MQ para JMS gravam o formato, a codificação e o identificador do conjunto de caractere da mensagem no descritor de mensagens, MQMDou no cabeçalho JMS, MQRFH2. Depende se a mensagem tem um corpo de estilo do IBM WebSphere MQ. Não configure os campos MQMD manualmente.

Um método existe para configurar as propriedades do descritor de mensagens manualmente. Ele usa as propriedades JMS_IBM_MQMD_*. Deve-se configurar a propriedade de destino, WMQ_MQMD_WRITE_ENABLED para configurar as propriedades JMS_IBM_MQMD_*:

```
((MQDestination)destination).setMQMDWriteEnabled(true);
```

Deve-se configurar a propriedade de destino WMQ_MQMD_READ_ENABLED para ler as propriedades.

Use o JMS_IBM_MQMD_* somente se você tomar o controle total da carga útil da mensagem. Diferente das propriedades JMS_IBM_* , as propriedades JMS_IBM_MQMD_* não controlam como classes IBM WebSphere MQ para JMS constroem uma mensagem JMS. É possível criar as propriedades do descritor de mensagens que entram em conflito com as propriedades da mensagem JMS

e. As linhas de códigos que concluem o método serializam os atributos na classe como campos na mensagem.

Os atributos de sequência são preenchidos com espaços em branco. As sequências são convertidas em bytes usando o conjunto de caracteres definido para o registro e truncado para o comprimento dos campos de mensagem.

5. Conclua a classe incluindo as importações.

```
package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;
```

6. Crie uma classe para estender a classe RECORD para incluir campos adicionais. Incluir um construtor padrão.

```
public class MyRecord extends RECORD {
    private static final long serialVersionUID = -370551723162299429L;
    private final static int FLAGS = 1;
    private final static String STRUCT_ID = "MYRD";
    private final static int DATA_LENGTH = 32;
    private final static String FORMAT = "MYRECORD";
    private int flags = FLAGS;
    private String recordData = "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345";

    public MyRecord() {
        super();
        super.setStructID(STRUCT_ID);
        super.setHeaderFormat(FORMAT);
        super.setStructLength(super.getStructLength() + MQLONG_LENGTH
                + DATA_LENGTH);
    }
```

Nota:

a. A subclasse RECORD, MyRecord, customiza o destaque, formato e comprimento do cabeçalho.

7. Crie ou gere os getters e setters.

a) Crie os getters:

```
    public int getFlags() { return flags; }
    public String getRecordData() { return recordData; }Â .
```

b) Crie os setters:

```
    public void setFlags(int flags) {
        this.flags = flags; }
    public void setRecordData(String recordData) {
        this.recordData = recordData; }
}
```

8. Crie um construtor para criar uma instância MyRecord a partir de um JMSBytesMessage.

```
    public MyRecord(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super(message);
        setFlags(message.readInt());
        byte[] recordData = new byte[DATA_LENGTH];
        message.readBytes(recordData, DATA_LENGTH);
        setRecordData(new String(recordData, super.getMessageCharset()));
    }
```

Nota:

a. Os campos que formam o modelo de mensagem padrão são lidos primeiro pela classe RECORD.

b. O texto recordData será convertido em String usando a propriedade do conjunto de caracteres da mensagem.

9. Crie um método estático para obter uma mensagem a partir de um consumidor e crie uma nova instância MyRecord.

```
    public static MyRecord get(MyConsumer myConsumer) throws JMSException,
            MQDataException, IOException {
        BytesMessage message = (BytesMessage) myConsumer.receive();
        return new MyRecord(message);
    }
```

Nota:

a. No exemplo, para abreviar, o construtor MyRecord(BytesMessage) é chamado a partir do método get estático. Geralmente, você pode separar o recebimento da mensagem a partir da criação de uma nova instância MyRecord.

10. Crie um método put para anexar os campos do cliente a uma JMSBytesMessage que contém um cabeçalho da mensagem.

```
    public BytesMessage put(MyProducer myProducer) throws JMSException,
            IOException {
        BytesMessage bytes = super.put(myProducer);
        bytes.writeInt(getFlags());
        bytes.writeBytes(String.format("%1$-" + DATA_LENGTH + "."
                        + DATA_LENGTH + "s",getRecordData())
                        .getBytes(super.getMessageCharset()), 0, DATA_LENGTH);
        myProducer.send(bytes);
        return bytes;
    }
```

Nota:

a. As chamadas de método no código serializam os atributos na classe MyRecord como campos na mensagem.

- O atributo recordData String é preenchido com espaços em branco, convertidos em bytes usando o conjunto de caracteres definido para o registro e truncado para o comprimento dos campos RecordData.

11. Conclua a classe, incluindo as instruções de inclusão.

```
package com.ibm.mq.id;
import java.io.IOException;
import javax.jms.BytesMessage;
```

```
import javax.jms.JMSException;
import com.ibm.mq.headers.MQDataException;
```

Resultados

Resultados:

- Os resultados da execução da classe `TryMyRecord`:
  - Enviando mensagem no conjunto de caracteres codificados 37 e usando uma saída de conversão do gerenciador de filas:

```
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
In  flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 273 CCSID UTF-8
```

  - Enviando mensagem no conjunto de caracteres codificados 37 e *não* usando uma saída de conversão do gerenciador de filas:

```
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
In  flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID IBM037
```

- Os resultados da modificação da classe `TryMyRecord` para não receber a mensagem, e em vez do recebimento, usando a amostra `amqsget0.c` modificada. A amostra modificada aceita um registro formatado; consulte Figura 156 na página 863 no “Trocando um registro formatado com um aplicativo não JMS” na página 860.
  - Enviando mensagem no conjunto de caracteres codificados 37 e usando uma saída de conversão do gerenciador de filas:

```
Sample AMQSGET0 start
ccsid <850>, flags <1>, message <ABCDEFGHIJKLMNOPQRSTUVWXYZ012345>
no more messages
Sample AMQSGET0 end
```

  - Enviando mensagem no conjunto de caracteres codificados 37 e *não* usando uma saída de conversão do gerenciador de filas:

```
Sample AMQSGET0 start
MQGET ended with reason code 2110
ccsid <37>, flags <1>, message <--+-+ãÃ++ÐÊËÈiÐÎÐ+ÔÒõõµþÞÚ-±=¾¶§>
no more messages
Sample AMQSGET0 end
```

Para tentar o exemplo e experimento com diferentes páginas de códigos e uma saída de conversão de dados. Crie as classes Java, configure IBM WebSphere MQe execute o programa principal `TryMyRecord`; consulte Figura 157 na página 871.

1. Configure IBM WebSphere MQ e JMS para executar o exemplo. As instruções são para executar o exemplo no Windows.

   1. Criar um gerenciador de filas

```
crtmqm -sa -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1
```

   2. Criar uma fila

```
echo DEFINE QL('Q1') REPLACE | runmqsc QM1
```

   3. Criar um diretório JNDI

```
cd c:\
md JNDI-Directory
```

4. **Alternar para o diretório bin do JMS**
   O programa JMS Administration deve ser executado a partir daqui. O caminho é *MQ_INSTALLATION_PATH*\java\bin.
5. **Crie as seguintes definições de JMS em um arquivo chamado `JMSQM1Q1.txt`**

```
DEF CF(QM1) PROVIDERVERSION(7) QMANAGER(QM1)
DEF Q(Q1) CCSID(37) ENCODING(RRR) MSGBODY(MQ) QMANAGER(QM1) QUEUE(Q1) TARGCLIENT(MQ)
VERSION(7)
END
```

6. **Execute o programa JMSAdmin para criar os recursos JMS**

```
JMSAdmin < JMSQM1Q1.txt
```

2. É possível criar, alterar e procurar as definições criadas usando o IBM WebSphere MQ Explorer.
3. Execute o `TryMyRecord`.

Classes usadas para executar o exemplo

As classes listadas nas figuras de Figura 157 na página 871 a Figura 162 na página 875 também estão disponíveis em um arquivo compactado; faça download de jm25529_.zip ou de jm25529_.tar.gz.

```
package com.ibm.mq.id;
public class TryMyRecord {
    public static void main(String[] args) throws Exception {
        MyProducer producer = new MyProducer();
        MyRecord outrec = new MyRecord();
        System.out.println("Out flags " + outrec.getFlags() + " text "
                + outrec.getRecordData() + " Encoding "
                + producer.getEncoding() + " CCSID " + producer.getCCSID()
                + " MQ " + producer.getMQDest());
        outrec.put(producer);
        System.out.println("Out flags " + outrec.getFlags() + " text "
                + outrec.getRecordData() + " Encoding "
                + producer.getEncoding() + " CCSID " + producer.getCCSID()
                + " MQ " + producer.getMQDest());
        MyRecord inrec = MyRecord.get(new MyConsumer());
        System.out.println("In  flags " + inrec.getFlags() + " text "
                + inrec.getRecordData() + " Encoding "
                + inrec.getMessageEncoding() + " CCSID "
                + inrec.getMessageCharset());
    }
}
```

Figura 157. TryMyRecord

```
package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;

public abstract class RECORD implements Serializable {
    private static final long serialVersionUID = -1616617232750561712L;
    protected final static int UTF8 = 1208;
    protected final static int MQLONG_LENGTH = 4;
    protected final static int RECORD_STRUCT_ID_LENGTH = 4;
    protected final static int RECORD_VERSION_1 = 1;
    protected final String RECORD_STRUCT_ID = "BLNK";
    protected final String RECORD_TYPE = "BLANK   ";
    private String structID = RECORD_STRUCT_ID;
    private int version = RECORD_VERSION_1;
    private int structLength = RECORD_STRUCT_ID_LENGTH + MQLONG_LENGTH * 2;
    private int headerEncoding = WMQConstants.WMQ_ENCODING_NATIVE;
    private String headerCharset = "UTF-8";
    private String headerFormat = RECORD_TYPE;

    public RECORD() {
        super();
    }

    public RECORD(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super();
        setHeaderCharset(message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET));
        setHeaderEncoding(message.getIntProperty(WMQConstants.JMS_IBM_ENCODING));
        byte[] structID = new byte[RECORD_STRUCT_ID_LENGTH];
        message.readBytes(structID, RECORD_STRUCT_ID_LENGTH);
        setStructID(new String(structID, getMessageCharset()));
        setVersion(message.readInt());
        setStructLength(message.readInt());
    }

    public String getHeaderFormat() { return headerFormat; }
    public int getHeaderEncoding() { return headerEncoding;    }
    public String getMessageCharset() { return headerCharset; }
    public int getMessageEncoding() { return headerEncoding; }
    public String getStructID() { return structID; }
    public int getStructLength() { return structLength; }
    public int getVersion() { return version; }


    protected BytesMessage put(MyProducer myProducer) throws IOException,
            JMSException, UnsupportedEncodingException {
        setHeaderEncoding(myProducer.getEncoding());
        setHeaderCharset(myProducer.getCharset());
        myProducer.setMQClient(true);
        BytesMessage bytes = myProducer.session.createBytesMessage();
        bytes.setStringProperty(WMQConstants.JMS_IBM_FORMAT, getHeaderFormat());
        bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getHeaderEncoding());
        bytes.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET,
                             myProducer.getCCSID());
        bytes.writeBytes(String.format("%1$-" + RECORD_STRUCT_ID_LENGTH + "."
                         + RECORD_STRUCT_ID_LENGTH + "s", getStructID())
                         .getBytes(getMessageCharset()), 0, RECORD_STRUCT_ID_LENGTH);
        bytes.writeInt(getVersion());
        bytes.writeInt(getStructLength());
        return bytes;
    }

    public void setHeaderCharset(String charset) {
        this.headerCharset = charset; }
    public void setHeaderEncoding(int encoding) {
        this.headerEncoding = encoding; }
    public void setHeaderFormat(String headerFormat) {
        this.headerFormat = headerFormat; }
    public void setStructID(String structID) {
        this.structID = structID; }
    public void setStructLength(int structLength) {
        this.structLength = structLength; }
    public void setVersion(int version) {
        this.version = version; }
}
```

Figura 158. RECORD

```
package com.ibm.mq.id;
import java.io.IOException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.headers.MQDataException;

public class MyRecord extends RECORD {
    private static final long serialVersionUID = -370551723162299429L;
    private final static int FLAGS = 1;
    private final static String STRUCT_ID = "MYRD";
    private final static int DATA_LENGTH = 32;
    private final static String FORMAT = "MYRECORD";
    private int flags = FLAGS;
    private String recordData = "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345";

    public MyRecord() {
        super();
        super.setStructID(STRUCT_ID);
        super.setHeaderFormat(FORMAT);
        super.setStructLength(super.getStructLength() + MQLONG_LENGTH
                + DATA_LENGTH);
    }

    public MyRecord(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super(message);
        setFlags(message.readInt());
        byte[] recordData = new byte[DATA_LENGTH];
        message.readBytes(recordData, DATA_LENGTH);
        setRecordData(new String(recordData, super.getMessageCharset()));
    }

    public static MyRecord get(MyConsumer myConsumer) throws JMSException,
            MQDataException, IOException {
        BytesMessage message = (BytesMessage) myConsumer.receive();
        return new MyRecord(message);
    }

    public int getFlags() { return flags; }
    public String getRecordData() { return recordData; }Â .


    public BytesMessage put(MyProducer myProducer) throws JMSException,
            IOException {
        BytesMessage bytes = super.put(myProducer);
        bytes.writeInt(getFlags());
        bytes.writeBytes(String.format("%1$-" + DATA_LENGTH + "."
                        + DATA_LENGTH + "s",getRecordData())
                        .getBytes(super.getMessageCharset()), 0, DATA_LENGTH);
        myProducer.send(bytes);
        return bytes;
    }

    public void setFlags(int flags) {
        this.flags = flags; }
    public void setRecordData(String recordData) {
        this.recordData = recordData; }
}
```

Figura 159. MyRecord

```
package com.ibm.mq.id;
import java.io.UnsupportedEncodingException;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.Session;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import com.ibm.mq.headers.CCSID;
import com.ibm.mq.jms.MQDestination;
import com.ibm.msg.client.wmq.WMQConstants;
public abstract class EndPoint {
    public Context ctx;
    public ConnectionFactory cf;
    public Connection connection;
    public Destination destination;
    public Session session;
    protected EndPoint() throws NamingException, JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup("QM1");
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup("Q1");
        ((MQDestination)destination).setReceiveConversion
                                    (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); }
    protected EndPoint(String cFactory, String dest) throws NamingException,
            JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup(cFactory);
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup(dest);
        ((MQDestination)destination).setReceiveConversion
                                    (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);  }
    public int getCCSID() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_CCSID));  }
    public String getCharset() throws UnsupportedEncodingException,
            JMSException {
        return CCSID.getCodepage(getCCSID());  }
    public int getEncoding() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_ENCODING));  }
    public boolean getMQDest() throws JMSException {
        if ((((MQDestination) destination).getMessageBodyStyle()
               == WMQConstants.WMQ_MESSAGE_BODY_MQ)
            || ((((MQDestination) destination).getMessageBodyStyle()
                  == WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED)
               && (((MQDestination) destination).getTargetClient()
                   == WMQConstants.WMQ_TARGET_DEST_MQ)))
            return true;
        else
            return false; }
    public void setCCSID(int ccsid) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_CCSID,
                ccsid); }
    public void setEncoding(int encoding) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_ENCODING,
                encoding); }
    public void setMQBody() throws JMSException {
        ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);  }
    public void setMQBody(boolean mqbody) throws JMSException {
        if (mqbody)  ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
        else         ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_JMS);  }
    public void setMQClient(boolean mqclient) throws JMSException {
       if (mqclient){
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
            if (!getMQDest()) setMQBody();
       }
       else
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_JMS); }
    }
```

Figura 160. EndPoint

```
package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageProducer;
import javax.naming.NamingException;
public class MyProducer extends EndPoint {
    public MessageProducer producer;
    public MyProducer() throws NamingException, JMSException {
        super();
        producer = session.createProducer(destination); }
    public MyProducer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
        producer = session.createProducer(destination); }
    public void send(Message message) throws JMSException {
        producer.send(message); }
}
```

Figura 161. MyProducer

```
package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.naming.NamingException;
public class MyConsumer extends EndPoint {
    public MessageConsumer consumer;
    public MyConsumer() throws NamingException, JMSException {
        super();
        consumer = session.createConsumer(destination);
        connection.start();  }
    public MyConsumer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
        consumer = session.createConsumer(destination);
        connection.start();  }
    public Message receive() throws JMSException {
        return consumer.receive();  }
}
```

Figura 162. MyConsumer

Criando e configurando connection factories e destinos em uma classe WebSphere MQ para o aplicativo JMS

Uma classe WebSphere MQ para o aplicativo JMS pode criar connection factories e destinos recuperando-os como objetos administrados de um namespace Java Naming and Directory Interface (JNDI), usando as extensões JMS IBM ou usando as extensões JMS WebSphere MQ . Um aplicativo também pode usar as extensões JMS do IBM ou WebSphere MQ para configurar as propriedades de connection factories e destinos.

Connection factories e destinos são pontos de início no fluxo de lógica de um aplicativo JMS. Um aplicativo usa um objeto ConnectionFactory para criar uma conexão com um servidor de sistema de mensagens, e usa um objeto Fila ou Tópico como um destino para enviar mensagens para ou uma origem da qual receber mensagens. Um aplicativo, portanto, precisa criar pelo menos um connection factory e um ou mais destinos. Ter criado uma connection factory ou destino, o aplicativo poderá então precisar configurar o objeto, configurando uma ou mais de suas propriedades.

Em resumo, um aplicativo pode criar e configurar connection factories e destinos das seguintes maneiras:

Usando JNDI para recuperar objetos administrados
Um administrador pode usar a ferramenta de administração JMS do WebSphere MQ ou WebSphere MQ Explorer para criar e configurar connection factories e destinos como objetos administrados em um namespace JNDI. Um aplicativo pode então recuperar os objetos administrados do espaço de nomes JNDI. Tendo recuperado um objeto administrado, o aplicativo pode, se necessário, configurar ou alterar uma ou mais de suas propriedades usando as extensões JMS do IBM ou as extensões JMS do WebSphere MQ .

Usando as extensões JMS do IBM
Um aplicativo pode usar as extensões JMS IBM para criar connection factories e destinos dinamicamente no tempo de execução. O aplicativo cria primeiramente um objeto JmsFactoryFactory

e, em seguida, usa métodos desse objeto para criar connection factories e destinos. Depois de criar um connection factory ou o destino, o aplicativo pode usar métodos herdado da interface JmsPropertyContext para configurar suas propriedades. Como alternativa, o aplicativo pode utilizar um URI (Uniform Resource Identifier) para especificar uma ou mais propriedades de um destino, quando ele cria o destino.

Usando as extensões JMS do WebSphere MQ
Um aplicativo também pode usar as extensões JMS do WebSphere MQ para criar connection factories e destinos dinamicamente no tempo de execução.. O aplicativo usa os construtores fornecidos para criar connection factories e destinos. Depois de criado um connection factory ou destino, o aplicativo pode usar os métodos do objeto para configurar suas propriedades. Como alternativa, o aplicativo pode utilizar um URI para especificar uma ou mais propriedades de um destino, quando ele cria o destino.

Usando JNDI para Recuperar Objetos Administrados em um Aplicativo JMS

Para recuperar objetos administrados de um namespace Java Naming and Directory Interface (JNDI), um aplicativo JMS deve criar um contexto inicial e, em seguida, usar o método lookup () para recuperar os objetos.

Antes de um aplicativo poder recuperar objetos administrados a partir de um namespace do JNDI, um administrador deverá primeiro criar os objetos administrados. O administrador pode usar a ferramenta de administração JMS do WebSphere MQ ou o Explorer do WebSphere MQ para criar e manter objetos administrados em um namespace JNDI. Para obter informações sobre como usar a ferramenta de administração JMS WebSphere MQ , consulte “Usando a ferramenta de administração JMS do WebSphere MQ” na página 946. Para obter informações sobre como usar o WebSphere MQ Explorer, consulte a ajuda fornecida com o WebSphere MQ Explorer. No entanto, um servidor de aplicativos geralmente fornece seu próprio repositório para objetos administrados e suas próprias ferramentas para criar e manter os objetos.

Para recuperar objetos administrados de um namespace do JNDI, um aplicativo deve criar primeiro um contexto inicial, conforme mostrado no exemplo a seguir:

```
import javax.jms.*;
import javax.naming.*;
import javax.naming.directory.*;
.
.
.
String url = "ldap://server.company.com/o=company_us,c=us";
String icf = "com.sun.jndi.ldap.LdapCtxFactory";
.
java.util.Hashtable environment = new java.util.Hashtable();
environment.put(Context.PROVIDER_URL, url);
environment.put(Context.INITIAL_CONTEXT_FACTORY, icf);
Context ctx = new InitialDirContext(environment);
```

Nesse código, as variáveis String `url` e `icf` possuem os seguintes significados:

url
O localizador uniforme de recursos (URL) do serviço de diretório. A URL pode ter um dos formatos a seguir:

- `ldap://`*`hostname`*`/`*`contextName`*, para um serviço de diretório com base em um servidor LDAP
- `file:/`*`directoryPath`*, para um serviço de diretório com base no sistema de arquivos local

icf
O nome da classe da factory de contexto inicial, que pode ser um dos seguintes valores:

- `com.sun.jndi.ldap.LdapCtxFactory`, para um serviço de diretório com base em um servidor LDAP
- `com.sun.jndi.fscontext.RefFSContextFactory`, para um serviço de diretório com base no sistema de arquivos local

Observe que algumas combinações de um pacote JNDI e um provedor de serviços Lightweight Directory Access Protocol (LDAP) podem causar o erro LDAP 84 para ocorrer. Para resolver esse problema, insira a seguinte linha de código antes da chamada para InitialDirContext():

```
environment.put(Context.REFERRAL, "throw");
```

Após um contexto inicial ser obtido, o aplicativo poderá recuperar objetos administrados a partir do namespace do JNDI, usando o método lookup(), conforme mostrado no exemplo a seguir:

```
ConnectionFactory factory;
Queue queue;
Topic topic;
.
.
.
factory = (ConnectionFactory)ctx.lookup("cn=myCF");
queue = (Queue)ctx.lookup("cn=myQ");
topic = (Topic)ctx.lookup("cn=myT");
```

Esse código recupera os seguintes objetos a partir de um namespace base do LDAP:

- Um limite de objeto ConnectionFactory com o nome myCF
- Um limite de objeto Queue com o nome myQ
- Um limite de objeto Topic com o nome myT

Usando as extensões JMS do IBM

WebSphere MQ classes para JMS contém um conjunto de extensões para a API JMS chamado IBM extensões JMS. Um aplicativo pode utilizar essas extensões para criar connection factories e destinos dinamicamente no tempo de execução e para configurar as propriedades das classes WebSphere MQ para objetos JMS. As extensões podem ser usadas com qualquer provedor de sistema de mensagens.

As extensões do JMS do IBM são um conjunto de interfaces e classes nos seguintes pacotes:

- com.ibm.msg.client.jms
- com.ibm.msg.client.services

Os pacotes podem ser localizados em `com.ibm.mqjms.jar`, localizado em `<MQ_Install_Dir>/java/lib`.

Essas extensões fornecem a seguinte função:

- Um mecanismo baseado em factory para criar connection factories e destinos dinamicamente no tempo de execução, em vez de recuperá-los como objetos administrados a partir de um namespace Java Naming and Directory Interface (JNDI)
- Um conjunto de métodos para configurar as propriedades de classes do WebSphere MQ para objetos JMS
- Um conjunto de classes de exceção com métodos para obter informações detalhadas sobre um problema
- Um conjunto de métodos para controlar o rastreio
- Um conjunto de métodos para obter informações de versão sobre classes do WebSphere MQ para JMS

Com relação à criação dinâmica de connection factories e destinos no tempo de execução e à configuração e obtenção de suas propriedades, as extensões JMS do IBM fornecem um conjunto alternativo de interfaces para as extensões JMS do WebSphere MQ . No entanto, considerando que as extensões JMS do WebSphere MQ são específicas para o provedor de sistemas de mensagens do WebSphere MQ , as extensões JMS do IBM não são específicas para o WebSphere MQ e podem ser usadas com qualquer provedor de sistemas de mensagens dentro da arquitetura em camadas descrita em “Uma arquitetura em camadas” na página 809.

A interface com.ibm.msg.client.wmq.WMQConstants contém as definições de constantes, que um aplicativo pode usar ao configurar as propriedades das classes WebSphere MQ para objetos JMS usando as extensões JMS do IBM . A interface contém constantes para o provedor de sistemas de

mensagens WebSphere MQ e constantes JMS que são independentes de qualquer provedor de sistemas de mensagens.

Os exemplos de código a seguir supõem que as instruções de importação a seguir foram executadas:

```
import com.ibm.msg.client.jms.*;
import com.ibm.msg.client.services.*;
import com.ibm.msg.client.wmq.WMQConstants;
```

Criando connection factories e destinos

Antes que um aplicativo possa criar connection factories e destinos usando as extensões JMS do IBM , ele deve primeiro criar um objeto de Factory JmsFactory.. Para criar um objeto JmsFactoryFactory, o aplicativo chama o método getInstance() da classe JmsFactoryFactory, conforme mostrado no exemplo a seguir:

```
JmsFactoryFactory ff = JmsFactoryFactory.getInstance(JmsConstants.WMQ_PROVIDER);
```

O parâmetro na chamada getInstance() é uma constante que identifica o provedor de sistemas de mensagens WebSphere MQ como o provedor de sistemas de mensagens escolhido. O aplicativo pode então usar o objeto JmsFactoryFactory para criar connection factories e destinos.

Para criar um connection factory, o aplicativo chama o método createConnectionFactory() do objeto JmsFactoryFactory, conforme mostrado no exemplo a seguir:

```
JmsConnectionFactory factory = ff.createConnectionFactory();
```

Esta instrução cria um objeto JmsConnectionFactory com os valores padrão para todas as suas propriedades, o que significa que o aplicativo se conecta ao gerenciador de filas padrão no modo de ligações. Se desejar que um aplicativo se conecte no modo cliente ou se conecte a um gerenciador de filas diferente do gerenciador de filas padrão, o aplicativo deve configurar as propriedades apropriadas do objeto JmsConnectionFactory antes de criar a conexão. Para obter informações sobre como fazer isso, consulte “Configurando as Propriedades de Classes WebSphere MQ para Objetos JMS” na página 879.

A classe JmsFactoryFactory também contém métodos para criar connection factories dos tipos a seguir:

- JmsQueueConnectionFactory
- JmsTopicConnectionFactory
- JmsXAConnectionFactory
- JmsXAQueueConnectionFactory
- JmsXATopicConnectionFactory

Para criar um objeto Queue, o aplicativo chama o método createQueue() do objeto JmsFactoryFactory, conforme mostrado no exemplo a seguir:

```
JmsQueue q1 = ff.createQueue("Q1");
```

Esta instrução cria um objeto JmsQueue com os valores padrão para todas as suas propriedades. O objeto representa uma fila do WebSphere MQ chamada Q1 que pertence ao gerenciador de fila local.. Essa fila pode ser uma fila local, uma fila de alias ou uma definição de fila remota.

O método createQueue() também pode aceitar um identificador uniforme de recursos (URI) da fila como um parâmetro. Um URI de fila é uma cadeia que especifica o nome de uma fila do WebSphere MQ e, opcionalmente o nome do gerenciador de filas que possui a fila e uma ou mais propriedades do objeto JmsQueue . A instrução a seguir contém um exemplo de um URI de fila:

```
JmsQueue q2 = ff.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

O objeto JmsQueue criado por essa instrução representa uma fila do WebSphere MQ chamada Q2 que pertence ao gerenciador de filas QM2e todas as mensagens enviadas para esse destino são persistentes e têm uma prioridade de 5 Para obter mais informações sobre URIs de filas, consulte “Identificadores uniformes de recursos (URIs)” na página 891. Para um modo alternativo para configurar propriedades

de um objeto JmsQueue, consulte “Configurando as Propriedades de Classes WebSphere MQ para Objetos JMS” na página 879.

Para criar um objeto Topic, um aplicativo pode usar o método createTopic() do objeto JmsFactoryFactory, conforme mostrado no exemplo a seguir:

```
JmsTopic t1 = ff.createTopic("Sport/Football/Results");
```

Esta instrução cria um objeto JmsTopic com os valores padrão para todas as suas propriedades. O objeto representa um tópico chamado Sport/Football/Results.

O método createTopic() também pode aceitar um URI de tópico como um parâmetro. Um URI de tópico é uma sequência que especifica o nome de um tópico e, como opção, uma ou mais propriedades do objeto JmsTopic. As instruções a seguir contêm um exemplo de um URI de tópico:

```
String s1 = "topic://Sport/Tennis/Results?persistence=1&priority=0";
JmsTopic t2 = ff.createTopic(s1);
```

O objeto JmsTopic criado por essas instruções representa um tópico chamado Esporte/Tênis/Resultados, e todas as mensagens enviadas para este destino são não persistentes e têm prioridade de 0. Para obter mais informações sobre URIs de tópico de, consulte “Identificadores uniformes de recursos (URIs)” na página 891 Para um modo alternativo para configurar propriedades de um objeto JmsTopic, consulte “Configurando as Propriedades de Classes WebSphere MQ para Objetos JMS” na página 879.

Após um aplicativo ter criado um connection factory ou destino, esse objeto pode ser usado somente com o provedor de sistema de mensagens selecionado.

Configurando as Propriedades de Classes WebSphere MQ para Objetos JMS

Para configurar as propriedades de classes WebSphere MQ para objetos JMS usando as extensões JMS do IBM , um aplicativo usa os métodos da interface com.ibm.msg.client.JmsPropertyContext .

Para cada tipo de dados Java, a interface de contexto JmsPropertycontém um método para configurar o valor de uma propriedade com esse tipo de dados e um método para obter o valor de uma propriedade com esse tipo de dados. Por exemplo, um aplicativo chama o método setIntProperty() para configurar uma propriedade com um valor de número inteiro e chama o método getIntProperty() para obter uma propriedade com um valor de número inteiro.

As instâncias de classes no pacote com.ibm.mq.jms também herdam os métodos da interface JmsPropertyContext. Um aplicativo pode, portanto, usar esses métodos para configurar as propriedades dos objetos MQConnectionFactory, MQQueue e MQTopic.

Quando um aplicativo cria um WebSphere MQ classes para objeto JMS, quaisquer propriedades com valores padrão são configuradas automaticamente Quando um aplicativo configura uma propriedade, o novo valor substitui qualquer valor anterior da propriedade. Após uma propriedade ser configurada, ela não pode ser excluída, mas seu valor poderá ser mudado.

Se um aplicativo tentar configurar uma propriedade para um valor que não seja válido para a propriedade, as classes do WebSphere MQ para JMS emitirá uma exceção JMSException. Se um aplicativo tentar obter uma propriedade não configurada, o comportamento será conforme descrito na especificação JMS. WebSphere MQ classes para JMS lança uma exceção de exceção NumberFormatpara tipos de dados primitivos e retorna nulo para tipos de dados referenciados.

Além das propriedades predefinidas de classes WebSphere MQ para objeto JMS, um aplicativo pode configurar suas próprias propriedades. Essas propriedades definidas pelo aplicativo são ignorados por classes do WebSphere MQ para JMS

Para obter mais informações sobre as propriedades das classes do WebSphere MQ para objetos JMS, consulte Propriedades de IBM WebSphere MQ classes for JMS objetos

O código a seguir é um exemplo de como configurar propriedades usando as extensões JMS do IBM . O código configura cinco propriedades de um connection factory.

```
factory.setIntProperty(WMQConstants.WMQ_CONNECTION_MODE,
                       WMQConstants.WMQ_CM_CLIENT);
```

```
factory.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "QM1");
factory.setStringProperty(WMQConstants.WMQ_HOST_NAME, "HOST1");
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
factory.setStringProperty(WMQConstants.WMQ_CHANNEL, "QM1.SVR");
factory.setStringProperty(WMQConstants.WMQ_APPLICATIONNAME, "My Application");
```

O efeito de configurar estas propriedades é que o aplicativo se conecta ao gerenciador de filas QM1 no modo cliente, usando um canal MQI chamado QM1.SVR. O gerenciador de filas está em execução em um sistema com nome do host HOST1 e o listener para o gerenciador de filas está atendendo no número da porta 1415. Esta conexão e outras conexões do gerenciador de filas associadas a outras sessões sob ela têm o nome do aplicativo “Meu aplicativo" associado a elas.

Nota: Os gerenciadores de filas em execução nas plataformas z/OS não suportam a configuração de nomes de aplicativos e essa configuração é, portanto, ignorados

A interface JmsPropertyContext também contém o método setObjectProperty(), que um aplicativo pode usar para configurar as propriedades. O segundo parâmetro do método é um objeto que contém o valor da propriedade. Por exemplo, o código a seguir cria um objeto de Número inteiro que contém o número inteiro 1415 e, em seguida, chama setObjectProperty() para configurar a propriedade PORT de um connection factory para o valor 1415:

```
Integer port = new Integer(1415);
factory.setObjectProperty(WMQConstants.WMQ_PORT, port);
```

Esse código é, portanto, equivalente à instrução a seguir:

```
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
```

Por outro lado, o método getObjectProperty() retorna um objeto que contém o valor de uma propriedade.

Conversão implícita de um valor de propriedade de um tipo de dados para outro

Quando um aplicativo usa um método da interface de contexto JmsPropertypara configurar ou obter a propriedade de classes do WebSphere MQ para objeto JMS, o valor da propriedade pode ser implicitamente convertido de um tipo de dados para outro.

Por exemplo, a instrução a seguir configura a propriedade PRIORITY do objeto JmsQueue q1:

```
q1.setStringProperty(WMQConstants.WMQ_PRIORITY, "5");
```

A propriedade PRIORITY tem um valor de número inteiro e, portanto, a chamada setStringProperty() converte implicitamente a sequência "5" (o valor de origem) para o número inteiro 5 (o valor de destino), que, então, se torna o valor da propriedade PRIORITY.

Por outro lado, a instrução a seguir obtém a propriedade PRIORITY do objeto JmsQueue q1:

```
String s1 = q1.getStringProperty(WMQConstants.WMQ_PRIORITY);
```

O número inteiro 5 (o valor de origem), que é o valor da propriedade PRIORITY, é convertido implicitamente para a sequência "5" (o valor de destino) pela chamada getStringProperty().

As conversões suportadas pelo WebSphere MQ classes para JMS são mostradas em Tabela 122 na página 880.

Tabela 122. Conversões suportadas de um tipo de dados para outro

Tipo de dados de origem	**Tipos de dados de destino suportados**
booleano	Sequência
byte	int, long, short, String
caractere	Sequência
duplo	Sequência

Tabela 122. Conversões suportadas de um tipo de dados para outro (continuação)	
Tipo de dados de origem	**Tipos de dados de destino suportados**
float	double, String
int	long, String
grande	Sequência
short	int, long, String
Sequência	boolean, byte, double, float, int, long, short

As regras gerais que regem as conversões suportadas são as seguintes:

- Valores numéricos podem ser convertidos de um tipo de dados para outro desde que nenhum dado seja perdido durante a conversão. Por exemplo, um valor com tipo de dados `int` pode ser convertido para um valor com tipo de dados `long`, mas não pode ser convertido para um valor com tipo de dados `short`.
- Um valor de qualquer tipo de dados pode ser convertido para uma sequência.
- Uma sequência pode ser convertida em um valor de qualquer outro tipo de dados (exceto `char`) desde que a sequência esteja no formato correto para a conversão. Se um aplicativo tentar converter uma sequência que não está no formato correto, as classes do WebSphere MQ para JMS emitirá uma exceção de exceção NumberFormat
- Se um aplicativo tentar uma conversão que não seja suportada, as classes WebSphere MQ para JMS emitirá uma exceção de exceção MessageFormat.

As regras específicas para converter um valor de um tipo de dados para outro são as seguintes:

- Ao converter um valor booleano para uma sequência, o valor `true` é convertido para a sequência "true" e o valor `false` para a sequência "false".
- Ao converter uma sequência para um valor booleano, a sequência "true" (sem distinção entre maiúsculas e minúsculas) é convertida para `true` e a sequência "false" (sem distinção entre maiúsculas e minúsculas) é convertido para `false`. Qualquer outra sequência é convertida para `false`.
- Ao converter uma sequência para um valor com tipo de dados `byte`, `int`, `long` ou `short`, a sequência deve ter o seguinte formato:

  [*espaços em branco*][*sinal*] *dígitos*

  Os significados dos componentes da sequência são os seguintes:

  espaços em branco
  Caracteres em branco à esquerda opcionais.

  sinal
  Um sinal de mais (+) ou de menos (-) opcional.

  dígitos
  Uma sequência contínua de dígitos (0 a 9). Pelo menos um dígito deve estar presente.

  Após a sequência de dígitos, a sequência pode conter outros caracteres que não são dígitos, mas a conversão para assim que o primeiro desses caracteres for atingido. A sequência é assumida para representar um número inteiro decimal.

  Se a sequência não estiver no formato correto, as classes do WebSphere MQ para JMS emitirá uma exceção de exceção NumberFormat.
- Ao converter uma sequência para um valor com tipo de dados `double` ou `float`, a sequência deve ter o formato a seguir:

  [*espaços em branco*] [*sign*]*dígitos*[*e_char*[*e_sign*]*e_digits*]

  Os significados dos componentes da sequência são os seguintes:

espaços em branco
Caracteres em branco à esquerda opcionais.

sinal
Um sinal de mais (+) ou de menos (-) opcional.

dígitos
Uma sequência contínua de dígitos (0 a 9). Pelo menos um dígito deve estar presente.

e_char
Um caractere expoente, que é um *E* ou *e*.

e_sign
Um sinal de mais (+) ou de menos (-) opcional para o expoente.

e_digits
Uma sequência contínua de dígitos (0 a 9) para o expoente. Pelo menos um dígito deve estar presente se a sequência contiver um caractere expoente.

Após a sequência de dígitos ou dos caracteres opcionais que representam um expoente, a sequência pode conter outros caracteres que não são dígitos, mas a conversão para assim que o primeiro desses caracteres for atingido. Supõe-se que a sequência represente um número de vírgula flutuante decimal com um expoente que é uma potência de 10.

Se a sequência não estiver no formato correto, as classes do WebSphere MQ para JMS emitirá uma exceção de exceção NumberFormat.

- Ao converter um valor numérico (incluindo um valor com tipo de dados `byte`) em uma sequência, o valor é convertido para a representação em sequência do valor como um número decimal, não a sequência contendo o caractere ASCII para esse valor. Por exemplo, o número inteiro 65 será convertido para a sequência "65", não para a sequência "A".

Configurando mais de uma propriedade em uma única chamada

A interface JmsPropertyContext também contém o método setBatchProperties(), que um aplicativo pode usar para configurar mais de uma propriedade em uma única chamada. O parâmetro do método é um objeto de Mapa que contém um conjunto de pares nome-valor de propriedades.

Por exemplo, o código a seguir usa o método setBatchProperties() para configurar as mesmas cinco propriedades de um connection factory conforme mostrado em “Configurando as Propriedades de Classes WebSphere MQ para Objetos JMS” na página 879. O código cria uma instância da classe HashMap, que implementa a interface de Map.

```
HashMap batchProperties = new HashMap();
batchProperties.put(WMQConstants.WMQ_CONNECTION_MODE,
                    new Integer(WMQConstants.WMQ_CM_CLIENT));
batchProperties.put(WMQConstants.WMQ_QUEUE_MANAGER, "QM1");
batchProperties.put(WMQConstants.WMQ_WMQ_HOST_NAME, "HOST1");
batchProperties.put(WMQConstants.WMQ_PORT, "1414");
batchProperties.put(WMQConstants.WMQ_CHANNEL, "QM1.SVR");
factory.setBatchProperties(batchProperties);
```

Observe que o segundo parâmetro do método Map.put() deve ser um objeto. Portanto, um valor de propriedade com um tipo de dados primitivo deve estar contido em um objeto ou ser representado por uma sequência conforme mostrado no exemplo.

O método setBatchProperties() valida cada propriedade. Se o método setBatchProperties() não puder configurar uma propriedade porque, por exemplo, seu valor não é válido, nenhuma das propriedades especificadas serão configuradas.

Nomes e valores de propriedade

Se um aplicativo usar os métodos da interface de contexto JmsPropertypara configurar e obter as propriedades das classes WebSphere MQ para objetos JMS, o aplicativo poderá especificar os nomes e valores de propriedades de qualquer uma das maneiras a seguir: Cada um dos exemplos fornecidos

mostra como configurar a propriedade PRIORITY do objeto JmsQueue q1 de forma que uma mensagem enviada para a fila tenha a prioridade especificada na chamada send().

Usando os nomes e valores de propriedades definidos como constantes na interface com.ibm.msg.client.wmq.WMQConstants

A instrução a seguir é um exemplo de como especificar os nomes e valores de propriedades dessa maneira:

```
q1.setIntProperty(WMQConstants.WMQ_PRIORITY, WMQConstants.WMQ_PRI_APP);
```

Usando os nomes e valores de propriedades que podem ser usados nos identificadores uniformes de recursos (URIs) da fila e do tópico

A instrução a seguir é um exemplo de como especificar os nomes e valores de propriedades dessa maneira:

```
q1.setIntProperty("priority", -2);
```

Somente os nomes e valores de propriedades de destinos podem ser especificados dessa maneira.

Usando os nomes e valores de propriedades que são reconhecidos pela ferramenta de administração JMS do WebSphere MQ

A instrução a seguir é um exemplo de como especificar os nomes e valores de propriedades dessa maneira:

```
q1.setStringProperty("PRIORITY", "APP");
```

A forma abreviada do nome da propriedade também é aceitável, conforme mostrado na instrução a seguir:

```
q1.setStringProperty("PRI", "APP");
```

Quando um aplicativo obtém uma propriedade, o valor retornado depende da maneira que o aplicativo especifica o nome da propriedade. Por exemplo, se um aplicativo especifica a constante WMQConstants.WMQ_PRIORITY como o nome da propriedade, o valor retornado é o número inteiro -2:

```
int n1 = getIntProperty(WMQConstants.WMQ_PRIORITY);
```

O mesmo valor será retornado se o aplicativo especificar a sequência “priority" como o nome da propriedade:

```
int n2 = getIntProperty("priority");
```

No entanto, se o aplicativo especificar a sequência "PRIORITY" ou "PRI" como o nome da propriedade, o valor retornado será a sequência "APP":

```
String s1 = getStringProperty("PRI");
```

Internamente, as classes do WebSphere MQ para JMS armazenam nomes de propriedades e valores como os valores literais definidos na interface com.ibm.msg.client.wmq.WMQConstants Esse é o formato canônico definido para os nomes e valores de propriedades. Como regra geral, se um aplicativo configurar propriedades usando uma das outras duas maneiras de especificar nomes e valores de propriedades, as classes WebSphere MQ para JMS devem converter os nomes e valores do formato de entrada especificado no formato canônico. Da mesma forma, se um aplicativo obtém propriedades usando uma das outras duas maneiras de especificar nomes e valores de propriedades, as classes do WebSphere MQ para JMS devem converter os nomes do formato de entrada especificado no formato canônico e converter os valores do formato canônico no formato de saída necessário. Precisar executar essas conversões pode ter implicações no desempenho.

Os nomes de propriedades e valores retornados por exceções, em arquivos de rastreamento ou nas classes do WebSphere MQ para o log JMS estão sempre no formato canônico

Usando a interface Map

A interface JmsPropertyContext estende a interface java.util.Map. Portanto, um aplicativo pode usar os métodos da interface de Mapa para acessar as propriedades de uma classe WebSphere MQ para objeto JMS.

Por exemplo, o código a seguir imprime os nomes e os valores de todas as propriedades de um connection factory. O código usa somente os métodos da interface Map para obter os nomes e os valores das propriedades.

```
// Get the names of all the properties
Set propNames = factory.keySet();

// Loop round all the property names and get the property values
Iterator iterator = propNames.iterator();
while (iterator.hasNext()){
    String pName = (String)iterator.next();
    System.out.println(pName+"="+factory.get(pName));
}
```

Usar os métodos da interface Map não ignora quaisquer validações ou conversões de propriedades.

Usando as extensões JMS do WebSphere MQ

As classes do WebSphere MQ para JMS contêm um conjunto de extensões para a API JMS chamada WebSphere MQ extensões JMS. Um aplicativo pode usar estas extensões para criar connection factories e destinos dinamicamente no tempo de execução, e para configurar suas propriedades.

WebSphere MQ classes para JMS contém um conjunto de classes nos pacotes com.ibm.jms e com.ibm.mq.jms. Essas classes implementam as interface JMS e contêm as extensões JMS do WebSphere MQ . Os exemplos de código a seguir supõem que estes pacotes tenham sido importados pelas seguintes instruções:

```
import com.ibm.jms.*;
import com.ibm.mq.jms.*;
```

Um aplicativo pode usar as extensões JMS do WebSphere MQ para executar as seguintes funções:

- Crie connection factories e destinos dinamicamente no tempo de execução, em vez de recuperá-los como objetos administrados a partir de um namespace Java Naming and Directory Interface (JNDI)
- Configure as propriedades de connection factories e destinos

Criando connection factories

Para criar um connection factory, um aplicativo pode usar o construtor MQConnectionFactory, conforme mostrado no exemplo a seguir:

```
MQConnectionFactory factory = new MQConnectionFactory();
```

Esta instrução cria um objeto MQConnectionFactory com os valores padrão para todas as suas propriedades, o que significa que o aplicativo se conecta ao gerenciador de filas padrão no modo de ligações. Se desejar que um aplicativo se conecte no modo cliente ou se conecte a um gerenciador de filas diferente do gerenciador de filas padrão, o aplicativo deve configurar as propriedades apropriadas do objeto MQConnectionFactory antes de criar a conexão. Para obter informações sobre como fazer isso, consulte “Configurando as propriedades de connection factories” na página 885.

Um aplicativo pode criar connection factories dos tipos a seguir de maneira semelhante:

- MQQueueConnectionFactory
- MQTopicConnectionFactory
- MQXAConnectionFactory
- MQXAQueueConnectionFactory
- MQXATopicConnectionFactory

Configurando as propriedades de connection factories

Um aplicativo pode configurar as propriedades de um connection factory chamando os métodos apropriados do connection factory. O connection factory pode ser um objeto administrado ou um objeto criado dinamicamente no tempo de execução.

Considere o seguinte código, por exemplo:

```
MQConnectionFactory factory = new MQConnectionFactory();
.
factory.setTransportType(WMQConstants.WMQ_CM_CLIENT);
factory.setQueueManager("QM1");
factory.setHostName("HOST1");
factory.setPort(1415);
factory.setChannel("QM1.SVR");
```

Este código cria um objeto MQConnectionFactory e, em seguida, define cinco propriedades do objeto. O efeito de configurar estas propriedades é que o aplicativo se conecta ao gerenciador de filas QM1 no modo cliente usando um canal de MQI chamado QM1.SVR. O gerenciador de filas está em execução em um sistema com nome do host HOST1 e o listener para o gerenciador de filas está atendendo no número da porta 1415.

Para uma conexão em tempo real com um broker, um aplicativo pode usar o seguinte código:

```
MQConnectionFactory factory = new MQConnectionFactory();
.
factory.setTransportType(WMQConstants.WMQ_CM_DIRECT);
factory.setHostName("HOST2");
factory.setPort(1507);
```

Esse código supõe que o broker está sendo executado em um sistema com o nome do host HOST2 e atendendo no número de porta 1507.

Um aplicativo que usa uma conexão em tempo real com um broker pode usar apenas o estilo de sistema de mensagens de publicação/assinatura. Ele não pode usar o estilo ponto a ponto do sistema de mensagens.

Apenas determinadas combinações de propriedades de um connection factory são válidas. Para obter informações sobre quais combinações são válidas, consulte Dependências entre propriedades de classes do WebSphere MQ para objetos JMS

Para obter mais informações sobre as propriedades de um connection factory, além dos métodos usados para configurar as propriedades dele, consulte Propriedades de objetos do IBM WebSphere MQ classes for JMS.

Criando destinos

Para criar um objeto de Fila, um aplicativo pode usar o construtor MQQueue, conforme mostrado no exemplo a seguir:

```
MQQueue q1 = new MQQueue("Q1");
```

Esta instrução cria um objeto MQQueue com os valores padrão para todas as suas propriedades. O objeto representa uma fila do WebSphere MQ chamada Q1 que pertence ao gerenciador de fila local.. Essa fila pode ser uma fila local, uma fila de alias ou uma definição de fila remota.

Uma forma alternativa do construtor MQQueue possui dois parâmetros, conforme mostrado no exemplo a seguir:

```
MQQueue q2 = new MQQueue("QM2", "Q2");
```

O objeto MQQueue criado por essa instrução representa uma fila do WebSphere MQ chamada Q2 que pertence ao gerenciador de filas QM2. O gerenciador de filas identificado dessa forma pode ser o gerenciador de filas locais ou um gerenciador de filas remotas. Se for um gerenciador de filas remotas, o WebSphere MQ deverá ser configurado para que, quando o aplicativo enviar uma mensagem para esse

destino, o Websphere MQ possa rotear a mensagem do gerenciador de fila local para o gerenciador de filas remotas..

O construtor MQQueue também pode aceitar um Identificador Uniforme de Recurso (URI) de fila como um único parâmetro. Um URI de fila é uma cadeia que especifica o nome de uma fila do WebSphere MQ e, opcionalmente, o nome do gerenciador de filas que possui a fila e uma ou mais propriedades do objeto MQQueue A instrução a seguir contém um exemplo de um URI de fila:

```
MQQueue q3 = new MQQueue("queue://QM3/Q3?persistence=2&priority=5");
```

O objeto MQQueue criado por essa instrução representa uma fila do WebSphere MQ chamada Q3 que pertence ao gerenciador de filas QM3e todas as mensagens enviadas para esse destino são persistentes e têm prioridade 5. Para obter mais informações sobre URIs de fila, consulte “Identificadores uniformes de recursos (URIs)” na página 891 Para obter uma maneira alternativa de configurar as propriedades de um objeto MQQueue, consulte “Configurando as propriedades de destinos” na página 886.

Para criar um objeto Tópico, um aplicativo pode usar o construtor MQTopic, conforme mostrado no exemplo a seguir:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
```

Essa instrução cria um objeto MQTopic com os valores padrão para todas as suas propriedades. O objeto representa um tópico chamado Sport/Football/Results.

O construtor MQTopic também pode aceitar um URI de tópico como um parâmetro. Um URI de tópico é uma sequência que especifica o nome de um tópico e, opcionalmente, uma ou mais propriedades do objeto MQTopic. A instrução a seguir contém um exemplo de um URI de tópico:

```
MQTopic t2 = new MQTopic("topic://Sport/Tennis/Results?persistence=1&priority=0");
```

O objeto MQTopic criado por esta instrução representa um tópico chamado Esporte/Tênis/Resultados, e todas as mensagens enviadas para este destino são não persistentes e têm uma prioridade de 0. Para obter mais informações sobre URIs de tópicos, consulte “Identificadores uniformes de recursos (URIs)” na página 891. Para obter uma maneira alternativa de configurar as propriedades de um objeto MQTopic, consulte “Configurando as propriedades de destinos” na página 886.

Configurando as propriedades de destinos

Um aplicativo pode configurar as propriedades de um destino chamando os métodos apropriados do destino. O destino pode ser um objeto administrado ou um objeto criado dinamicamente no tempo de execução.

Considere o seguinte código, por exemplo:

```
MQQueue q1 = new MQQueue("Q1");
.
q1.setPersistence(WMQConstants.WMQ_PER_PER);
q1.setPriority(5);
```

Este código cria um objeto MQQueue e, em seguida, configura duas propriedades do objeto. O efeito de configurar estas propriedades é que todas as mensagens enviadas ao destino são persistentes e têm uma prioridade 5.

Um aplicativo pode configurar as propriedades de objeto MQTopic de maneira semelhante, conforme mostrado no exemplo a seguir:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
.
t1.setPersistence(WMQConstants.WMQ_PER_NON);
t1.setPriority(0);
```

Este código cria um objeto MQTopic e, em seguida, configura duas propriedades do objeto. O efeito de configurar estas propriedades é que todas as mensagens enviadas ao destino são não persistentes e têm prioridade de 0.

Para obter mais informações sobre as propriedades de um destino, além dos métodos usados para configurar as propriedades dele, consulte Propriedades de objetos do IBM WebSphere MQ classes for JMS.

Construindo uma conexão em um aplicativo JMS

Para construir uma conexão, um aplicativo JMS usa um objeto ConnectionFactory para criar um objeto Connection e, em seguida, inicia a conexão.

Para criar um objeto Connection, um aplicativo usa o método createConnection() de um objeto ConnectionFactory, conforme mostrado no exemplo a seguir:

```
ConnectionFactory factory;
Connection connection;
.
.
.
connection = factory.createConnection();
```

Quando uma conexão JMS é criada, as classes do IBM WebSphere MQ classes for JMS cria uma manipulação de conexões (Hconn) e inicia uma conversa com o gerenciador de filas.

A interface QueueConnectionFactory e a interface TopicConnectionFactory herda, cada uma, o método createConnection() da interface ConnectionFactory. Portanto, é possível usar o método createConnection() para criar um objeto específico do domínio, conforme mostrado no exemplo a seguir:

```
QueueConnectionFactory qcf;
Connection connection;
.
.
.
connection = qcf.createConnection();
```

Esse fragmento de código cria um objeto QueueConnection. Um aplicativo pode agora executar uma operação independente de domínio nesse objeto ou uma operação que é aplicável apenas ao domínio ponto a ponto. Entretanto, se o aplicativo tenta executar uma operação que é aplicável apenas ao domínio de publicação/assinatura, uma exceção IllegalStateException é emitida com a seguinte mensagem:

```
JMSMQ1112: Operation for a domain specific object was not valid.
           Operation createProducer() is not valid for type com.ibm.mq.jms.MQTopic
```

Isso ocorre porque a conexão foi criada a partir de um connection factory específico do domínio.

Nota: Observe que o ID do processo do aplicativo é usado como a identidade do usuário padrão a ser transmitida para o gerenciador de filas. Se o aplicativo estiver em execução no modo de transporte do cliente, esse ID de processo deve existir, com as autorizações relevantes, no servidor. Se você deseja usar uma identidade, use o método createConnection (nome do usuário, senha).

A especificação JMS indica que uma conexão é criada no estado `stopped` .. Até que uma conexão é iniciada, um consumidor de mensagem que está associado à conexão não pode receber nenhuma mensagem. Para iniciar uma conexão, um aplicativo usa o método start() de um objeto Connection, conforme mostrado no exemplo a seguir:

```
connection.start();
```

Criando uma sessão em um aplicativo JMS

Para criar uma sessão, um aplicativo JMS utiliza o método createSession() de um objeto Connection.

O método createSession() possui dois parâmetros:

1. Um parâmetro que especifica se a sessão está transacionada ou não transacionada
2. Um parâmetro que especifica o modo de confirmação para a sessão

Por exemplo, o código a seguir cria uma sessão que não está transacionada e tem um modo de confirmação de AUTO_ACKNOWLEDGE:

```
Session session;
.
boolean transacted = false;
session = connection.createSession(transacted, Session.AUTO_ACKNOWLEDGE);
```

Quando uma sessão JMS é criada, as classes do IBM WebSphere MQ classes for JMS cria uma manipulação de conexões (Hconn) e inicia uma conversa com o gerenciador de filas.

Um objeto Session, e qualquer objeto MessageProducer ou MessageConsumer criado a partir dele, não pode ser usado simultaneamente por diferentes encadeamentos de um aplicativo multiencadeado. A maneira mais simples de garantir que esses objetos não sejam usados simultaneamente é criar um objeto Session separado para cada encadeamento.

Sessões transacionadas em aplicativos JMS..

Os aplicativos JMS podem executar transações locais criando primeiro uma sessão transacionada Um aplicativo pode confirmar ou retroceder uma transação.

Aplicativos JMS podem executar transações locais. Uma transação local é uma transação que envolve mudanças apenas para os recursos do gerenciador de filas ao qual o aplicativo está conectado. Para executar transações locais, um aplicativo deve primeiro criar uma sessão transacionada chamando o método createSession() de um objeto Connection, especificando como um parâmetro que a sessão é transacionada. Em seguida, todas as mensagens enviadas e recebidas dentro da sessão são agrupadas em uma sequência de transações. Uma transação terminará quando o aplicativo confirmar ou recuperar as mensagens que ele enviou e recebeu desde o início da transação.

Para confirmar uma transação, um aplicativo chama o método commit() do objeto Session. Quando uma transação for confirmada, todas as mensagens enviadas dentro da transação se tornarão disponíveis para entrega para outros aplicativos e todas as mensagens recebidas dentro da transação serão confirmadas para que o servidor de sistema de mensagens não tente entregá-las ao aplicativo novamente. No domínio ponto a ponto, o servidor de sistema de mensagens também remove as mensagens recebidas a partir de suas filas.

Para retroceder uma transação, um aplicativo chama o método rollback() do objeto Session. Quando uma transação for retrocedida, todas as mensagens enviadas dentro da transação serão descartadas pelo servidor de sistema de mensagens e todas as mensagens recebidas dentro da transação se tornarão disponíveis para entrega novamente. No domínio de ponto a ponto, as mensagens que foram recebidas são colocadas de volta em suas filas e se tornam visíveis a outros aplicativos novamente.

Uma nova transação será iniciada automaticamente quando um aplicativo criar uma sessão transacionada ou chamar o método commit() ou rollback(). Portanto, uma sessão transacionada sempre possui uma transação ativa.

Quando um aplicativo fechar uma sessão transacionada, um retrocesso implícito ocorrerá. Quando um aplicativo fechar uma conexão, um retrocesso implícito ocorrerá para todas as sessões transacionadas da conexão.

Se um aplicativo terminar sem fechar uma conexão, um retrocesso implícito também ocorrerá para todas as sessões transacionadas da conexão.

Uma transação é completamente contida dentro de uma sessão transacionada. Uma transação não pode abranger as sessões. Isso significa que não é possível para um aplicativo enviar e receber mensagens em duas ou mais sessões transacionadas e, em seguida, confirmar ou retroceder todas estas ações como uma única transação.

Modos de Confirmação de sessões JMS

Cada sessão que não é transacionada possui um modo de confirmação que determina como as mensagens recebidas pelo aplicativo são confirmadas. Três modos de confirmação estão disponíveis e a opção de modo de confirmação afeta o design do aplicativo.

Se uma sessão não for transacionada, a maneira que as mensagens recebidas pelo aplicativo são confirmadas será determinada pelo modo de confirmação da sessão. Três modos de confirmação são descritos nos parágrafos a seguir:

AUTO_ACKNOWLEDGE

A sessão confirma automaticamente cada mensagem recebida pelo aplicativo.

Se as mensagens forem entregues de forma síncrona para o aplicativo, a sessão confirmará o recebimento de uma mensagem toda vez que uma chamada Receive for concluída com sucesso. Se as mensagens forem entregues de forma assíncrona, a sessão confirmará o recebimento de uma mensagem toda vez que uma chamada ao método onMessage() de um listener de mensagens for concluída com sucesso.

Se o aplicativo receber uma mensagem com sucesso, mas uma falha evitar a ocorrência de confirmação, a mensagem se tornará disponível para a entrega novamente. O aplicativo deve, portanto, ser capaz de manipular uma mensagem que é entregue novamente.

DUPS_OK_ACKNOWLEDGE

A sessão confirma as mensagens recebidas pelo aplicativo em momentos que ele seleciona.

O uso desse modo de confirmação reduz a quantidade de trabalho que a sessão deve fazer, mas uma falha que evita a confirmação de mensagens pode fazer com que mais de uma mensagem se torne disponível para entrega novamente. O aplicativo deve, portanto, ser capaz de manipular mensagens que são entregues novamente.

Restrição: Nos modos AUTO_RECONHEÇO e DUPS_OK_RECONHEÇO, o JMS não suporta um aplicativo lançando uma exceção não manipulada em um listener de mensagem Isso significa que as mensagens serão sempre confirmadas quando o listener de mensagem retornar, independentemente de se ele foi processado com sucesso (contanto que quaisquer falhas não sejam fatais e não evitem que o aplicativo continue). Se você requerer melhor controle de confirmação da mensagem, use o CLIENT_ACKNOWLEDGE ou modos transacionados, que fornecem ao aplicativo controle total das funções de confirmação.

CLIENT_ACKNOWLEDGE

O aplicativo confirma as mensagens que ele recebe chamando o método Acknowledge da classe Message.

O aplicativo pode confirmar o recebimento de cada mensagem individualmente ou receber um lote de mensagens e chamar o método Acknowledge apenas para a última mensagem que ele recebe. Quando o método Acknowledge for chamado, todas as mensagens recebidas desde a última vez que o método foi chamado serão confirmadas.

Juntamente com qualquer um destes modos de confirmação, um aplicativo pode interromper e reiniciar a entrega de mensagens em uma sessão chamando o método Recover da classe Session. As mensagens recebidas, mas não confirmadas anteriormente são entregues novamente. No entanto, elas não podem ser entregues na mesma sequência em que foram entregues anteriormente. No entretanto, mensagens de prioridade superior podem ter chegado e algumas das mensagens originais podem ter expirado. No domínio ponto a ponto, algumas das mensagens originais podem ter sido consumidas por outro aplicativo.

Um aplicativo pode determinar se uma mensagem está sendo entregue novamente examinando o conteúdo do campo de cabeçalho JMSRedelivered da mensagem. O aplicativo faz isso chamando o método getJMSRedelivered() da classe Message.

Criando destinos em um aplicativo JMS

Em vez de recuperar destinos como objetos administrados de um namespace Java Naming and Directory Interface (JNDI), um aplicativo JMS pode usar uma sessão para criar destinos dinamicamente no tempo de execução. Um aplicativo pode usar um identificador uniforme de recursos (URI) para identificar uma fila do WebSphere MQ ou um tópico e, opcionalmente, para especificar uma ou mais propriedades de um objeto de Fila ou Tópico.

Usando uma sessão para criar objetos Queue

Para criar um objeto Queue, um aplicativo pode usar o método createQueue() de um objeto Session, conforme mostrado no exemplo a seguir:

```
Session session;
.
Queue q1 = session.createQueue("Q1");
```

Esse código cria um objeto Queue com os valores padrão para todas as suas propriedades. O objeto representa uma fila do WebSphere MQ chamada Q1 que pertence ao gerenciador de fila local.. Essa fila pode ser uma fila local, uma fila de alias ou uma definição de fila remota.

O método createQueue() também aceita um URI de URI como um parâmetro. Um URI de fila é uma sequência que especifica o nome de uma fila do WebSphere MQ e, opcionalmente, o nome do gerenciador de filas que possui a fila e uma ou mais propriedades do objeto de fila A instrução a seguir contém um exemplo de um URI de fila:

```
Queue q2 = session.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

O objeto Queue criado por essa instrução representa uma fila do WebSphere MQ chamada Q2 que pertence a um gerenciador de filas chamado QM2e todas as mensagens enviadas para esse destino são persistentes e têm uma prioridade 5. O gerenciador de filas identificado dessa forma pode ser o gerenciador de filas locais ou um gerenciador de filas remotas. Se for um gerenciador de fila remoto, o WebSphere MQ deverá ser configurado para que, quando o aplicativo enviar uma mensagem para esse destino, o Websphere MQ possa rotear a mensagem do gerenciador de fila local para o gerenciador de filas QM2. Para obter mais informações sobre URIs, consulte “Identificadores uniformes de recursos (URIs)” na página 891.

Observe que o parâmetro no método createQueue() contém informações específicas do provedor. Portanto, usar o método createQueue() para criar um objeto Queue, em vez de recuperar um objeto Queue como um objeto administrado de um namespace JNDI, pode tornar seu aplicativo menos móvel.

Um aplicativo pode criar um objeto TemporaryQueue usando o método createTemporaryQueue() de um objeto Session, conforme mostrado no exemplo a seguir:

```
TemporaryQueue q3 = session.createTemporaryQueue();
```

Embora uma sessão seja usada para criar uma fila temporária, o escopo de uma fila temporária é a conexão que foi usada para criar a sessão. Qualquer uma das sessões da conexão pode criar produtores e consumidores de mensagens para a fila temporária. A fila temporária permanece até o término da conexão ou até o aplicativo excluir explicitamente a fila temporária usando o método TemporaryQueue.delete(), o que ocorrer primeiro.

Quando um aplicativo cria uma fila temporária, as classes do WebSphere MQ para JMS criam uma fila dinâmica no gerenciador de filas ao qual o aplicativo está conectado A propriedade TEMPMODEL do connection factory especifica o nome da fila modelo usada para criar a fila dinâmica e a propriedade TEMPQPREFIX do connection factory especifica o prefixo usado para formar o nome da fila dinâmica.

Usando uma sessão para criar objetos Topic

Para criar um objeto Topic, um aplicativo pode usar o método createTopic() de um objeto Session, conforme mostrado no exemplo a seguir:

```
Session session;
.
Topic t1 = session.createTopic("Sport/Football/Results");
```

Este código cria um objeto Topic com os valores padrão para todas as suas propriedades. O objeto representa um tópico chamado Sport/Football/Results.

O método createTopic() também aceita um URI de tópico como um parâmetro. Um URI de tópico é uma sequência que especifica o nome de um tópico e, como opção, uma ou mais propriedades do objeto Topic. O código a seguir contém um exemplo de um URI de tópico:

```
String uri = "topic://Sport/Tennis/Results?persistence=1&priority=0";
Topic t2 = session.createTopic(uri);
```

O objeto Tópico criado por este código representa um tópico chamado Esporte/Tênis/Resultados, e todas as mensagens enviadas para este destino são não persistentes e têm prioridade de 0. Para obter mais informações sobre URIs de tópicos, consulte "Identificadores uniformes de recursos (URIs)" na página 891.

Observe que o parâmetro no método createTopic() contém informações específicas do provedor. Portanto, usar o método createTopic() para criar um objeto Topic, em vez de recuperar um objeto Topic como um objeto administrado a partir de um namespace JNDI, pode tornar seu aplicativo menos móvel.

Um aplicativo pode criar um objeto TemporaryTopic usando o método createTemporaryTopic() de um objeto Session, conforme mostrado no exemplo a seguir:

```
TemporaryTopic t3 = session.createTemporaryTopic();
```

Embora uma sessão seja usada para criar um tópico temporário, o escopo de um tópico temporário é a conexão que foi usada para criar a sessão. Qualquer uma das sessões da conexão pode criar produtores e consumidores de mensagens para o tópico provisório. O tópico provisório permanece até o término da conexão ou até o aplicativo excluir explicitamente o tópico provisório usando o método TemporaryTopic.delete(), o que ocorrer primeiro.

Quando um aplicativo cria um tópico temporário, as classes WebSphere MQ para JMS criam um tópico com um nome que começa com os caracteres TEMP/*tempTopicPrefix*, em que *tempTopicPrefix* é o valor da propriedade TEMPTOPICPREFIX do connection factory.

Identificadores uniformes de recursos (URIs)

Um URI de fila é uma sequência que especifica o nome de uma fila do WebSphere MQ e, opcionalmente, o nome do gerenciador de filas que possui a fila e uma ou mais propriedades do objeto de fila criado pelo aplicativo Um URI de tópico é uma sequência que especifica o nome de um tópico e, como opção, uma ou mais propriedades do objeto Topic criado pelo aplicativo.

Um URI de fila tem o formato a seguir:

```
queue://[qMgrName]/qName[?propertyName1=propertyValue1
                         &propertyName2=propertyValue2
                         &...]
```

Um URI de tópico tem o formato a seguir:

```
topic://topicName[?propertyName1=propertyValue1
                  &propertyName2=propertyValue2
                  &...]
```

As variáveis nesses formatos têm os significados a seguir:

qMgrName
O nome do gerenciador de filas que possui a fila identificada pelo URI.

O gerenciador de filas pode ser o gerenciador de filas locais ou um gerenciador de filas remotas. Se ele for um gerenciador de filas remotas, o WebSphere MQ deverá ser configurado para que, quando um aplicativo enviar uma mensagem para a fila, o Websphere MQ possa rotear a mensagem do gerenciador de fila local para o gerenciador de filas remotas.

Se nenhum nome for especificado, o gerenciador de filas locais será assumido.

qName
O nome da fila do WebSphere MQ ..

A fila pode ser uma fila local, uma fila de alias ou uma definição de fila remota.

Para obter as regras para criar nomes de filas, consulte Regras para nomear IBM WebSphere MQ objetos

topicName
O nome do tópico.

Para obter as regras para criação de nomes de tópicos, consulte Regras para nomenclatura de objetos do IBM WebSphere MQ. Evite o uso dos caracteres curingas +, #, * e ? em nomes de tópicos. Nomes de tópicos contendo esses caracteres podem causar resultados inesperados quando assinados. Consulte Usando sequências de tópicos.

***propertyName1*, *propertyName2*, ...**
Os nomes das propriedades do objeto Queue ou Topic criado pelo aplicativo. Tabela 123 na página 892 lista os nomes de propriedades válidos que podem ser usados em um URI.

Se nenhuma propriedade for especificada, o objeto Queue ou Topic terá os valores padrão para todas as suas propriedades.

***propertyValue1*, *propertyValue2*, ...**
Os valores das propriedades do objeto Queue ou Topic criado pelo aplicativo. Tabela 123 na página 892 lista os valores de propriedades válidos que podem ser usados em um URI.

Os colchetes ([]) denotam um componente opcional e as reticências (...) significam que a lista de pares de nome-valor de propriedades, se presente, pode conter um ou mais pares de nome-valor.

Tabela 123 na página 892 lista os nomes de propriedades válidos e os valores válidos que podem ser usados em URIs de fila e de tópico. Embora a ferramenta de administração JMS do WebSphere MQ use constantes simbólicas para os valores de propriedades, URIs não podem conter constantes simbólicas..

Tabela 123. Nomes de propriedades e valores válidos para uso nos URIs de fila e tópico

Nome da Propriedade	**Descrição**	**Valores Válidos**
CCSID	Como os dados de caractere no corpo de uma mensagem são representados quando WebSphere MQ classes para JMS encaminha a mensagem para o destino	• Qualquer identificador de conjunto de caractere codificado suportado pelo WebSphere MQ.
codificação	Como os dados numéricos no corpo de uma mensagem são representados quando WebSphere MQ classes para JMS encaminha a mensagem para o destino	• Qualquer valor válido para o campo *Codificação* em um descritor de mensagem WebSphere MQ .
expiração	O tempo de vida das mensagens enviadas ao destino	• -2 - Conforme especificado na chamada send() ou, se não especificado na chamada send(), o tempo de vida padrão do produtor de mensagem. • 0 - Uma mensagem enviada ao destino nunca expira. • Um número inteiro positivo que especifica o tempo de vida em milissegundos.

Tabela 123. Nomes de propriedades e valores válidos para uso nos URIs de fila e tópico (continuação)

Nome da Propriedade	**Descrição**	**Valores Válidos**
multicast	A configuração de multicast para um tópico ao usar uma conexão em tempo real com um broker	A lista a seguir contém os valores válidos. Associado a cada valor está o valor correspondente da propriedade MULTICAST, conforme usado na ferramenta de administração JMS do WebSphere MQ . Para obter uma descrição da propriedade MULTICAST e dos respectivos valores válidos, consulte Propriedades de objetos do IBM WebSphere MQ classes for JMS. • -1 - ASCF • 0 - DISABLED • 3 - NOTR • 5 - RELIABLE • 7 - ENABLED
persistence	A persistência de mensagens enviadas ao destino	• -2 - Conforme especificado na chamada send() ou, se não especificado na chamada send(), a persistência padrão do produtor de mensagem. • -1-Conforme especificado no atributo *DefPersistence* da fila ou tópico WebSphere MQ . • 1 - Não persistente. • 2 - Persistente. • 3-Equivalente ao valor HIGH para a propriedade PERSISTENCE, conforme usado na ferramenta de administração JMS do WebSphere MQ . Para obter uma explicação desse valor, consulte “Mensagens persistentes do JMS” na página 916.
priority	A prioridade de mensagens enviadas ao destino	• -2 - Conforme especificado na chamada send() ou, se não especificado na chamada send(), a prioridade padrão do produtor de mensagem. • -1-Conforme especificado pelo atributo *DefPriority* da fila ou tópico WebSphere MQ . • Um número inteiro no intervalo de 0 a 9 que especifica a prioridade de mensagens enviadas ao destino.

Tabela 123. Nomes de propriedades e valores válidos para uso nos URIs de fila e tópico (continuação)		
Nome da Propriedade	**Descrição**	**Valores Válidos**
targetClient	Se as mensagens enviadas ao destino contêm um cabeçalho MQRFH2	• 0 - As mensagens contêm um cabeçalho MQRFH2. • 1 - As mensagens não contêm um cabeçalho MQRFH2.

Por exemplo, o URI a seguir identifica uma fila do WebSphere MQ chamada Q1 que pertence ao gerenciador de fila local. Um objeto Queue criado usando esse URI tem os valores padrão para todas as suas propriedades.

```
queue:///Q1
```

O URI a seguir identifica uma fila do WebSphere MQ chamada Q2 que pertence a um gerenciador de filas chamado QM2. Todas as mensagens enviadas para este destino têm uma prioridade de 6. As propriedades remanescentes do objeto Fila criado usando esta URI possuem seus valores padrão.

```
queue://QM2/Q2?priority=6
```

O URI a seguir identifica um tópico chamado Sport/Athletics/Results. Todas as mensagens enviadas para este destino são não persistentes e têm prioridade de 0. As propriedades remanescentes do objeto Tópico criado usando esta URI possuem seus valores padrão.

```
topic://Sport/Athletics/Results?persistence=1&priority=0
```

Enviando mensagens em um aplicativo JMS

Antes de um aplicativo JMS poder enviar mensagens para um destino, ele deve primeiro criar um objeto MessageProducer para o destino.. Para enviar uma mensagem para o destino, o aplicativo cria um objeto de mensagem e, em seguida, chama o método send() do objeto MessageProducer.

Um aplicativo usa um objeto MessageProducer para enviar mensagens. Um aplicativo normalmente cria um objeto MessageProducer para um destino específico, que pode ser uma fila ou um tópico, para que todas as mensagens enviadas usando o produtor de mensagens sejam enviadas ao mesmo destino. Portanto, antes de um aplicativo poder criar um objeto MessageProducer, ele deve primeiro criar um objeto Queue ou Topic. Para obter informações sobre como criar um objeto Queue ou Topic, consulte os tópicos a seguir:

- “Usando JNDI para Recuperar Objetos Administrados em um Aplicativo JMS” na página 876
- “Usando as extensões JMS do IBM” na página 877
- “Usando as extensões JMS do WebSphere MQ” na página 884
- “Criando destinos em um aplicativo JMS” na página 889

Para criar um objeto MessageProducer, um aplicativo usa o método createProducer() de um objeto de sessão conforme mostrado no exemplo a seguir:

```
MessageProducer producer = session.createProducer(destination);
```

O parâmetro `destination` é um objeto Queue ou Topic que o aplicativo criou anteriormente.

Antes de um aplicativo poder enviar uma mensagem, ele deve criar um objeto de mensagem. O corpo de uma mensagem contém os dados do aplicativo e o JMS define cinco tipos de corpo da mensagem:

- bytes
- Mapear
- Object

- Fluxo
- text

Cada tipo de corpo da mensagem tem sua própria interface JMS, que é uma subinterface da interface Message e um método na interface Session para criar uma mensagem com esse tipo de corpo. Por exemplo, a interface para uma mensagem de texto é chamada TextMessage, e um aplicativo usa o método createTextMessage() de um objeto de sessão para criar uma mensagem de texto, conforme mostrado na seguinte instrução:

```
TextMessage outMessage = session.createTextMessage(outString);
```

Para obter mais informações sobre mensagens e corpos de mensagem, consulte “Mensagens JMS” na página 818.

Para enviar uma mensagem, um aplicativo usa o método send() de um objeto MessageProducer conforme mostrado no exemplo a seguir:

```
producer.send(outMessage);
```

Um aplicativo pode usar o método send() para enviar mensagens em qualquer domínio de mensagens. A natureza do destino determina qual domínio de mensagens será usado. No entanto, TopicPublisher, a sub-interface do MessageProducer que é específica para o domínio de publicação/assinatura, também tem um método publish(), que pode ser usado no lugar do método send(). Os dois métodos são funcionalmente os mesmos.

Um aplicativo pode criar um objeto MessageProducer com nenhum destino especificado. Neste caso, o aplicativo deve especificar o destino ao chamar o método send().

Se um aplicativo enviar uma mensagem em uma transação, a mensagem não será entregue ao seu destino até que a transação seja confirmada. Isso significa que um aplicativo não pode enviar uma mensagem e receber uma resposta para a mensagem na mesma transação.

Um destino pode ser configurado para que quando um aplicativo enviar mensagens para ele, as classes do WebSphere MQ para JMS encaminhe a mensagem e retorne o controle para o aplicativo sem determinar se o gerenciador de filas recebeu a mensagem com segurança. Isto é, às vezes, referido como *postagem assíncrona*. Para obter mais informações, consulte “Colocando mensagens assincronamente em classes IBM WebSphere MQ para JMS” na página 932.

Recebendo mensagens em um aplicativo JMS

Um aplicativo usa um consumidor de mensagens para receber mensagens. Um assinante de tópico permanente é um consumidor de mensagens que recebe todas as mensagens enviadas a um destino, incluindo aquelas enviadas enquanto o consumidor está inativo. Um aplicativo pode selecionar quais mensagens deseja receber usando um seletor de mensagens e pode receber mensagens de forma assíncrona usando um listener de mensagem.

Um aplicativo usa um objeto MessageConsumer para receber mensagens. Um aplicativo cria um objeto MessageConsumer para um destino específico, que pode ser uma fila ou um tópico, para que todas as mensagens recebidas usando o consumidor de mensagens sejam recebidas do mesmo destino. Portanto, antes de um aplicativo poder criar um objeto MessageConsumer, ele deverá primeiramente criar um objeto Fila ou Tópico. Para obter informações sobre como criar um objeto Queue ou Topic, consulte os tópicos a seguir:

- “Usando JNDI para Recuperar Objetos Administrados em um Aplicativo JMS” na página 876
- “Usando as extensões JMS do IBM” na página 877
- “Usando as extensões JMS do WebSphere MQ” na página 884
- “Criando destinos em um aplicativo JMS” na página 889

Para criar um objeto MessageConsumer, um aplicativo usa o método createConsumer() de um objeto de Sessão, conforme mostrado no exemplo a seguir:

```
MessageConsumer consumer = session.createConsumer(destination);
```

O parâmetro `destination` é um objeto Queue ou Topic que o aplicativo criou anteriormente.

O aplicativo, então, usa o método receive() do objeto MessageConsumer para receber uma mensagem do destino, conforme mostrado no exemplo a seguir:

```
Message inMessage = consumer.receive(1000);
```

O parâmetro na chamada receive() especifica quanto tempo em milissegundos o método aguarda uma mensagem adequada chegar, se nenhuma mensagem estiver disponível imediatamente. Se você omitir esse parâmetro, a chamada é bloqueada indefinidamente até que uma mensagem adequada chegue. Se você não quiser que o aplicativo aguarde uma mensagem, use o método receiveNoWait() no lugar.

O método receive() retorna uma mensagem de um tipo específico. Por exemplo, quando um aplicativo recebe uma mensagem de texto, o objeto retornado pela chamada receive() é um objeto TextMessage.

No entanto, o tipo declarado do objeto retornado por uma chamada receive() é um objeto de Mensagem. Portanto, para extrair os dados do corpo de uma mensagem que acabou de ser recebida, o aplicativo deve lançar da classe de Mensagem para a subclasse mais específica, como TextMessage. Se o tipo da mensagem não for conhecido, o aplicativo poderá usar o operador `instanceof` para determinar o tipo. É sempre uma boa prática para um aplicativo determinar o tipo de uma mensagem antes do casting para que os erros possam ser manipulados com êxito.

O código a seguir usa o operador `instanceof` e mostra como extrair os dados do corpo de uma mensagem de texto:

```
if (inMessage instanceof TextMessage) {
  String replyString = ((TextMessage) inMessage).getText();
  .
  .
  .
} else {
  // Print error message if Message was not a TextMessage.
  System.out.println("Reply message was not a TextMessage");
}
```

Se um aplicativo enviar uma mensagem em uma transação, a mensagem não será entregue ao seu destino até que a transação seja confirmada. Isso significa que um aplicativo não pode enviar uma mensagem e receber uma resposta para a mensagem na mesma transação.

Se um consumidor de mensagens receber mensagens de um destino que esteja configurado para leitura antecipada, quaisquer mensagens não persistentes que estejam no buffer de leitura antecipada quando o aplicativo terminar serão descartadas.

No domínio de publicação / assinatura, o JMS identifica dois tipos de consumidor de mensagens, assinante de tópico não durável e assinante de tópico durável, que são descritos nas duas seções a seguir...

Assinantes de tópico não duráveis

Um assinante de tópico não durável recebe apenas aquelas mensagens que são publicadas enquanto o assinante está ativo. Uma assinatura não durável é iniciada quando um aplicativo cria um assinante de tópico não durável e é finalizado quando o aplicativo fecha o assinante ou quando o assinante está fora do escopo. Como uma extensão nas classes do WebSphere MQ para JMS, um assinante de tópico não durável também recebe publicações retidas, mas não ao usar uma conexão em tempo real com um broker

Para criar um assinante de tópico não durável, um aplicativo pode usar o método createConsumer independente de domínio (), especificando um objeto Topic como o destino. Como alternativa, um aplicativo pode usar o método createSubscriber() específico de domínio, conforme mostrado no exemplo a seguir:

```
TopicSubscriber subscriber = session.createSubscriber(topic);
```

O parâmetro `topic` é um objeto do Tópico que o aplicativo criou anteriormente.

Assinantes de tópico duráveis

Restrição: Um aplicativo não pode criar os assinantes do tópico durável ao usar uma conexão em tempo real com um broker.

Um assinante de tópico durável recebe todas as mensagens que são publicadas durante a vida útil de uma assinatura durável. Essas mensagens incluem todas aquelas que são publicadas enquanto o assinante não está ativo. Como uma extensão nas classes WebSphere MQ para JMS, um assinante de tópico durável também recebe publicações retidas.

Para criar um assinante de tópico durável, um aplicativo usa o método createDurableSubscriber() de um objeto de Sessão, conforme mostrado no exemplo a seguir:

```
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001");
```

Na chamada createDurableSubscriber(), o primeiro parâmetro é um objeto do Tópico que o aplicativo criou anteriormente, e o segundo parâmetro é um nome que é usado para identificar a assinatura durável.

A sessão usada para criar um assinante de tópico durável deve ter um identificador de cliente associado. O identificador de cliente associado a uma sessão é o mesmo que o identificador do cliente para a conexão que é usado para criar a sessão. O identificador de cliente pode ser especificado configurando a propriedade CLIENTID do objeto ConnectionFactory. Como alternativa, um aplicativo pode especificar o identificador de cliente chamando o método setClientID() do objeto Conexão.

O nome que é usado para identificar uma assinatura durável deve ser exclusivo somente no identificador de cliente e, portanto, o identificador de cliente faz parte do identificador integral exclusivo de uma assinatura durável. Para continuar a usar uma assinatura durável que foi criada anteriormente, um aplicativo deve criar um assinante de tópico durável usando uma sessão com o mesmo identificador de cliente que o associado à assinatura durável e usando o mesmo nome de assinatura.

Uma assinatura durável é iniciada quando um aplicativo cria um assinante de tópico durável usando um identificador de cliente e nome de assinatura para a qual nenhuma assinatura durável existe atualmente. No entanto, uma assinatura durável não termina quando o aplicativo fecha o assinante de tópico durável. Para finalizar uma assinatura durável, um aplicativo deve chamar o método unsubscribe() de um objeto de Sessão que possui o mesmo identificador de cliente que o associado à assinatura durável. O parâmetro na chamada unsubscribe() é o nome de assinatura, conforme mostrado no exemplo a seguir:

```
session.unsubscribe("D_SUB_000001");
```

O escopo de uma assinatura durável é um gerenciador de filas. Se uma assinatura durável existir em um gerenciador de filas e um aplicativo conectado a outro gerenciador de filas criar uma assinatura durável com o mesmo identificador de cliente e nome de assinatura, as duas assinaturas duráveis serão completamente independentes.

Seletores de mensagens

Um aplicativo pode especificar que apenas aquelas mensagens que atendam a determinados critérios sejam retornadas pelas chamadas receive() sucessivas. Ao criar um objeto MessageConsumer, o aplicativo pode especificar uma expressão de Linguagem de Consulta Estruturada (SQL) que determina quais mensagens são recuperadas. Esta expressão SQL é chamada de *seletor de mensagem*. O seletor de mensagem pode conter os nomes de campos de cabeçalho da mensagem JMS e propriedades de mensagens Para obter informações sobre como construir um seletor de mensagem, consulte “Seletores de mensagens no JMS..” na página 819.

O exemplo a seguir mostra como um aplicativo pode selecionar mensagens com base em uma propriedade definida pelo usuário chamado myProp:

```
MessageConsumer consumer;
.
consumer = session.createConsumer(destination, "myProp = 'blue'");
```

A especificação JMS não permite que um aplicativo altere o seletor de mensagem de um consumidor de mensagem.. Após um aplicativo criar um consumidor de mensagem com um seletor de mensagem, o seletor de mensagem permanecerá durante a existência desse consumidor. Se um aplicativo precisar de mais de um seletor de mensagem, o aplicativo deverá criar um consumidor de mensagem para cada seletor de mensagem.

Observe que, quando um aplicativo é conectado a um gerenciador de filas da Versão 7, a propriedade MSGSELECTION do connection factory não tem efeito. Para otimizar o desempenho, toda seleção de mensagens é feita pelo gerenciador de filas.

Suprimindo as publicações locais

Um aplicativo pode criar um consumidor de mensagem que ignora as publicações feitas na própria conexão do consumidor. O aplicativo faz isso configurando o terceiro parâmetro em uma chamada createConsumer() para `true`, conforme mostrado no exemplo a seguir:

```
MessageConsumer consumer = session.createConsumer(topic, null, true);
```

Em uma chamada createDurableSubscriber(), o aplicativo faz isso configurando o quarto parâmetro para `true`, conforme mostrado no exemplo a seguir

```
String selector = "company = 'IBM'";
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001",
                                                              selector, true);
```

Entrega assíncrona de mensagens

Um aplicativo pode receber mensagens de forma assíncrona registrando um listener de mensagem com um consumidor de mensagens. O listener de mensagem tem um método chamado onMessage, que é chamado de maneira assíncrona quando uma mensagem adequada está disponível e cuja finalidade é processar a mensagem. O código a seguir ilustra o mecanismo:

```
import javax.jms.*;

public class MyClass implements MessageListener
{
  // The method that is called asynchronously when a suitable message is available
  public void onMessage(Message message)
  {
    System.out.println("Message is "+message);

    // The code to process the message
    .
    .
    .
  }
}
.
.
.
// Main program (possibly in another class)
.
// Creating the message listener
MyClass listener = new MyClass();

// Registering the message listener with a message consumer
consumer.setMessageListener(listener);

// The main program now continues with other processing
```

Um aplicativo pode usar uma sessão, para receber mensagens de forma síncrona usando chamadas receive() ou para receber mensagens de forma assíncrona usando listeners de mensagens, mas não para ambos. Se um aplicativo precisar receber mensagens de forma síncrona e assíncrona, ele deve criar sessões separadas.

Assim que uma sessão for configurada para receber mensagens de forma assíncrona, os métodos a seguir não poderão ser chamados na sessão ou em objetos criados a partir dessa sessão:

- MessageConsumer.receive()
- MessageConsumer.receive(long)
- MessageConsumer.receiveNoWait()
- Session.acknowledge()
- MessageProducer.send(Destination, Message)
- MessageProducer.send(Destination, Message, int, int, long)
- MessageProducer.send(Message)
- MessageProducer.send(Message, int, int, long)
- Session.commit()
- Session.createBrowser(Queue)
- Session.createBrowser(Queue, String)
- Session.createBytesMessage()
- Session.createConsumer(Destination)
- Session.createConsumer(Destination, String, boolean)
- Session.createDurableSubscriber(Topic, String)
- Session.createDurableSubscriber(Topic, String, String, boolean)
- Session.createMapMessage()
- Session.createMessage()
- Session.createObjectMessage()
- Session.createObjectMessage(Serializable)
- Session.createProducer(Destination)
- Session.createQueue(String)
- Session.createStreamMessage()
- Session.createTemporaryQueue()
- Session.createTemporaryTopic()
- Session.createTextMessage()
- Session.createTextMessage(String)
- Session.createTopic()
- Session.getAcknowledgeMode()
- Session.getMessageListener()
- Session.getTransacted()
- Session.rollback()
- Session.unsubscribe(String)

Se qualquer um desses métodos é chamado, uma JMSException contendo a mensagem:

```
JMSCC0033: Uma chamada de método síncrono não é permitida quando uma sessão está sendo usada de
forma assíncrona: 'method name'
```

é lançado.

Recebendo mensagens suspeitas

Um aplicativo pode receber uma mensagem que não pode ser processada. Pode haver várias razões pelas quais a mensagem não pode ser processada, por exemplo, a mensagem pode ter um formato incorreto. Essas mensagens são descritas como mensagens suspeitas e requerem tratamento especial para impedir a mensagem que está sendo processada recursivamente.

Para obter detalhes sobre como manipular mensagens suspeitas, consulte “Manipulando mensagens suspeitas no IBM WebSphere MQ classes for JMS” na página 901.

V 7.5.0.8 Recuperação de dados do usuário da assinatura

Se as mensagens que um aplicativo IBM WebSphere MQ classes for JMS está consumindo de uma fila são colocadas por uma assinatura durável definida administrativamente, o aplicativo precisa acessar as informações de dados do usuário que estão associadas à assinatura. Essas informações são incluídas na mensagem como uma propriedade.

A partir de Version 7.5.0, Fix Pack 8, quando uma mensagem é consumida de uma fila que contém um cabeçalho RFH2 com a pasta MQPS, o valor que está associado à chave Sud, se ela existir, é incluído como uma propriedade de Sequência para o objeto de Mensagem JMS retornado para o aplicativo IBM WebSphere MQ classes for JMS . Para ativar a recuperação dessa propriedade da mensagem, a constante JMS_IBM_SUBSCRIPTION_USER_DATA na interface JmsConstants pode ser usada com o método javax.jms.Message.getStringProperty método(java.lang.String) para obter os dados do usuário de assinatura.

No exemplo a seguir, uma assinatura durável administrativa é definida usando o comando MQSC **DEFINE SUB**:

```
DEFINE SUB('MY.SUBCRIPTION') TOPICSTR('PUBLIC') DEST('MY.SUBSCRIPTION.Q')
USERDATA('Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q')
```

As cópias de mensagens que são publicadas na sequência de tópicos `PUBLIC` são colocadas na fila, `MY.SUBSCRIPTION.Q`. Os dados do usuário que estão associados à assinatura durável são então incluídos como uma propriedade na mensagem, que é armazenada na pasta MQPS do cabeçalho RFH2 com a chave Sud.

O aplicativo IBM WebSphere MQ classes for JMS pode chamar:

```
javax.jms.Message.getStringProperty(JmsConstants.JMS_IBM_SUBSCRIPTION_USER_DATA);
```

A Sequência a seguir é então retornada:

```
Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q
```

.

Conceitos relacionados

“O cabeçalho MQRFH2 e o JMS” na página 823

Tarefas relacionadas

Definindo uma assinatura administrativa

Referências relacionadas

DEFINE SUB

Interface JmsConstants

Fechando um WebSphere MQ classes para aplicativo JMS

É importante para um WebSphere MQ classes para o aplicativo JMS para fechar determinados objetos JMS explicitamente antes de parar. Finalizadores não podem ser chamados, portanto, não dependem deles para liberar recursos. Não permita que um aplicativo finalize com o rastreio compactado ativo.

A coleta de lixo sozinha não pode liberar todas as classes do WebSphere MQ para os recursos JMS e WebSphere MQ em tempo hábil, especialmente se um aplicativo criar muitos objetos JMS de curta duração no nível de sessão ou inferior É, portanto, importante que um aplicativo feche um objeto Conexão, Sessão, MessageConsumer ou MessageProducer quando ele não for mais necessário.

Se um aplicativo finalizar sem fechar uma Conexão, uma reversão implícita ocorre para todas as sessões transacionadas da conexão. Para assegurar que quaisquer mudanças feitas pelo aplicativo sejam confirmadas, feche a Conexão explicitamente antes de fechar o aplicativo.

Não use finalizadores em um aplicativo para fechar os objetos JMS Como os finalizadores podem não ser chamados, os recursos podem não ser liberados. Quando uma Conexão é fechada, ela fecha todas as Sessões que foram criadas a partir dela. Da mesma forma, MessageConsumers e MessageProducers

criados a partir de uma Sessão são fechados quando a Sessão é fechada. No entanto, considere o fechamento de Sessões, MessageConsumers e MessageProducers explicitamente para assegurar que os recursos sejam liberados de uma maneira oportuna.

Se a compactação de rastreio estiver ativada, encerramentos System.Halt() e términos JVM não controlados e anormais provavelmente resultarão em um arquivo de rastreio corrompido. Sempre que possível, desative o recurso de rastreio quando você tiver coletado as informações de rastreio de que você precisa. Se você estiver rastreando um aplicativo até uma finalização anormal, use a saída de rastreio descompactada.

Manipulando mensagens suspeitas no IBM WebSphere MQ classes for JMS

Uma mensagem suspeita é aquela que não pode ser processada por um aplicativo MDB de recebimento. Se uma mensagem suspeita for encontrada, os objetos JMS MessageConsumer e ConnectionConsumer poderão enfileirá-la novamente de acordo com duas propriedades da fila, BOQNAME, e BOTHRESH

Às vezes, uma mensagem mal formatada incorretamente chega em uma fila. Nesse contexto, mal formatada significa que o aplicativo de recebimento não pode processar a mensagem corretamente. Essa mensagem pode fazer com que o aplicativo de recebimento falhe e restaure essa mensagem mal formatada. A mensagem pode então ser entregue repetidamente à fila de entrada e recuperada repetidamente pelo aplicativo. Essas mensagens são conhecidas como *mensagens suspeitas*. O objeto JMS MessageConsumer detecta mensagens suspeitas e roteia novamente para um destino alternativo.

O gerenciador de filas do IBM WebSphere MQ mantém um registro do número de vezes que cada mensagem foi restaurada. Quando esse número atinge um valor limite configurável, o consumidor de mensagem recoloca a mensagem em uma fila de restauração denominada. Se esse novo enfileiramento falhar por qualquer razão, a mensagem será removida da fila de entrada e um enfileirada novamente na fila de mensagens não entregues ou descartada. Consulte “Removendo mensagens da fila em ASF” na página 941 para obter mais detalhes.

Há uma diferença entre a maneira na qual as mensagens suspeitas são enfileiradas novamente por MessageConsumers e ConnectionConsumers. ConnectionConsumers são capazes de enfileirar mensagens suspeitas novamente sem afetar a entrega das mensagens. O processo de novo enfileiramento ocorre fora de qualquer unidade de trabalho associada à entrega da mensagem real para o código do aplicativo. Isso é possível devido à natureza multiencadeada da operação do ConnectionConsumer.

MessageConsumers, no entanto, são de encadeamento único abaixo do nível de Sessão e qualquer novo enfileiramento de mensagens suspeitas ocorre dentro da unidade de trabalho atual. Isso não afeta a operação do aplicativo, no entanto, quando as mensagens suspeitas são enfileiradas novamente sob uma Sessão transacionada ou de Client_acknowledge, a ação de novo enfileiramento em si não será confirmada até a unidade de trabalho atual ser confirmada pelo código do aplicativo ou, se apropriado, peço código do contêiner do aplicativo.

Os objetos JMS ConnectionConsumer manipulam mensagens suspeitas da mesma maneira e usando as mesmas propriedades da fila.. Se diversos consumidores de conexão estiverem monitorando a mesma fila, é possível que a mensagem suspeita possa ser entregue a um aplicativo mais vezes do que o valor limite antes que o novo enfileiramento ocorra. Este comportamento ocorre devido à maneira como consumidores de conexões individuais monitoram filas e enfileiram mensagens suspeitas novamente.

O valor do limite e o nome da fila de restauração são atributos de uma fila do IBM WebSphere MQ. Os nomes dos atributos são BackoutThreshold e BackoutRequeueQName. A fila à qual eles se aplicam é a seguinte:

- Para o sistema de mensagens ponto a ponto, é a fila local subjacente. Isso é importante quando os consumidores de mensagens e os consumidores de conexão usam aliases de filas.
- Para o sistema de mensagens de publicar/assinar no modo normal do provedor do sistema de mensagens do IBM WebSphere MQ, a fila gerenciada de Topic será criada a partir da fila modelo.
- Para o sistema de mensagens de publicar/assinar no modo de migração do provedor do sistema de mensagens do IBM WebSphere MQ, é a fila CCSUB definida no objeto TopicConnectionFactory ou a fila CCDSUB definida no objeto Topic.

O IBM WebSphere MQ classes for JMS consulta BackoutThreshold e BackoutRequeueQName da fila. Deve-se, portanto, conceder acesso de consulta na fila ao usuário que está executando o aplicativo.

V 7.5.0.9 Se a fila de destino for uma fila de clusters, as autoridades necessárias dependerão da versão do IBM WebSphere MQ classes for JMS que está sendo usado:

- Ao usar o IBM WebSphere MQ classes for JMS para Version 7.5.0, Fix Pack 9 mais uma correção temporária para o APAR IT26482, o acesso de consulta é necessário.
- Para todas as outras versões, conceda consulta, navegue e obtenha acesso.

Para configurar os atributos BackoutThreshold e BackoutRequeueQName, emita o comando MQSC a seguir:

```
ALTER QLOCAL(your.queue.name) BOTHRESH(threshold value) BOQNAME(your.backout.queue.name)
```

Se o atributo BackoutThreshold for configurado para um valor diferente de zero, para evitar um comportamento inesperado, configure o atributo BackoutRequeueQName para um nome de fila válido.

Para o sistema de mensagens de publicação / assinatura, se o seu sistema criar uma fila dinâmica para cada assinatura, esses valores de atributo serão obtidos da fila modelo IBM WebSphere MQ classes for JMS , SYSTEM.JMS.MODEL.QUEUE. Para alterar essas configurações, use:

```
ALTER QMODEL(SYSTEM.JMS.MODEL.QUEUE) BOTHRESH(threshold value) BOQNAME(your.backout.queue.name)
```

Se o valor do limite de restauração for zero, a manipulação de mensagens suspeitas será desativada e as mensagens suspeitas permanecerão na fila de entrada. Caso contrário, quando a contagem de restaurações atingir o valor do limite, a mensagem será enviada para a fila de restauração denominada. Se a contagem de restaurações atingir o valor do limite, mas a mensagem não puder ir para a fila de restauração, a mensagem será enviada para a fila de mensagens não entregues ou será descartada. Essa situação ocorre se a fila de restauração não estiver definida ou se o objeto MessageConsumer não puder enviar a mensagem para a fila de restauração. Consulte “Removendo mensagens da fila em ASF” na página 941 para obter detalhes adicionais.

Quando uma mensagem é enfileirada novamente na fila de reenfileiramento de restauração, alguns dos valores do campo no Descritor de mensagens (MQMD) da mensagem mudam. Consulte MQMD - Descritor de mensagens para obter detalhes sobre o formato do MQMD.

Os campos MQMD a seguir mudam o valor quando a mensagem vai para a fila de restauração.

- PutDate é atualizado para a data em que ele vai para a fila de reenfileiramento de restauração.
- PutTime é atualizado para o horário em que ele vai para a fila de reenfileiramento de restauração.
- A contagem de backout é reconfigurada para zero.
- A expiração da mensagem é atualizada para refletir a expiração restante no momento em que a mensagem original foi recebida pelo aplicativo JMS.

Os valores nos campos a seguir permanecem os mesmos quando a mensagem vai para a fila de restauração:

- StructId
- Versão
- Relatório
- MessageType
- Feedback
- Encoding
- CodedCharSetId
- MsgId
- CorrelId
- ReplyToQ

- ReplyToQMgr
- Formato
- Persistence
- Priority

Exceções em IBM WebSphere MQ classes for JMS

Um aplicativo IBM WebSphere MQ classes for JMS deve ser capaz de manipular exceções que são lançadas por uma API JMS chamadas ou entregues para um manipulador de exceções

IBM WebSphere MQ classes for JMS relata problemas de tempo de execução lançando as exceções. JMSException é a classe-raiz para exceções lançadas por métodos JMS, e capturar exceções JMSException fornece uma maneira genérica de manipular todas as exceções relacionadas ao JMS.

Cada exceção JMSException encapsula as seguintes informações:

- Uma mensagem de exceção específica do provedor, que um aplicativo obtém chamando o método Throwable.getMessage().
- Um código de erro específico do provedor, que um aplicativo obtém chamando o método JMSException.getErrorCode().
- Uma exceção vinculada. Uma exceção lançada por uma chamada de API JMS é geralmente o resultado de um problema de nível inferior, que é relatado por outra exceção vinculada a essa exceção Um aplicativo obtém uma exceção vinculada chamando o método JMSException.getLinkedException() ou Throwable.getCause().

A maioria das exceções lançadas pelo IBM WebSphere MQ classes for JMS são instâncias de subclasses do JMSException. Essas subclasses implementam a interface com.ibm.msg.client.jms.JmsExceptionDetail, que fornece as seguintes informações adicionais:

- Uma explicação da mensagem de exceção, que um aplicativo obtém chamando o método JmsExceptionDetail.getExplanation().
- Uma resposta do usuário recomendada para a exceção, que um aplicativo obtém chamando o método JmsExceptionDetail.getUserAction().
- As chaves para as inserções de mensagem na mensagem de exceção. Um aplicativo obtém um agente iterativo para todas as chaves, chamando o método JmsExceptionDetail.getKeys().
- As inserções de mensagem na mensagem de exceção. Por exemplo, uma inserção de mensagem pode ser o nome da fila que causou a exceção e pode ser útil para que um aplicativo consiga acessar esse nome. Um aplicativo obtém a inserção de mensagem correspondente a uma chave especificada, chamando o método JmsExceptionDetail.getValue().

Todos os métodos na interface JmsExceptionDetail podem retornar nulo, se nenhum detalhe estiver disponível.

Por exemplo, se um aplicativo tentar criar um produtor de mensagem para uma fila do IBM WebSphere MQ que não existe, uma exceção será lançada com as informações a seguir:

```
Message : JMSWMQ2008: Failed to open MQ queue 'Q_test'.
Class : class com.ibm.msg.client.jms.DetailedInvalidDestinationException
Error Code : JMSWMQ2008
Explanation : JMS attempted to perform an MQOPEN, but WebSphere MQ reported an
              error.
User Action : Use the linked exception to determine the cause of this error. Check
              that the specified queue and queue manager are defined correctly.
```

A exceção lançada, com.ibm.msg.client.jms.DetailedInvalidDestinationException, é uma subclasse do javax.jms.InvalidDestinationException e implementa a interface com.ibm.msg.client.jms.JmsExceptionDetail.

Exceções Vinculadas

Uma exceção vinculada fornece informações adicionais sobre um problema de tempo de execução. Portanto, para cada exceção JMSException que é lançada, um aplicativo deve verificar a exceção

vinculada. A exceção vinculada sozinha pode ter outra exceção vinculada e assim as exceções vinculadas formam uma cadeia levando de volta ao problema subjacente original. Uma exceção vinculada é implementada usando o mecanismo de exceção em cadeia da classe java.lang.Throwable e um aplicativo obtém uma exceção vinculada chamando o método Throwable.getCause(). Para uma exceção JMSException, o método getLinkedException() delega na realidade ao método Throwable.getCause().

Por exemplo, se um aplicativo especificar um número de porta incorreto ao se conectar a um gerenciador de filas, as exceções formarão a seguinte cadeia:

```
com.ibm.msg.client.jms.DetailIllegalStateException
   |
   +--->com.ibm.mq.MQException
           |
           +--->com.ibm.mq.jmqi.JmqiException
                   |
                   +--->java.net.ConnectionException
```

Geralmente, cada exceção em uma cadeia é lançada a partir de uma camada diferente no código. Por exemplo, as exceções na cadeia anteriormente são lançadas pelas seguintes camadas:

- A primeira exceção, uma instância de uma subclasse de JMSException, é lançada pela camada comum em IBM WebSphere MQ classes for JMS.
- A próxima exceção, uma instância de com.ibm.mq.MQException, é lançada pelo provedor de sistema de mensagens do IBM WebSphere MQ.
- A próxima exceção, uma instância de com.ibm.mq.jmqi.JmqiException, é lançada pela interface Java comum para o MQI
- A exceção final, uma instância de java.net.ConnectionException, é lançada pela biblioteca de classes Java

Para obter mais informações sobre a arquitetura em camadas do IBM WebSphere MQ classes for JMS, consulte “Classes IBM WebSphere MQ para arquitetura JMS” na página 809

Usando o código semelhante ao seguinte código, um aplicativo pode iterar por meio desta cadeia para extrair todas as informações apropriadas:

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import javax.jms.JMSException;
.
.
.
catch (JMSException je) {
    System.err.println("Caught JMSException");

    // Check for linked exceptions in JMSException
    Throwable t = je;
    while (t != null) {
        // Write out the message that is applicable to all exceptions
        System.err.println("Exception Msg: " + t.getMessage());
        // Write out the exception stack trace
        t.printStackTrace(System.err);

        // Add on specific information depending on the type of exception
        if (t instanceof JMSException) {
            JMSException je1 = (JMSException) t;
            System.err.println("JMS Error code: " + je1.getErrorCode());

            if (t instanceof JmsExceptionDetail){
                JmsExceptionDetail jed = (JmsExceptionDetail)je1;
                System.err.println("JMS Explanation: " + jed.getExplanation());
                System.err.println("JMS Explanation: " + jed.getUserAction());
            }
        } else if (t instanceof MQException) {
            MQException mqe = (MQException) t;
            System.err.println("WMQ Completion code: " + mqe.getCompCode());
            System.err.println("WMQ Reason code: " + mqe.getReason());
        } else if (t instanceof JmqiException){
            JmqiException jmqie = (JmqiException)t;
            System.err.println("WMQ Log Message: " + jmqie.getWmqLogMessage());
            System.err.println("WMQ Explanation: " + jmqie.getWmqMsgExplanation());
            System.err.println("WMQ Msg Summary: " + jmqie.getWmqMsgSummary());
```

```
            System.err.println("WMQ Msg User Response: "
                               + jmqie.getWmqMsgUserResponse());
            System.err.println("WMQ Msg Severity: " + jmqie.getWmqMsgSeverity());
        }

        // Get the next cause
        t = t.getCause();
    }
}
```

Observe que um aplicativo sempre deve verificar o tipo de cada exceção em uma cadeia porque o tipo da exceção pode variar e as exceções dos diferentes tipos encapsulam diferentes informações.

Obtendo informações específicas do IBM WebSphere MQ sobre um problema

As instâncias de com.ibm.mq.MQException e com.ibm.mq.jmqi.JmqiException contêm informações específicas do IBM WebSphere MQ sobre um problema.

Uma exceção MQException encapsula as seguintes informações:

- Um código de conclusão, que um aplicativo obtém chamando o método getCompCode()
- Um código de razão, que um aplicativo obtém chamando o método getReason()

Uma exceção JmqiException também encapsula um código de conclusão e um código de razão. Além disso, no entanto, uma exceção JmqiException encapsula as informações em uma mensagem AMQ*nnnn* ou CSQ*nnnn* , se uma estiver associada à exceção. Ao chamar os métodos apropriados da exceção, um aplicativo pode obter diversos componentes desta mensagem, como a gravidade, a explicação e a resposta do usuário.

Para obter exemplos de como usar os métodos mencionados nesta seção, consulte o código de amostra em “Exceções Vinculadas” na página 903.

Fazendo upgrade de versões anteriores do IBM WebSphere MQ classes for JMS

Em comparação com versões anteriores do IBM WebSphere MQ classes for JMS, a maioria dos códigos de erro e mensagens de exceção foram alterados na Versão 7. O motivo dessas mudanças é que o IBM WebSphere MQ classes for JMS agora tem uma arquitetura em camadas e as exceções são lançadas de diferentes camadas no código.

Por exemplo, se um aplicativo tenta se conectar a um gerenciador de filas que não existe, uma versão anterior do IBM WebSphere MQ classes for JMS emitiu uma exceção JMSException com as informações a seguir:

```
MQJMS2005: Failed to create MQQueueManager for 'localhost:QM_test'.
```

Esta exceção continha uma exceção MQException vinculada às seguintes informações:

```
MQJE001: Completion Code 2, Reason 2058
```

Por comparação nas mesmas circunstâncias, a Versão 7 do IBM WebSphere MQ classes for JMS lança uma exceção JMSException com as seguintes informações:

```
Message : JMSWMQ0018: Failed to connect to queue manager 'QM_test' with
          connection mode 'Client' and host name 'localhost'.
Class : class com.ibm.msg.client.jms.DetailedJMSException
Error Code : JMSWMQ0018
Explanation : null
User Action : Check the queue manager is started and if running in client mode,
              check there is a listener running. Please see the linked exception
              for more information.
```

Esta exceção contém uma exceção MQException vinculada às seguintes informações:

```
Message : JMSCMQ0001: WebSphere MQ call failed with compcode '2' ('MQCC_FAILED')
          reason '2058' ('MQRC_Q_MGR_NAME_ERROR').
Class : class com.ibm.mq.MQException
```

```
Completion Code : 2
Reason Code : 2058
```

Se seu aplicativo analisar ou testar mensagens de exceção retornadas pelo método Throwable.getMessage() ou códigos de erro retornados pelo método JMSException.getErrorCode() e você estiver fazendo upgrade de uma liberação anterior à Versão 7, seu aplicativo provavelmente precisará ser modificado para usar a Versão 7 de IBM WebSphere MQ classes for JMS.

Listeners de Exceção

Um aplicativo pode registrar um listener de exceção com um objeto de Conexão. Subsequentemente, se ocorrer um problema que transforma a conexão inutilizável, o IBM WebSphere MQ classes for JMS entregará uma exceção ao listener de exceção chamando seu método onException(). O aplicativo então tem a oportunidade de restabelecer a conexão.

V 7.5.0.8 O APAR IT14820, incluído por meio do IBM WebSphere MQ Version 7.5.0, Fix Pack 8, corrigiu um defeito no qual o ExceptionListener do JMS de um aplicativo não seria chamado para exceções não relativas à conexão quebrada (por exemplo, MQRC_GET_INHIBITED), mesmo que a propriedade ASYNC_EXCEPTIONS no Connection Factory do JMS usado pelo aplicativo fosse configurada como ASYNC_EXCEPTIONS_ALL. Esse era o valor padrão antes da Version 7.5.0, Fix Pack 8.

V 7.5.0.8 Para manter o comportamento para os aplicativos JMS atuais que configuram um MessageListener JMS e um ExceptionListenerJMS e para assegurar que o IBM WebSphere MQ classes for JMS esteja consistente com a especificação JMS, o valor padrão para a propriedade ConnectionFactory JMS de ASYNC_EXCEPTIONS foi alterado para ASYNC_EXCEPTIONS_CONNECTIONBROKEN para IBM WebSphere MQ classes for JMS. Como resultado, por padrão, somente exceções correspondentes a códigos de erro de conexão quebrada são entregues ao JMS ExceptionListener de um aplicativo.

V 7.5.0.8 Na Version 7.5.0, Fix Pack 8, as IBM WebSphere MQ classes for JMS também foram atualizadas de forma que os JMSExceptions relacionados a erros não relativos à conexão quebrada, os quais ocorrem durante a entrega de mensagem para consumidores de mensagem assíncrona, ainda serão entregues a um ExceptionListener registrado quando o ConnectionFactory do JMS usado pelo aplicativo tiver a propriedade ASYNC_EXCEPTIONS configurada para o valor ASYNC_EXCEPTIONS_ALL.

V 7.5.0.8 Para obter mais informações sobre o que foi alterado para listeners de exceção para Version 7.5.0, Fix Pack 8 e por que as mudanças foram feitas a partir de liberações anteriores, consulte JMS: Mudanças do listener de exceção na Versão 7.5.

Para qualquer outro tipo de problema, uma exceção JMSException é lançada pela chamada da API JMS atual

Se um aplicativo não registrar um listener de exceção com um objeto Connection, quaisquer exceções que teriam sido entregues ao listener de exceção serão gravadas no log IBM WebSphere MQ classes for JMS .

Referências relacionadas

ASYNCEXCEPTION

Registrando erros nas classes do WebSphere MQ para JMS

As informações sobre problemas de tempo de execução que podem requerer ação corretiva pelo usuário são gravadas nas classes WebSphere MQ para log JMS.

Por exemplo, se um aplicativo tentar configurar uma propriedade de um connection factory, mas o nome da propriedade não for reconhecido; as classes do WebSphere MQ para JMS gravam informações sobre o problema em seu registro.

Por padrão, o arquivo que contém as criações de log é chamado mqjms.log e está no diretório de trabalho atual. No entanto, é possível alterar o nome e o local do arquivo de log configurando a propriedade com.ibm.msg.client.commonservices.log.outputName nas classes WebSphere MQ para o arquivo de configuração JMS. Para obter informações sobre as classes do WebSphere MQ para o arquivo de configuração JMS, consulte “O arquivo de configuração IBM WebSphere MQ classes

for JMS” na página 735e para obter mais detalhes sobre valores válidos para a propriedade com.ibm.msg.client.commonservices.log.outputName , consulte “Criação de log eIBM WebSphere MQ classes for JMS” na página 806.

Primeira tecnologia de suporte de falha (FFST) em WebSphere MQ classes para JMS

Se ocorrer um erro interno grave nas classes do WebSphere MQ para JMS, as informações de tecnologia de suporte de primeira falha (FFST) serão geradas..

As informações de FFST são gravadas em um arquivo chamado JMSC*nnnn*.FDC, em que *nnnn* é um número de quatro dígitos. Esse arquivo está em um diretório chamado FFDC, que é um subdiretório do diretório no qual a saída de rastreio é gravada Por padrão, a saída de rastreio é gravada no diretório ativo atual, mas é possível redirecionar a saída de rastreio para um diretório diferente configurando a propriedade **com.ibm.msg.client.commonservices.trace.outputName** nas classes WebSphere MQ para o arquivo de configuração JMS. Para obter informações sobre as classes WebSphere MQ para o arquivo de configuração JMS, consulte “O arquivo de configuração IBM WebSphere MQ classes for JMS” na página 735.

Se o rastreio for ativado quando as informações FFST forem geradas, as informações FFST também serão gravadas no arquivo de rastreio. Para obter mais informações sobre como rastrear programas JMS, consulte Rastreio IBM WebSphere MQ classes for JMS aplicativos.

Para suprimir a produção de arquivos FFDC, configure a propriedade **com.ibm.msg.client.commonservices.ffst.suppress**, conforme a seguir:

0
Saída de todos os arquivos FFDC (padrão)

-1
Emita apenas os primeiros arquivos FFDC de um tipo específico

integer
Suprimir todos os arquivos FFDC, exceto aqueles que são múltiplos deste número.

Acessando recursos do WebSphere MQ a partir de classes do WebSphere MQ para o aplicativo JMS

As classes do WebSphere MQ para JMS fornecem recursos para explorar vários recursos do WebSphere MQ.

Atenção: Esses recursos estão fora da especificação JMS ou, em certos casos, violam a especificação JMS Se você os usar, seu aplicativo não será compatível com outros provedores JMS. Os recursos não compatíveis com a especificação JMS são rotulados com um aviso de Atenção.

Lendo e Gravando o Descritor de Mensagens a partir de Classes WebSphere MQ para Aplicativo JMS

Controle a capacidade de acessar o descritor de mensagens (MQMD) configurando as propriedades em um Destino e uma Mensagem.

Alguns aplicativos WebSphere MQ requerem que valores específicos sejam configurados no MQMD de mensagens enviadas para eles.. As classes do WebSphere MQ para JMS fornecem atributos de mensagem que permitem que os aplicativos JMS configurem campos MQMD e, portanto, ativem aplicativos JMS para "drive" WebSphere MQ aplicativos.

Deve-se configurar a propriedade do objeto Destination WMQ_MQMD_WRITE_ENABLED para true para que a configuração de propriedades do MQMD tenham qualquer efeito. Então é possível usar os métodos de configuração de propriedades da mensagem (por exemplo, setStringProperty) para designar valores para os campos MQMD. Todos os campos MQMD são expostos, exceto StrucId e Version; BackoutCount pode ser lido, mas não gravado.

Este exemplo resulta em uma mensagem sendo colocada em uma fila ou em um tópico com MQMD.UserIdentifier configurado como "JoeBloggs".

```
// Create a ConnectionFactory, connection, session, producer, message
// ...

// Create a destination
// ...

// Enable MQMD write
dest.setBooleanProperty(WMQConstants.WMQ_MQMD_WRITE_ENABLED, true);

// Optionally, set a message context if applicable for this MD field
dest.setIntProperty(WMQConstants.WMQ_MQMD_MESSAGE_CONTEXT,
  WMQConstants.WMQ_MDCTX_SET_IDENTITY_CONTEXT);

// On the message, set property to provide custom UserId
msg.setStringProperty("JMS_IBM_MQMD_UserIdentifier", "JoeBloggs");

// Send the message
// ...
```

É necessário configurar WMQ_MQMD_MESSAGE_CONTEXT antes de configurar JMS_IBM_MQMD_UserIdentifier. Para obter mais informações sobre o uso do WMQ_MQMD_MESSAGE_CONTEXT, consulte “Propriedades do objeto de mensagem JMS.” na página 910.

De forma semelhante, será possível extrair o conteúdo dos campos MQMD configurando WMQ_MQMD_READ_ENABLED como true antes de receber uma mensagem e, em seguida, usar os métodos get da mensagem, como getStringProperty. As propriedades recebidas são somente leitura.

Este exemplo resulta no campo *value* que mantém o valor do campo MQMD.ApplIdentityData de uma mensagem obtida a partir de uma fila ou de um tópico.

```
// Create a ConnectionFactory, connection, session, consumer
// ...

// Create a destination
// ...

// Enable MQMD read
dest.setBooleanProperty(WMQConstants.WMQ_MQMD_READ_ENABLED, true);

// Receive a message
// ...

// Get MQMD field value using a property
String value = rcvMsg.getStringProperty("JMS_IBM_MQMD_ApplIdentityData");
```

Propriedades do objeto de destino JMS
Duas propriedades do objeto de Destino controlam o acesso ao MQMD do JMS e um terceiro controla o contexto da mensagem.

Tabela 124. Nomes e descrições das propriedades

Propriedade	**Formato curto**	**Descrição**
WMQ_MQMD_WRITE_ENABLED	MDW	Se um aplicativo JMS pode configurar os valores de campos MQMD
WMQ_MQMD_READ_ENABLED	MDR	Se um aplicativo JMS pode extrair os valores de campos MQMD
WMQ_MQMD_MESSAGE _CONTEXT	MDCTX	Qual nível de contexto de mensagens deve ser configurado por um aplicativo JMS O aplicativo deve estar em execução com autoridade de contexto apropriada para que essa propriedade entre em vigor

Tabela 125. Nomes de propriedades, valores e métodos configurados

Propriedade	Valores válidos na ferramenta de administração (padrões em negrito)	Valores válidos em programas	Método configurado
WMQ_MQMD_WRITE_ENABLED	• **NÃO** Todas as propriedades JMS_IBM_MQMD* são ignoradas e seus valores não são copiados para a estrutura MQMD subjacente. • SIM As propriedades JMS_IBM_MQMD* são processadas. Seus valores serão copiados para a estrutura do MQMD subjacente.	• **False** • True	setMQMDWriteEnabled
WMQ_MQMD_READ_ENABLED	• **NÃO** Ao enviar mensagens, as propriedades JMS_IBM_MQMD* em uma mensagem enviada não são atualizadas para refletir os valores de campos atualizados no MQMD. Ao receber mensagens, nenhuma das propriedades JMS_IBM_MQMD* estarão disponíveis em uma mensagem recebida, mesmo que o emissor tenha configurado algumas ou todas elas. • SIM Ao enviar mensagens, todas as propriedades JMS_IBM_MQMD* em uma mensagem enviada são atualizadas para refletir os valores de campos atualizados no MQMD, incluindo aquelas que o emissor não configurou explicitamente. Ao receber mensagens, todas as propriedades JMS_IBM_MQMD* estão disponíveis em uma mensagem recebida, incluindo aquelas que o emissor não configurou explicitamente.	• **False** • True	setMQMDReadEnabled

Tabela 125. Nomes de propriedades, valores e métodos configurados (continuação)			
Propriedade	**Valores válidos na ferramenta de administração (padrões em negrito)**	**Valores válidos em programas**	**Método configurado**
WMQ_MQMD _MESSAGE_CONTEXT	• **Default** A chamada da API MQOPEN e a estrutura de MQPMO não especificam nenhuma opção de contexto de mensagem explícita • SET_IDENTITY_CONTEXT A chamada da API MQOPEN especifica a opção de contexto de mensagem MQOO_SET_IDENTITY_CONTEXT e a estrutura de MQPMO especifica MQPMO_SET_IDENTITY_CONTEXT • SET_ALL_CONTEXT A chamada da API MQOPEN especifica a opção de contexto de mensagem MQOO_SET_ALL_CONTEXT e a estrutura de MQPMO especifica MQPMO_SET_ALL_CONTEXT	• **WMQ_MDCTX_DEFAULT** • WMQ_MDCTX_SET_IDENTITY_CONTEXT • WMQ_MDCTX_SET_ALL_CONTEXT	setMQMDMessageContext

Propriedades do objeto de mensagem JMS.
Propriedades do objeto de mensagem prefixadas com JMS_IBM_MQMD permitem definir ou ler o campo MQMD correspondente.

Enviando mensagens

Todos os campos MQMD, exceto StrucId e Version, são representados. Essas propriedades referem-se apenas aos campos MQMD; quando uma propriedade ocorre tanto no MQMD quanto no cabeçalho MQRFH2, a versão no MQRFH2 não é configurada nem extraída.

Qualquer uma dessas propriedades pode ser configurada, exceto JMS_IBM_MQMD_BackoutCount. Qualquer valor configurado para JMS_IBM_MQMD_BackoutCount é ignorado.

Se uma propriedade tiver um comprimento máximo e você fornecer um valor que é muito longo, o valor será truncado.

Para determinadas propriedades, também deve-se definir a propriedade WMQ_MQMD_MESSAGE_CONTEXT no objeto de Destino. O aplicativo deve estar em execução com autoridade de contexto apropriado para esta propriedade entrar em vigor. Se você não configurar WMQ_MQMD_MESSAGE_CONTEXT para um valor apropriado, o valor da propriedade será ignorado. Se você configura WMQ_MQMD_MESSAGE_CONTEXT para um valor apropriado, mas você não tem autoridade de contexto suficiente para o gerenciador de filas, uma JMSException é emitida. Propriedades que requerem valores específicos de WMQ_MQMD_MESSAGE_CONTEXT são conforme a seguir.

As propriedades a seguir requerem que WMQ_MQMD_MESSAGE_CONTEXT seja configurado para WMQ_MDCTX_SET_IDENTITY_CONTEXT ou WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- JMS_IBM_MQMD_ApplIdentityData

As propriedades a seguir requerem WMQ_MQMD_MESSAGE_CONTEXT seja configurado para WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_PutApplType
- JMS_IBM_MQMD_PutApplName
- JMS_IBM_MQMD_PutDate
- JMS_IBM_MQMD_PutTime
- JMS_IBM_MQMD_ApplOriginData

Como receber mensagens

Todas estas propriedades estão disponíveis em uma mensagem recebida se a propriedade WMQ_MQMD_READ_ENABLED estiver configurada como true, não importa quais propriedades reais estão definidas para o aplicativo de produção. Um aplicativo não pode modificar as propriedades de uma mensagem recebida a menos que todas as propriedades sejam limpas primeiro, de acordo com a especificação JMS. A mensagem recebida pode ser transmitida sem modificar as propriedades.

Atenção: Se o seu aplicativo recebe uma mensagem de um destino com a propriedade WMQ_MQMD_READ_ENABLED definida como true, e a encaminha para um destino com WMQ_MQMD_WRITE_ENABLED configurado como true, isto resulta em todos os valores de campo MQMD da mensagem recebida copiados na mensagem encaminhada.

Tabela de propriedades

Esta tabela lista as propriedades do objeto de mensagem que representa os campos MQMD. Consulte os links para obter descrições completas dos campos e seus valores permitidos.

Tabela 126. Propriedade nomes, descrições e tipos

Propriedade	Descrição	Tipo Java	Link para descrição completa
JMS_IBM_MQMD_Report	Opções para as mensagens de relatório	Integer	Report
JMS_IBM_MQMD_MsgType	Tipo de Mensagem	Integer	MsgType
JMS_IBM_MQMD_Expiry	Tempo de vida da mensagem	Integer	Expiração
JMS_IBM_MQMD_Feedback	Feedback ou código de razão	Integer	Feedback
JMS_IBM_MQMD_Encoding	Codificação numérica de dados da mensagem	Integer	Encoding
JMS_IBM_MQMD_CodedCharSetId	Identificador do conjunto de caracteres de dados da mensagem	Integer	CodedCharSetId
JMS_IBM_MQMD_Format	Nome do formato dos dados da mensagem	Sequência	Format
JMS_IBM_MQMD_Priority [1]	Prioridade da mensagem	Integer	Prioridade
JMS_IBM_MQMD_Persistence	Persistência de mensagem	Integer	Persistência
JMS_IBM_MQMD_MsgId [2]	ID da Mensagem	Objeto (byte[]) [4]	MsgId
JMS_IBM_MQMD_CorrelId [3]	Identificador de correlação	Objeto (byte[]) [4]	CorrelId
JMS_IBM_MQMD_BackoutCount	contador de backout	Integer	BackoutCount

Tabela 126. Propriedade nomes, descrições e tipos (continuação)

Propriedade	Descrição	Tipo Java	Link para descrição completa
JMS_IBM_MQMD_ReplyToQ	Nome da fila de resposta	Sequência	ReplyToQ
JMS_IBM_MQMD_ReplyToQMgr	Nome do gerenciador de filas de resposta	Sequência	ReplyToQMgr
JMS_IBM_MQMD_UserIdentifier	Identificador de usuário	Sequência	UserIdentifier
JMS_IBM_MQMD_AccountingToken	Símbolo de contabilidade	Objeto (byte[]) [4]	AccountingToken
JMS_IBM_MQMD_ApplIdentityData	dados do aplicativo relacionados à identidade	Sequência	ApplIdentityData
JMS_IBM_MQMD_PutApplType	Tipo de aplicativo que coloca a mensagem	Integer	PutApplType
JMS_IBM_MQMD_PutApplName	Nome do aplicativo que coloca a mensagem	Sequência	PutApplName
JMS_IBM_MQMD_PutDate	Data quando a mensagem foi colocada	Sequência	PutDate
JMS_IBM_MQMD_PutTime	Hora quando a mensagem foi colocada	Sequência	PutTime
JMS_IBM_MQMD_ApplOriginData	Os dados do aplicativo relacionados à origem	Sequência	ApplOriginData
JMS_IBM_MQMD_GroupId	Identificador de grupo	Objeto (byte[]) [4]	GroupId
JMS_IBM_MQMD_MsgSeqNumber	Número de sequência de mensagem lógica dentro do grupo	Integer	MsgSeqNumber
JMS_IBM_MQMD_Offset	Deslocamento dos dados na mensagem física a partir do início da mensagem lógica	Integer	Offset
JMS_IBM_MQMD_MsgFlags	Sinalizadores de mensagem	Integer	MsgFlags
JMS_IBM_MQMD_OriginalLength	Comprimento da mensagem original	Integer	OriginalLength

1. **Atenção:** Se você designar um valor para JMS_IBM_MQMD_Priority que não esteja dentro do intervalo 0-9, isso violará a especificação JMS.
2. **Atenção:** A especificação JMS indica que o ID de mensagem deve ser configurado pelo provedor JMS e que deve ser exclusivo ou nulo. Se você designar um valor para JMS_IBM_MQMD_MsgId, esse valor será copiado para o JMSMessageID. Portanto, ele não é configurado pelo provedor JMS e pode não ser exclusivo: isso viola a especificação JMS.
3. **Atenção:** Se você designar um valor para JMS_IBM_MQMD_CorrelId que inicia com a sequência 'ID:', isso viola a especificação JMS.
4. **Atenção:** O uso de propriedades de matriz de bytes em uma mensagem viola a especificação JMS

Acessando IBM WebSphere MQ Dados da mensagem de um aplicativo usando classes WebSphere MQ para JMS

É possível acessar os dados da mensagem do WebSphere MQ completos dentro de um aplicativo usando classes IBM WebSphere MQ para JMS Para acessar todos os dados, a mensagem deve ser uma JMSBytesMessage. O corpo do JMSBytesMessage inclui qualquer cabeçalho MQRFH2, quaisquer outros cabeçalhos do IBM WebSphere MQ e os dados de mensagens a seguir.

Configure a propriedade WMQ_MESSAGE_BODY do destino como WMQ_MESSAGE_BODY_MQ para receber todos os dados do corpo da mensagem no JMSBytesMessage.

Se WMQ_MESSAGE_BODY for configurado como WMQ_MESSAGE_BODY_JMS ou WMQ_MESSAGE_BODY_UNSPECIFIED, o corpo da mensagem será retornado sem o cabeçalho MQRFH2 JMS e as propriedades do JMSBytesMessage refletem as propriedades configuradas no RFH2.

Alguns aplicativos não podem usar as funções descritas neste tópico. Se um aplicativo estiver conectado a um gerenciador de filas do WebSphere MQ V6 ou se ele tiver configurado PROVIDERVERSION como 6, as funções não estarão disponíveis

Enviando uma mensagem

Ao enviar mensagens a propriedade de destino, WMQ_MESSAGE_BODY, terá precedência sobre WMQ_TARGET_CLIENT.

Se WMQ_MESSAGE_BODY estiver configurado como WMQ_MESSAGE_BODY_JMS, as classes WebSphere MQ para JMS gerarão automaticamente um cabeçalho MQRFH2 com base nas configurações das propriedades e dos campos de cabeçalho JMSMessage .

Se WMQ_MESSAGE_BODY for configurado como WMQ_MESSAGE_BODY_MQ, nenhum cabeçalho adicional será incluído no corpo da mensagem

Se WMQ_MESSAGE_BODY estiver configurado como WMQ_MESSAGE_BODY_UNSPECIFIED, as classes WebSphere MQ para JMS enviam um cabeçalho MQRFH2 , a menos que WMQ_TARGET_CLIENT esteja configurado como WMQ_TARGET_DEST_MQ. No recebimento, configurando WMQ_TARGET_CLIENT como WMQ_TARGET_DEST_MQ resulta em nenhum MQRFH2 que está sendo removido do corpo da mensagem.

Nota: JMSBytesMessage e JMSTextMessage não requerem um MQRFH2, enquanto que o JMSStreamMessage, JMSMapMessage e JMSObjectMessage requerem.

WMQ_MESSAGE_BODY_UNSPECIFIED é a configuração padrão para WMQ_MESSAGE_BODY e WMQ_TARGET_DEST_JMS é a configuração padrão para WMQ_TARGET_CLIENT.

Se você enviar um JMSBytesMessage, poderá substituir as configurações padrão para o corpo da mensagem JMS quando a mensagem WebSphere MQ for construída. Use as seguintes propriedades:

- JMS_IBM_Format ou JMS_IBM_MQMD_Format: Esta propriedade especifica o formato do cabeçalho ou carga útil do aplicativo WebSphere MQ que inicia o corpo da mensagem JMS se não houver cabeçalho do Websphere MQ anterior.
- JMS_IBM_Character_Set ou JMS_IBM_MQMD_CodedCharSetId: Essa propriedade especifica o CCSID do cabeçalho WebSphere MQ ou da carga útil do aplicativo que inicia o corpo da mensagem JMS se não houver cabeçalho anterior do Websphere MQ .
- JMS_IBM_Encoding ou JMS_IBM_MQMD_Encoding: Esta propriedade especifica a codificação do cabeçalho WebSphere MQ ou carga útil do aplicativo que inicia o corpo da mensagem JMS se não houver cabeçalho anterior do Websphere MQ .

Se ambos os tipos de propriedade estão especificados, as propriedades JMS_IBM_MQMD_* substituem as propriedades JMS_IBM_* correspondentes, contanto que a propriedade de destino WMQ_MQMD_WRITE_ENABLED seja configurada como true.

As diferenças em vigor entre as propriedades de mensagem de configuração usando JMS_IBM_MQMD_* e JMS_IBM_* são significativas:

1. As propriedades JMS_IBM_MQMD_* são específicas do provedor JMS do IBM WebSphere MQ .

2. As propriedades JMS_IBM_MQMD_* são configuradas apenas no MQMD. As propriedades JMS_IBM_* serão configuradas no MQMD somente se a mensagem não tiver um cabeçalho JMS do MQRFH2 Caso contrário, eles serão configurados no cabeçalho RFH2 do JMS
3. As propriedades JMS_IBM_MQMD_* não tem efeito na codificação de texto e nos números gravados em um JMSMessage.

   Um aplicativo de recebimento que presume provavelmente os valores de MQMD.Encoding e MQMD.CodedCharSetId corresponde ao conjunto de codificação e caracteres de números e textos no corpo da mensagem. Se as propriedades JMS_IBM_MQMD_* forem usadas, será responsabilidade do aplicativo de envio fazer isso. A codificação e o conjunto de caracteres de números e texto no corpo da mensagem são configurados pelas propriedades JMS_IBM_*.

   O fragmento codificado de forma inválida no Figura 163 na página 914 envia uma mensagem codificada no conjunto de caracteres 1208, com MQMD.CodedCharSetId configurado para 37.

   a. Enviar mensagem codificada de forma errada

```
TextMessage tmo = session.createTextMessage();
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_MQ);
((MQDestination)destination).setMQMDWriteEnabled(true);
tmo.setIntProperty(WMQConstants.JMS_IBM_MQMD_CODEDCHARSETID, 37);
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 1208);
tmo.setText("String one");
producer.send(tmo);
```

   b. Recebendo a mensagem, contando com o valor de JMS_IBM_CHARACTER_SET configurado pelo valor de MQMD.CodedCharSetId:

```
TextMessage tmi = (TextMessage) cons.receive();
System.out.println("Message is \"" + tmi.getText() + "\"");
```

   c. Saída resultante:

```
Message is "ëÈÊ'>...??>?"
```

Figura 163. MQMD e dados de mensagens inconsistentemente codificados

Um dos snippets de código em Figura 164 na página 914 resulta em uma mensagem sendo colocada em uma fila ou em um tópico com seu corpo que contém a carga útil do aplicativo sem um cabeçalho MQRFH2 gerado automaticamente que está sendo incluído.

1. Configurando WMQ_MESSAGE_BODY_MQ:

```
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

2. Configurando WMQ_TARGET_DEST_MQ:

```
((MQDestination) destination).setMessageBodyStyle
                 (WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);
((MQDestination) destination).
                setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
```

Figura 164. Envie uma mensagem com um corpo da mensagem MQ.

Recebendo uma mensagem

Se WMQ_MESSAGE_BODY for configurado para WMQ_MESSAGE_BODY_JMS, o tipo e o corpo da mensagem JMS de entrada serão determinados pelo conteúdo da mensagem recebida do Websphere MQ O tipo e o corpo de mensagem serão determinados pelos campos no cabeçalho MQRFH2 ou no MQMD, se não houver nenhum MQRFH2.

Se WMQ_MESSAGE_BODY for configurado como WMQ_MESSAGE_BODY_MQ, o tipo de mensagem JMS de entrada será JMSBytesMessage O corpo da mensagem JMS são os dados da mensagem retornados pela chamada de API subjacente MQGET. O comprimento do corpo da mensagem é o comprimento retornado pela chamada MQGET. O conjunto de caracteres e a codificação dos dados no corpo da mensagem é determinado pelos campos CodedCharSetId e Encoding do MQMD. O formato dos dados no corpo da mensagem é determinado pelo campo Format do MQMD

Se WMQ_MESSAGE_BODY estiver configurado como WMQ_MESSAGE_BODY_UNSPECIFIED, o valor padrão IBM WebSphere MQ classes para JMS o configurará como WMQ_MESSAGE_BODY_JMS.

Ao receber um JMSBytesMessage, será possível decodificá-lo por referência às seguintes propriedades:

- JMS_IBM_Format ou JMS_IBM_MQMD_Format: Esta propriedade especifica o formato do cabeçalho ou carga útil do aplicativo WebSphere MQ que inicia o corpo da mensagem JMS se não houver cabeçalho do Websphere MQ anterior.
- JMS_IBM_Character_Set ou JMS_IBM_MQMD_CodedCharSetId: Essa propriedade especifica o CCSID do cabeçalho WebSphere MQ ou da carga útil do aplicativo que inicia o corpo da mensagem JMS se não houver cabeçalho anterior do Websphere MQ .
- JMS_IBM_Encoding ou JMS_IBM_MQMD_Encoding: Esta propriedade especifica a codificação do cabeçalho WebSphere MQ ou carga útil do aplicativo que inicia o corpo da mensagem JMS se não houver cabeçalho anterior do Websphere MQ .

O fragmento de código a seguir resulta em uma mensagem recebida que é um JMSBytesMessage. Independentemente do conteúdo da mensagem recebida e do campo de formato do MQMD recebido, a mensagem será uma JMSBytesMessage.

```
((MQDestination)destination).setMessageBodyStyle
                            (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

Propriedade de destino WMQ_MESSAGE_BODY
WMQ_MESSAGE_BODY determina se um aplicativo JMS processa o MQRFH2 de uma mensagem WebSphere MQ como parte da carga útil da mensagem (ou seja, como parte do corpo da mensagem JMS).

Tabela 127. Nomes e descrições das propriedades

Propriedade	**Formato curto**	**Descrição**
WMQ_MESSAGE_BODY	MBODY	Se um aplicativo JMS processa o MQRFH2 de uma WebSphere MQ mensagem como parte da carga útil da mensagem (ou seja, como parte do corpo da mensagem JMS).

Tabela 128. Nomes de propriedades, valores e métodos configurados

Propriedade	Valores válidos na ferramenta de administração (padrões em negrito)	Valores válidos em programas	Método configurado
WMQ_MESSAGE_BODY	• **UNSPECIFIED** Ao enviar, as classes do WebSphere MQ para JMS geram ou não e incluem um cabeçalho MQRFH2 , dependendo do valor de WMQ_TARGET_CLIENT. Ao receber, age como valor JMS. • JMS Ao enviar, as classes WebSphere MQ para JMS geram automaticamente um cabeçalho MQRFH2 e o incluem na mensagem WebSphere MQ . Ao receber, as classes do WebSphere MQ para JMS configuram as propriedades de mensagem JMS de acordo com os valores no MQRFH2 (se presente); ele não apresenta o MQRFH2 como parte do corpo da mensagem JMS. • MQ Ao enviar, as classes do WebSphere MQ para JMS não geram um MQRFH2. Ao receber, WebSphere MQ classes para JMS apresenta o MQRFH2 como parte do corpo da mensagem JMS.	• **WMQ_MESSAGE_BODY_UNSPECIFIED** • WMQ_MESSAGE_BODY_JMS • WMQ_MESSAGE_BODY_MQ	setMessageBodyStyle

Mensagens persistentes do JMS

As classes do WebSphere MQ para aplicativos JMS podem usar o atributo da fila **NonPersistentMessageClass** para fornecer melhor desempenho para mensagens persistentes JMS, em detrimento de alguma confiabilidade

Uma fila do WebSphere MQ possui um atributo chamado **NonPersistentMessageClass** O valor desse atributo determina se mensagens não persistentes na fila serão descartadas quando o gerenciador de filas for reiniciado.

É possível configurar o atributo para uma fila local usando o comando DEFINE QLOCAL do WebSphere MQ Script (MQSC) com um dos seguintes parâmetros:

NPMCLASS(NORMAL)
Mensagens não persistentes na fila são descartadas quando o gerenciador de filas for reiniciado. Esse é o valor-padrão.

NPMCLASS(HIGH)
Mensagens não persistentes na fila não são descartadas quando o gerenciador de filas é reiniciado após um encerramento em modo quiesce ou imediato. Mensagens não persistentes podem ser descartadas, no entanto, após um encerramento prioritário ou uma falha.

Este tópico descreve como as classes do WebSphere MQ para aplicativos JMS podem usar esse atributo de fila para fornecer melhor desempenho para mensagens persistentes JMS

A propriedade PERSISTENCE de um objeto Queue ou Topic pode ter o valor HIGH. É possível usar a ferramenta de administração JMS do WebSphere MQ para configurar esse valor ou um aplicativo pode chamar o método Destination.setPersistence() passando o valor WMQConstants.WMQ_PER_NPHIGH como um parâmetro.

Se um aplicativo enviar uma mensagem persistente JMS ou uma mensagem não persistente JMS para um destino em que a propriedade PERSISTENCE tenha o valor HIGH e a fila subjacente do WebSphere MQ estiver configurada como NPMCLASS (HIGH), a mensagem será colocada na fila como uma mensagem não persistente do WebSphere MQ . Se a propriedade PERSISTENCE do destino não tiver o valor HIGH, ou se a fila subjacente estiver configurada como NPMCLASS (NORMAL), uma mensagem persistente JMS será colocada na fila como uma mensagem persistente WebSphere MQ e uma mensagem não persistente JMS será colocada na fila como uma mensagem não persistente WebSphere MQ .

Se uma mensagem persistente JMS for colocada em uma fila como uma mensagem não persistente do WebSphere MQ e você desejar assegurar que a mensagem não seja descartada após um encerramento em modo quiesce ou imediato de um gerenciador de filas, todas as filas por meio das quais a mensagem pode ser roteada deverão ser configuradas para NPMCLASS (HIGH). No domínio de publicar/assinar, essas filas incluem filas de assinantes. Como um auxílio para aplicar essa configuração, as classes WebSphere MQ para JMS emitirá uma exceção InvalidDestinationse um aplicativo tentar criar um consumidor de mensagens para um destino em que a propriedade PERSISTENCE tenha o valor HIGH e a fila subjacente do WebSphere MQ for configurada para NPMCLASS (NORMAL).

Configurar a propriedade PERSISTENCE de um destino para HIGH não afeta como uma mensagem é recebida desse destino. Uma mensagem enviada como uma mensagem persistente JMS é recebida como uma mensagem persistente JMS e uma mensagem enviada como uma mensagem não persistente JMS é recebida como uma mensagem não persistente JMS.

Quando um aplicativo envia a primeira mensagem para um destino em que a propriedade PERSISTENCE possui o valor HIGH ou quando um aplicativo cria o primeiro consumidor de mensagens para um destino em que a propriedade PERSISTENCE possui o valor HIGH, as classes WebSphere MQ para JMS emitirem uma chamada MQINQ para determinar se NPMCLASS (HIGH) está configurada na fila subjacente do WebSphere MQ . O aplicativo deve, portanto, ter a autoridade para consultar na fila. Além disso, WebSphere MQ classes para JMS preserva o resultado da chamada MQINQ até que o destino seja excluído e não emite mais chamadas MQINQ. Portanto, se você alterar a configuração NPMCLASS na fila subjacente enquanto o aplicativo ainda estiver usando o destino, as classes WebSphere MQ para JMS não observarão a nova configuração.

Ao permitir que as mensagens persistentes do JMS sejam colocadas nas filas do WebSphere MQ como WebSphere MQ mensagens não persistentes, você está obtendo desempenho em detrimento de alguma confiabilidade. Se você requerer confiabilidade máxima para mensagens persistentes JMS, não envie as mensagens para um destino em que a propriedade PERSISTENCE tenha o valor HIGH.

A Camada JMS pode usar SYSTEM.JMS.TEMPQ.MODEL, em vez de SYSTEM.DEFAULT.MODEL.QUEUE. SYSTEM.JMS.TEMPQ.MODEL cria filas dinâmicas permanentes que aceitam mensagens persistentes, porque SYSTEM.DEFAULT.MODEL.QUEUE não pode aceitar mensagens persistentes. Se você deseja usar filas provisórias para aceitar mensagens persistentes, deve-se portanto usar SYSTEM.JMS.TEMPQ.MODEL ou mudar a fila modelo para uma fila alternativa de sua escolha.

Usando Secure Sockets Layer (SSL) com classes WebSphere MQ para JMS

As classes WebSphere MQ para aplicativos JMS podem usar criptografia SSL. Para fazer isso, eles requerem um provedor JSSE.

As classes do WebSphere MQ para conexões JMS usando TRANSPORT (CLIENT) suportam criptografia Secure Sockets Layer (SSL). SSL fornece criptografia de comunicação, autenticação e integridade da

mensagem. É geralmente usada para comunicações seguras entre qualquer dois pontos na Internet ou em uma intranet.

WebSphere MQ classes para JMS usa Java Secure Socket Extension (JSSE) para manipular a criptografia SSL e, portanto, requer um provedor JSSE. JVMs JSE v1.4 têm um provedor JSSE integrado. Detalhes sobre como gerenciar e armazenar certificados podem variar de provedor para provedor. Para obter informações sobre isso, consulte a documentação do seu provedor JSSE.

Esta seção supõe que seu provedor JSSE esteja instalado e configurado corretamente e que os certificados adequados tenham sido instalados e disponibilizados para seu provedor JSSE. Agora é possível usar JMSAdmin para configurar várias propriedades administrativas.

Se suas classes do WebSphere MQ para o aplicativo JMS usarem uma tabela de definições de canais do cliente (CCDT) para se conectar a um gerenciador de fila, consulte "Usando uma tabela de definição de canal de cliente com classes IBM WebSphere MQ classes for JMS" na página 927...

Propriedade de objeto SSLCIPHERSUITE
Configure SSLCIPHERSUITE para ativar a criptografia SSL em um objeto ConnectionFactory.

Para ativar a criptografia SSL em um objeto ConnectionFactory, use JMSAdmin para configurar a propriedade SSLCIPHERSUITE para um CipherSuite suportado pelo seu provedor JSSE. Deve corresponder ao CipherSpec configurado no canal de destino. No entanto, CipherSuites são distintos dos CipherSpecs e, portanto, têm nomes diferentes. "SSL CipherSpecs e CipherSuites no JMS" na página 921 contém uma tabela mapeando os CipherSpecs suportados pelo WebSphere MQ para seus CipherSuites equivalentes, conforme conhecido pelo JSSE. Para obter mais informações sobre CipherSpecs e CipherSuites com WebSphere MQ, consulte Segurança.

Por exemplo, para configurar um objeto ConnectionFactory que pode ser usado para criar uma conexão sobre um canal MQI ativado para SSL com um CipherSpec de RC4_MD5_EXPORT, emita o seguinte comando para JMSAdmin:

```
ALTER CF(my.cf) SSLCIPHERSUITE(SSL_RSA_EXPORT_WITH_RC4_40_MD5)
```

Também pode ser configurado a partir de um aplicativo, usando o método setSSLCipherSuite() em um objeto MQConnectionFactory.

Por conveniência, se um CipherSpec for especificado na propriedade SSLCIPHERSUITE, JMSAdmin tentará mapear o CipherSpec para um CipherSuite apropriado e emitirá um aviso. Essa tentativa de mapear não será feita se a propriedade for especificada por um aplicativo.

Como alternativa, use a Tabela de definição de canal de cliente (CCDT). Para obter mais informações, consulte "Usando uma tabela de definição de canal de cliente com classes IBM WebSphere MQ classes for JMS" na página 927.

Propriedade de objeto SSLFIPSREQUIRED
Se você precisar de uma conexão para usar um CipherSuite que seja suportado pelo provedor IBM Java JSSE FIPS (IBMJSSEFIPS), configure a propriedade SSLFIPSREQUIRED do connection factory como YES.

O valor padrão dessa propriedade é NO, o que significa que uma conexão pode usar qualquer CipherSuite suportado pelo WebSphere MQ.

Se um aplicativo usar mais de uma conexão, o valor de SSLFIPSREQUIRED que é usado quando o aplicativo cria a primeira conexão determina o valor que é usado quando o aplicativo cria qualquer conexão subsequente. Isso significa que o valor da propriedade SSLFIPSREQUIRED do connection factory que é usado para criar uma conexão subsequente é ignorado. Deve-se reiniciar o aplicativo se você quiser usar um valor diferente de SSLFIPSREQUIRED.

Um aplicativo pode configurar esta propriedade chamando o método setSSLFipsRequired() de um objeto ConnectionFactory. A propriedade será ignorada se nenhum CipherSuite estiver configurado.

Tarefas relacionadas
Especificando que Apenas CipherSpecs Certificados por FIPS São Usados no Tempo de Execução no Cliente de MQI

Referências relacionadas

Federal Information Processing Standards (FIPS) para UNIX, Linux e Windows

Propriedade de objeto SSLPEERNAME

Use SSLPEERNAME para especificar um padrão de nome distinto para assegurar que seu aplicativo JMS se conecta ao gerenciador de filas correto.

Um aplicativo JMS pode assegurar que ele se conecta ao gerenciador de filas correto especificando um padrão de nome distinto (DN). A conexão será bem-sucedida somente se o gerenciador de filas apresentar um DN que corresponda ao padrão. Para obter mais detalhes sobre o formato desse padrão, consulte os tópicos relacionados.

O DN é configurado usando a propriedade SSLPEERNAME de um objeto ConnectionFactory. Por exemplo, o comando JMSAdmin a seguir configura um objeto ConnectionFactory para esperar que o gerenciador de filas se identifique com um Nome Comum começando com os caracteres `QMGR.` e com pelo menos dois nomes de Unidade Organizacional, o primeiro dos quais deve ser IBM e o segundo WEBSPHERE:

```
ALTER CF(my.cf) SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

A verificação não faz distinção entre maiúsculas e minúsculas e pontos-e-vírgulas podem ser usados no lugar de vírgulas. SSLPEERNAME também pode ser configurado a partir de um aplicativo usando o método setSSLPeerName() em um objeto MQConnectionFactory. Se essa propriedade não estiver configurada, nenhuma verificação será executada no Nome distinto fornecido pelo gerenciador de filas. Essa propriedade será ignorada se nenhum CipherSuite estiver configurado.

Propriedade de objeto SSLCERTSTORES

Utilize SSLCERTSTORES para especificar uma lista de servidores LDAP a serem usados para verificação da lista de revogação de certificado (CRL).

É comum usar a lista de revogação de certificado (CRL) para identificar os certificados que não são mais confiáveis. CRLs geralmente são hospedadas em servidores LDAP. O JMS permite que um servidor LDAP seja especificado para verificação de CRL em Java 2 v1.4 ou posterior. O exemplo de JMSAdmin a seguir direciona JMS para usar uma CRL hospedada em um servidor LDAP denominado crl1.ibm.com:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com)
```

Nota: Para usar um CertStore com sucesso com uma CRL hospedada em um servidor LDAP, certifique-se de que seu Java Software Development Kit (SDK) seja compatível com a CRL... Alguns SDKs requerem que a CRL esteja em conformidade com o RFC 2587, que define um esquema para o LDAP v2. A maioria dos servidores LDAP v3 usa RFC 2256 em vez disso.

Se seu servidor LDAP não estiver em execução na porta padrão 389, será possível especificar a porta anexando dois pontos (:) e o número da porta ao nome do host. Se o certificado apresentado pelo gerenciador de filas estiver presente na CRL hospedada em crl1.ibm.com, a conexão não será concluída. Para evitar um ponto único de falha, o JMS permite que vários servidores LDAP sejam fornecidos, fornecendo uma lista de servidores LDAP delimitados pelo caractere de espaço Aqui está um exemplo:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com ldap://crl2.ibm.com)
```

Quando vários servidores LDAP são especificados, o JMS tenta cada um por vez até localizar um servidor com o qual ele pode verificar com êxito o certificado do gerenciador de filas Cada servidor deve conter informações idênticas.

Uma sequência nesse formato pode ser fornecida por um aplicativo no método MQConnectionFactory.setSSLCertStores(). Como alternativa, o aplicativo pode criar um ou mais objetos java.security.cert.CertStore, coloque-os em um objeto Collection apropriado e forneça esse objeto Collection ao método setSSLCertStores(). Dessa maneira, o aplicativo pode customizar a verificação da CRL. Consulte a documentação de JSSE para obter detalhes sobre como construir e usar objetos CertStore.

O certificado apresentado pelo gerenciador de filas quando uma conexão que está sendo configurada for validada da seguinte forma:

1. O primeiro objeto CertStore no Collection identificado por sslCertStores é usado para identificar um servidor de CRL.
2. Uma tentativa é feita para contatar o servidor de CRL.
3. Se a tentativa for bem-sucedida, uma correspondência do certificado é procurada no servidor.
   a. Se o certificado tiver sido revogado, o processo de procura terminou e a solicitação de conexão falhará com o código de razão MQRC_SSL_CERTIFICATE_REVOKED.
   b. Se o certificado não for localizado, o processo de procura termina e a conexão tem permissão para continuar.
4. Se a tentativa de contatar o servidor não for bem-sucedida, o próximo objeto CertStore será usado para identificar um servidor CRL e o processo é repetido a partir da etapa 2.

   Se esse era o último CertStore no Collection ou se o Collection não contiver nenhum objeto CertStore, o processo de procura falhou e a solicitação de conexão falhará com o código de razão MQRC_SSL_CERT_STORE_ERROR.

O objeto Collection determina a ordem na qual CertStores são usados.

Se seu aplicativo usar setSSLCertStores() para configurar um Collection de objetos CertStore, o MQConnectionFactory não poderá mais ser ligado a namespace JNDI. A tentativa de fazer isso causa uma exceção. Se a propriedade sslCertStores não estiver configurada, nenhuma verificação de revogação será executada no certificado fornecido pelo gerenciador de filas. Essa propriedade será ignorada se nenhum CipherSuite estiver configurado.

Propriedade de objeto SSLRESETCOUNT
Essa propriedade representa o número total de bytes enviados e recebidos por uma conexão antes da chave secreta usada para criptografia ser renegociada.

O número de bytes enviados é o número antes da criptografia e o número de bytes recebidos é o número após a decriptografia. O número de bytes também inclui informações de controle enviadas e recebidas pelas classes WebSphere MQ para JMS.

Por exemplo, para configurar um objeto ConnectionFactory que possa ser usado para criar uma conexão sobre um canal MQI ativado para SSL com uma chave secreta que seja renegociada após a transmissão de 4 MB de dados, emita o comando a seguir para JMSAdmin:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Um aplicativo pode configurar essa propriedade chamando o método setSSLResetCount() de um objeto ConnectionFactory.

Se o valor dessa propriedade for zero, que é o valor padrão, a chave secreta nunca será renegociada. A propriedade será ignorada se nenhum CipherSuite estiver configurado.

Propriedade de objeto SSLSocketFactory
Para customizar outros aspectos da conexão SSL para um aplicativo, crie um SSLSocketFactory e configure JMS para usá-lo.

Você pode desejar customizar outros aspectos da conexão SSL para um aplicativo. Por exemplo, você pode desejar inicializar o hardware criptográfico ou mudar o keystore e o armazenamento confiável em uso. Para fazer isso, o aplicativo deve primeiramente criar um objeto javax.net.ssl.SSLSocketFactory customizado de acordo. Consulte sua documentação do JSSE para obter informações sobre como fazer isso, porque os recursos customizáveis variam de provedor para provedor. Após um objeto SSLSocketFactory adequado ser obtido, use o método MQConnectionFactory.setSSLSocketFactory () para configurar o JMS para usar o objeto SSLSocketFactory customizado.

Se seu aplicativo usar o método setSSLSocketFactory() para configurar um objeto SSLSocketFactory customizado, o objeto MQConnectionFactory não poderá mais ser ligado a um namespace JNDI.

A tentativa de fazer isso causa uma exceção. Se esta propriedade não for configurada, o objeto SSLSocketFactory padrão será usado. Consulte a documentação do JSSE para obter detalhes sobre o comportamento do objeto padrão SSLSocketFactory. Essa propriedade será ignorada se nenhum CipherSuite estiver configurado.

Importante: Não presuma que o uso das propriedades SSL garantirá a segurança quando um objeto ConnectionFactory for recuperado a partir de namespace JNDI que não é seguro em si. Especificamente, a implementação de LDAP padrão do JNDI não é segura. Um invasor pode imitar o servidor LDAP, enganando um aplicativo JMS para se conectar ao servidor errado sem perceber. Com o acordo de segurança adequada no lugar, outras implementações de JNDI (como a implementação fscontext) estão seguras.

Fazendo mudanças no keystore ou no armazenamento confiável de JSSE
Se você fizer mudanças no keystore ou no armazenamento confiável, deverá tomar determinadas ações para que as mudanças sejam captadas.

Se você alterar o conteúdo do keystore ou do armazenamento confiável JSSE ou alterar o local do keystore ou do arquivo de armazenamento confiável, as classes WebSphere MQ para aplicativos JMS que estão em execução no momento não selecionam automaticamente as mudanças. Para que as mudanças entrem em vigor, as seguintes ações devem ser executadas:

- Os aplicativos devem fechar todas as suas conexões e destruir quaisquer conexões não usadas em conjuntos de conexões.
- Se seu provedor JSSE armazenar em cache informações do keystore e do armazenamento confiável, essas informações deverão ser atualizadas.

Após essas ações terem sido executadas, os aplicativos poderão, então, recriar suas conexões.

Dependendo de como você projeta seus aplicativos e sobre a função fornecida pelo seu provedor JSSE, pode ser possível executar estas ações sem parar e reiniciar seus aplicativos. No entanto, parar e reiniciar o aplicativo poderá ser a solução mais simples.

SSL CipherSpecs e CipherSuites no JMS
CipherSpecs suportados pelo WebSphere MQ e seus CipherSuitesequivalentes

Tabela 129 na página 921 lista os CipherSpecs suportados pelo WebSphere MQ e seus CipherSuitesequivalentes.. Se a propriedade ConnectionFactory SSLFIPSREQUIRED for configurada como NO, uma classe WebSphere MQ para aplicativo JMS poderá se conectar a um gerenciador de filas se qualquer CipherSpec suportado for especificado na extremidade do servidor do canal MQI e o CipherSuite equivalente for especificado na extremidade do cliente. Se SSLFIPSREQUIRED for configurado como YES, a combinação de CipherSpec e CipherSuite determinará se o aplicativo pode se conectar ao gerenciador de filas

Na extremidade do servidor de um canal MQI, o nome de um CipherSpec pode ser especificado como o valor do parâmetro SSLCIPH em um comando DEFINE CHANNEL CHLTYPE (SVRCONn). Na extremidade do cliente de um canal MQI, o nome de um CipherSuite pode ser especificado das seguintes maneiras:

- Um aplicativo pode chamar o método setSSLCipherSuite () de um objeto ConnectionFactory .
- Usando a ferramenta de administração JMS do WebSphere MQ , é possível configurar a propriedade SSLCIPHERSUITE de um objeto ConnectionFactory .

Tabela 129. CipherSpecs suportados pelo WebSphere MQ e seus CipherSuites equivalentes

CipherSpec	**Equivalente CipherSuite**	**Conexão possível se o SFIPS [1] estiver configurado como YES?**
NULL_MD5	SSL_RSA_WITH_NULL_MD5	Não

Tabela 129. CipherSpecs suportados pelo WebSphere MQ e seus CipherSuites equivalentes (continuação)		
CipherSpec	**Equivalente CipherSuite**	**Conexão possível se o SFIPS [1] estiver configurado como YES?**
NULL_SHA	SSL_RSA_WITH_NULL_SHA	Não
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5	Não
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5	Não
RC4_SHA_US	SSL_RSA_WITH_RC4_128_SHA	Não
RC2_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	Não
DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA	Não
RC4_56_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_RC4_56_SHA	Não
DES_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA	Não
TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Não
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256	Nenhum [7]
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	Sim [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	Sim [5] [7]
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	Sim [5] [7]
AES_SHA_US[2]		
TLS_RSA_WITH_DES_CBC_SHA[8] [9]	SSL_RSA_WITH_DES_CBC_SHA	Não[3]
TLS_RSA_WITH_3DES_EDE_CBC_SHA[8]	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Sim
FIPS_WITH_DES_CBC_SHA	SSL_RSA_FIPS_WITH_DES_CBC_SHA	Nenhum[4]
FIPS_WITH_3DES_EDE_CBC_SHA	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA	Nenhum[6]

Notes:

1. Ao usar a ferramenta de administração JMS do WebSphere MQ , SFIPS é o nome abreviado da propriedade SSLFIPSREQUIRED ConnectionFactory .
2. Este CipherSpec não tem nenhum CipherSuiteequivalente
3. Esse CipherSpec foi certificado pelo FIPS 140-2 antes de 19th de maio de 2007
4. Esse CipherSpec foi certificado pelo FIPS 140-2 antes de 19th de maio de 2007 O nome FIPS_WITH_DES_CBC_SHA é histórico e reflete o fato de que esse CipherSpec era anteriormente (mas não é mais) compatível com FIPS. Esse CipherSpec foi descontinuado e seu uso não é recomendado.
5. Estas CipherSpecs (TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) não podem ser usadas para proteger uma conexão do WebSphere MQ Explorer para um gerenciador de filas, a menos que os arquivos de políticas irrestritos apropriados sejam aplicados ao JRE usado pelo Explorer

   Consulte Informações de Segurança para obter informações adicionais sobre os arquivos de políticas
6. O nome FIPS_WITH_3DES_EDE_CBC_SHA é histórico e reflete o fato de que esse CipherSpec era anteriormente (mas não é mais) compatível com FIPS.. Esse CipherSpec foi descontinuado e seu uso não é recomendado.

7. Esses CipherSpecs (TLS_RSA_WITH_NULL_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256) requerem IBM JREs 6.0 SR13 FP2 , 7.0 SR4 FP2 ou posterior.
8. Essas CipherSpecs (TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_DES_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA256) podem usar SSLv3 ou TLS. Por padrão, quando FIPS não está ativado, SSLv3 é usado. Para usar TLS, configure a Propriedade do sistema Java **com.ibm.mq.cfg.preferTLS** como `true`
9. Esse CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA foi descontinuado. No entanto, ele ainda pode ser usado para transferir até 32 GB de dados antes de a conexão ser finalizada com erro AMQ9288. Para evitar esse erro, você precisará evitar o uso de DES triplo ou ativar a reconfiguração de chave secreta ao usar esse CipherSpec.

Informações relacionadas

Especificando que Apenas CipherSpecs Certificados por FIPS São Usados no Tempo de Execução no Cliente de MQI

Federal Information Processing Standards (FIPS) para UNIX, Linux e Windows

Gravando saídas do canal em Java para WebSphere MQ classes para JMS

Crie saídas de canal definindo classes Java que implementam interfaces especificadas.

Três interfaces estão definidas no pacote com.ibm.mq.exits:

- WMQSendExit, para uma saída de envio
- WMQReceiveExit, para uma saída de recebimento
- WMQSecurityExit, para uma saída de segurança

O seguinte código de amostra define uma classe que implementa todas as três interfaces:

```
public class MyMQExits implements
WMQSendExit, WMQReceiveExit, WMQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method implements the send exit interface
  public ByteBuffer channelSendExit(
MQCXP channelExitParms,
                                  MQCD channelDefinition,
                                  ByteBuffer agentBuffer)
  {
    // Complete the body of the send exit here
  }
    // This method implements the receive exit interface
  public ByteBuffer channelReceiveExit(
MQCXP channelExitParms,
                                     MQCD channelDefinition,
                                     ByteBuffer agentBuffer)
  {
    // Complete the body of the receive exit here
  }
    // This method implements the security exit interface
  public ByteBuffer channelSecurityExit(
MQCXP channelExitParms,
                                      MQCD channelDefinition,
                                      ByteBuffer agentBuffer)
  {
    // Complete the body of the security exit here
  }
}
```

Cada saída recebe como parâmetros um objeto MQCXP e um objeto MQCD. Estes objetos representam as estruturas MQCXP e MQCD definidas na interface processual.

Quando uma saída de envio é chamada, o parâmetro agentBuffer conterá os dados que estiverem prestes a ser enviados ao gerenciador de filas do servidor. Um parâmetro de comprimento não é necessário porque a expressão agentBuffer.limit() fornece o comprimento dos dados. A saída de envio retorna em

seu valor os dados a serem enviados ao gerenciador de filas do servidor. No entanto, se a saída de envio não for a última saída de envio em uma sequência de saídas de envio, os dados retornados serão passados em vez da próxima saída de envio da sequência. A saída de envio pode retornar uma versão modificada dos dados que ele recebe no parâmetro agentBuffer ou pode retornar dados inalterados. O corpo de saída mais simples possível é portanto:

```
{ return agentBuffer; }
```

Quando uma saída de recebimento é chamada, o parâmetro agentBuffer conterá os dados que foram recebidos do gerenciador de filas do servidor. A saída de recebimento retorna como seu valor os dados a serem transmitidos ao aplicativo pelas classes do WebSphere MQ para JMS. No entanto, se a saída de recebimento não for a última saída de recebimento em uma sequência de saídas de recebimento, os dados retornados serão passados para próxima saída de recebimento na sequência.

Quando uma saída de segurança for chamada, o parâmetro agentBuffer conterá os dados que foram recebidos em um fluxo de segurança na saída de segurança ao final do servidor da conexão. A saída de segurança retorna em seu valor os dados a serem enviados em um fluxo de segurança para a saída de segurança do servidor.

Saídas de canal são chamadas com um buffer que tem uma matriz auxiliar. Para melhor desempenho, a saída deve retornar um buffer com uma matriz auxiliar.

Até 32 de dados do usuário podem ser transmitidos para uma saída de canal quando é chamada. A saída acessa os dados do usuário chamando o método getExitData() do objeto MQCXP. Embora a saída possa mudar os dados do usuário ao chamar o método setExitData(), os dados do usuário são atualizados sempre que a saída é chamada. Quaisquer mudanças feitas nos dados do usuário são, portanto, perdidas. No entanto, a saída pode passar os dados de uma chamada para a próxima usando a área do usuário de saída do objeto MQCXP. A saída acessa a área do usuário de saída por referência, chamando o método getExitUserArea().

Cada classe de saída deve ter um construtor. O construtor pode ser o construtor padrão, conforme mostrado no exemplo anterior ou um construtor com um parâmetro de sequência. O construtor é chamado para criar uma instância da classe de saída para cada saída definida na classe. Portanto, no exemplo anterior, uma instância da classe MyMQExits é criada para a saída de envio, outra instância é criada para a saída de recebimento e uma terceira instância é criada para a saída de segurança. Quando um construtor com um parâmetro de sequência for chamado, o parâmetro conterá os mesmos dados de usuário que são passados para a saída de canal para a qual a instância está sendo criada. Se uma classe de saída tiver um construtor padrão e um único construtor de parâmetro, esse único construtor de parâmetro terá precedência.

Não feche a conexão de dentro de uma saída do canal.

Quando os dados são enviados ao final do servidor de uma conexão, a criptografia SSL é executada *depois* que qualquer saída de canal for chamada. Da mesma forma, quando os dados são recebidos do final do servidor de uma conexão, a decriptografia SSL é executada *antes* de quaisquer saídas de canal serem chamadas.

Nas versões de classes do WebSphere MQ para JMS anteriores à Versão 7.0, as saídas de canal foram implementadas usando as interfaces MQSendExit, MQReceiveExite MQSecurityExit. Ainda é possível usar essas interfaces, mas as novas interfaces são preferenciais para melhor função e desempenho.

Configurando o IBM WebSphere MQ classes for JMS para usar saídas de canal

Um aplicativo IBM WebSphere MQ classes for JMS pode usar a segurança do canal, enviar e receber saídas no canal MQI que é iniciado quando o aplicativo se conecta a um gerenciador de filas. O aplicativo pode usar saídas por escrito em Java, C ou C++. O aplicativo também pode usar uma sequência de envio ou receber saídas que são executadas em sucessão.

As propriedades a seguir são usadas para especificar uma saída de envio ou uma sequência de saídas de envio, usada por uma conexão JMS:

- A propriedade **SENDEXIT** de um objeto MQConnectionFactory.

- A propriedade **sendexit** em uma especificação de ativação usada pelo adaptador de recursos do IBM WebSphere MQ para comunicação de entrada.
- A propriedade **sendexit** em um objeto ConnectionFactory usado pelo adaptador de recursos do IBM WebSphere MQ para comunicação de saída.

O valor da propriedade é uma sequência que consiste em um ou mais itens separados por vírgulas. Cada item identifica uma saída de envio de uma das maneiras a seguir:

- O nome de uma classe que implementa a interface `WMQSendExit` para uma saída de envio escrita em Java.
- Uma sequência no formato *libraryName* (*entryPointName*) para uma saída de envio escrita em C ou C++.

De maneira semelhante, as propriedades a seguir especificam a saída de recebimento, ou sequência de saídas de recebimento, usada por uma conexão:

- A propriedade **RECEXIT** de um objeto MQConnectionFactory.
- A propriedade **receiveexit** em uma especificação de ativação usada pelo adaptador de recursos do IBM WebSphere MQ para comunicação de entrada.
- A propriedade **receiveexit** em um objeto ConnectionFactory usado pelo adaptador de recursos do IBM WebSphere MQ para comunicação de saída.

As propriedades a seguir especificam a saída de segurança usada por uma conexão:

- A propriedade **SECEXIT** de um objeto MQConnectionFactory.
- A propriedade **securityexit** em uma especificação de ativação usada pelo adaptador de recursos do IBM WebSphere MQ para comunicação de entrada.
- A propriedade **securityexit** em um objeto ConnectionFactory usado pelo adaptador de recursos do IBM WebSphere MQ para comunicação de saída.

Para MQConnectionFactories, é possível configurar as propriedades **SENDEXIT**, **RECEXIT** e **SECEXIT** usando a ferramenta de administração IBM WebSphere MQ JMS ou IBM WebSphere MQ Explorer. Como alternativa, um aplicativo pode configurar as propriedades chamando os métodos `setSendExit()`, `setReceiveExit()` e `setSecurityExit()`.

As saídas de canal são carregadas por seu próprio carregador de classe. Para localizar uma saída do canal, o carregador de classes procura os locais a seguir na ordem especificada.

1. O caminho de classe especificado pela propriedade **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** ou pelo atributo **JavaExitsClassPath** na sub-rotina Channels do arquivo de configuração do cliente IBM WebSphere MQ.
2. O caminho de classe especificado pela propriedade de sistema Java **com.ibm.mq.exitClasspath**. Observe que essa propriedade agora está descontinuada.
3. O diretório de saídas do IBM WebSphere MQ, conforme mostrado em Tabela 130 na página 925. O carregador de classes primeiro procura no diretório os arquivos de classe que não são compactados em arquivos Java archive (JAR). Se a saída do canal não for encontrada, o carregador de classes, em seguida, procurará os arquivos JAR no diretório.

Tabela 130. O diretório de saídas IBM WebSphere MQ

Plataforma	Diretório
UNIX and Linux	`/var/mqm/exits` (32-bit saídas de canal) `/var/mqm/exits64` (64-bit saídas de canal)

Tabela 130. O diretório de saídas IBM WebSphere MQ (continuação)	
Plataforma	**Diretório**
Windows	*install_data_dir* \exits em que *install_data_dir* é o diretório que você escolheu para os arquivos de dados do IBM WebSphere MQ durante a instalação. O diretório padrão é `C:\Program Files\IBM\WebSphere MQ`.

Nota: Se uma saída de canal existir em mais de um local, o IBM WebSphere MQ classes for JMS carregará a primeira instância que ele encontrar.

O pai do carregador de classes é o carregador de classes que é usado para carregar o IBM WebSphere MQ classes for JMS. Portanto, é possível que o carregador de classes pai carregue uma saída do canal se ela não puder ser localizada em nenhum dos locais precedentes. No entanto, quando você estiver usando o IBM WebSphere MQ classes for JMS em um ambiente como um servidor de aplicativos JEE, provavelmente não poderá influenciar na escolha do carregador de classes-pai e, portanto, o carregador de classes deverá ser definido configurando a propriedade de sistema Java **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** no servidor de aplicativos.

Se seu aplicativo estiver sendo executado com o Java Security Manager ativado, o arquivo de configuração de política usado pelo ambiente de tempo de execução do Java no qual o aplicativo está em execução deverá ter as permissões para carregar uma classe de saída do canal. Para obter informações sobre como fazer isso, consulte Executando classes do IBM MQ para aplicativos JMS no gerenciador de segurança Java.

As interfaces `MQSendExit`, `MQReceiveExit` e `MQSecurityExit` fornecidas com versões do IBM WebSphere MQ anteriores à Version 7.0 ainda são suportadas. Se você usar saídas de canal que implementem essas interfaces, `com.ibm.mq.jar` deverá estar presente no caminho de classe.

Para obter informações sobre como gravar as saídas de canal em C, consulte “Programas de Saída de Canal para Canais de Mensagens” na página 402. Deve-se armazenar programas de saída de canal gravados em C ou C++ no diretório mostrado em Tabela 130 na página 925.

Se o seu aplicativo usar uma tabela de definição do canal do cliente (CCDT) para conectar a um gerenciador de filas, consulte “Usando uma tabela de definição de canal de cliente com classes IBM WebSphere MQ classes for JMS” na página 927.

Especificando os dados do usuário a serem passados para as saídas do canal ao usar as classes do WebSphere MQ para JMS

Até 32 de dados do usuário podem ser transmitidos para uma saída de canal quando é chamada.

A propriedade SENDEXITINIT de um objeto MQConnectionFactory especifica os dados do usuário que são transmitidos para cada saída de envio quando ele é chamado. O valor da propriedade é uma sequência que consiste em um ou mais itens de dados do usuário separados por vírgulas. A posição de cada item de dados do usuário dentro da sequência determina para qual saída de envio, em uma sequência de saídas de envio, os dados do usuário são transmitidos. Por exemplo, o primeiro item de dados do usuário na sequência é transmitido para a primeira saída de envio em uma sequência de saídas de envio.

É possível configurar a propriedade SENDEXITINIT usando a ferramenta de administração JMS do WebSphere MQ ou WebSphere MQ Explorer. Como alternativa, um aplicativo pode configurar a propriedade chamando o método setSendExitInit().

De forma semelhante, a propriedade RECEXITINIT de um objeto ConnectionFactory especifica os dados do usuário que são transmitidos para cada saída de recebimento e a propriedade SECEXITINIT especifica os dados do usuário transmitidos para uma saída de segurança. É possível configurar essas propriedades usando a ferramenta de administração JMS do WebSphere MQ ou WebSphere MQ Explorer. Como alternativa, um aplicativo pode definir as propriedades chamando os métodos setReceiveExitInit() e setSecurityExitInit().

Observe as regras a seguir ao especificar dados do usuário que são transmitidos para as saídas de canal:

- Se o número de itens de dados do usuário em uma sequência for maior que o número de saídas em uma sequência, os itens em excesso de dados do usuário são ignorados.
- Se o número de itens de dados do usuário em uma sequência for menor que o número de saídas em uma sequência, cada item não especificado de dados do usuário será configurado para uma sequência vazia. Duas vírgulas em sucessão dentro de uma sequência ou uma vírgula no início de uma sequência, também denotam um item não especificado de dados do usuário.

Se um aplicativo usar uma tabela de definição de canal do cliente (CCDT) para se conectar a um gerenciador de filas, quaisquer dados do usuário especificados em uma definição de canal de conexão do cliente serão transmitidos para as saídas de canal quando forem chamados. Para obter mais informações sobre como usar uma tabela de definição de canal do cliente, consulte "Usando uma tabela de definição de canal de cliente com classes IBM WebSphere MQ classes for JMS" na página 927.

Usando uma tabela de definição de canal de cliente com classes IBM WebSphere MQ classes for JMS

Uma classe IBM WebSphere MQ para o aplicativo JMS pode usar definições de canal de conexão do cliente que são armazenadas em uma tabela de definição de canal do cliente (CCDT)... Configure um objeto ConnectionFactory para usar a CCDT. Existem algumas restrições sobre seu uso.

Como uma alternativa para criar uma definição de canal de conexão do cliente configurando determinadas propriedades de um objeto ConnectionFactory , um aplicativo IBM WebSphere MQ classes for JMS pode usar definições de canal de conexão do cliente armazenadas em uma tabela de definição de canal do cliente. Essas definições são criadas por comandos IBM WebSphere MQ Script (MQSC) ou comandos IBM WebSphere MQ Programmable Command Format (PCF). Quando o aplicativo cria um objeto Connection, as classes IBM WebSphere MQ classes for JMS procura na tabela de definição de canal do cliente uma definição de canal de conexão do cliente adequada e usa a definição de canal para iniciar um canal MQI. Para obter mais informações sobre tabelas de definição de canal do cliente e como construir um, consulte Client Channel Definition Table.

Para usar uma tabela de definição de canal do cliente, a propriedade CCDTURL de um objeto ConnectionFactory deve ser configurada para um objeto de URL. O objeto de URL contém um localizador uniforme de recursos (URL) que identifica o nome e o local do arquivo contendo a tabela de definição de canal do cliente e especifica como o arquivo pode ser acessado. É possível configurar a propriedade CCDTURL usando a ferramenta de administração JMS do IBM WebSphere MQ ou um aplicativo pode configurar a propriedade criando um objeto URL e chamando o método setCCDTURL() do objeto ConnectionFactory .

Por exemplo, se o ccdt1.tab do arquivo contiver uma tabela de definição de canal do cliente e for armazenado no mesmo sistema no qual o aplicativo está em execução, o aplicativo pode configurar a propriedade CCDTURL da seguinte maneira:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
factory.setCCDTURL(chanTab1);
```

Como outro exemplo, suponha que o arquivo ccdt2.tab contenha uma tabela de definição de canal do cliente e esteja armazenada em um sistema diferente daquele no qual o aplicativo está em execução. Se o arquivo puder ser acessado usando o protocolo FTP, o aplicativo poderá configurar a propriedade CCDTURL da seguinte maneira:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
factory.setCCDTURL(chanTab2);
```

Além da configuração da propriedade CCDTURL do objeto ConnectionFactory, a propriedade QMANAGER do mesmo objeto deve ser configurada como um dos seguintes valores:

- O nome de um gerenciador de filas
- Um asterisco (*) seguido pelo nome de um grupo de gerenciadores de filas
- Um asterisco (*)

- Uma sequência vazia ou uma sequência contendo todos os caracteres em branco

Estes são os mesmos valores que podem ser usados para o parâmetro *QMgrName* em uma chamada MQCONN emitida por um aplicativo cliente que está usando o Message Queue Interface (MQI). Para obter mais informações sobre o significado desses valores, portanto, consulte MQCONN. É possível configurar a propriedade QMANAGER usando a ferramenta de administração JMS do WebSphere MQ ou IBM WebSphere MQ Explorer. Como alternativa, um aplicativo pode configurar a propriedade chamando o método setQueueManager() do objeto ConnectionFactory.

Se um aplicativo então cria um objeto Connection a partir do objeto ConnectionFactory , IBM WebSphere MQ classes for JMS acessa a tabela de definição de canal do cliente identificada pela propriedade CCDTURL, usa a propriedade QMANAGER para procurar a tabela para uma definição de canal de conexão do cliente adequada e, em seguida, usa a definição de canal para iniciar um canal MQI para um gerenciador de filas.

Observe que as propriedades CCDTURL e CHANNEL de um objeto ConnectionFactory não poderão ser ambas configuradas quando o aplicativo chamar o método createConnection(). Se ambas propriedades forem configuradas, o método lançará uma exceção. A propriedade CCDTURL ou CHANNEL será considerada para ser configurada ou se seu valor for algo diferente de nulo, uma sequência vazia ou uma sequência que contém todos os caracteres em branco.

Quando IBM WebSphere MQ classes for JMS localiza uma definição de canal de conexão do cliente adequada na tabela de definições de canal do cliente, ele usa apenas as informações extraídas da tabela para iniciar um canal MQI. Quaisquer propriedades relacionadas ao canal do objeto ConnectionFactory são ignoradas.

Em particular, observe os seguintes pontos se você estiver usando Secure Sockets Layer (SSL):

- Um canal MQI usará SSL somente se a definição de canal extraída da tabela de definições de canal do cliente especificar o nome de um CipherSpec suportado pelo IBM WebSphere MQ classes for JMS.
- Uma tabela de definição de canal do cliente também contém informações sobre o local dos servidores Lightweight Directory Access Protocol (LDAP) que mantém as listas de revogação de certificado (CRLs). IBM WebSphere MQ classes for JMS usa apenas essas informações para acessar servidores LDAP que contêm CRLs.
- Uma tabela de definição de canal de cliente também pode conter o local de um respondente do Online Certificate Status Protocol (OCSP). As classes IBM WebSphere MQ classes for JMS não podem usar as informações de OCSP em um arquivo de tabela de definições de canal do cliente. No entanto, você pode configurar o OCSP conforme descrito na seção Usando Online Certificate Protocol.

Para obter mais informações sobre como usar SSL com uma tabela de definição de canal do cliente, consulte Usando o cliente transitório estendido com canais SSL.

Observe também os pontos a seguir se estiver usando saídas de canal:

- Um canal MQI usa somente as saídas de canal e dados do usuário associados especificados pela definição de canal extraída da tabela de definição de canal do cliente.
- Uma definição de canal extraída de uma tabela de definição de canal do cliente pode especificar saídas de canal que são gravadas em Java.. Isso significa, por exemplo, que o parâmetro SCYEXIT no comando DEFINE CHANNEL para criar uma nova definição de canal de conexão do cliente pode especificar o nome de uma classe que implementa a interface WMQSecurityExit. De forma semelhante, o parâmetro SENDEXIT pode especificar o nome de uma classe que implementa a interface WMQSendExit e o parâmetro RCVEXIT pode especificar o nome de uma classe que implementa a interface WMQReceiveExit. Para obter mais informações sobre como gravar uma saída de canal em Java, consulte “Gravando saídas do canal em Java para WebSphere MQ classes para JMS” na página 923..

  O uso de saídas do canal gravadas em uma linguagem diferente de Java também é suportado. Para obter informações sobre como especificar os parâmetros SCYEXIT, SENDEXIT e RCVEXIT no comando DEFINE CHANNEL para saídas do canais gravadas em outra linguagem, consulte DEFINE CHANNEL.

Reconexão automática do cliente JMS

Configure seu cliente JMS para se reconectar automaticamente após uma falha de rede, gerenciador de fila ou servidor.

Use as propriedades `CONNECTIONNAMELIST` e `CLIENTRECONNECTOPTIONS` da classe `MQConnectionFactory` para configurar uma conexão do cliente para se reconectar automaticamente após uma falha de conexão ou uma solicitação administrativa para reconectar aplicativos clientes depois de parar um gerenciador de filas

A lista completa de nomes de conexão em uma Lista connectionNameé acessível apenas aos métodos set /getconnectionNameList que podem manipular uma lista de nomes de conexões. Métodos como get / setHostname que não manipulam listas de nomes, acessam o nome na lista.

As conexões do cliente reconectáveis automaticamente somente se tornam reconectáveis quando a conexão for estabelecida.

Se um aplicativo continua funcionando corretamente após ser reconectado automaticamente depende de seu design. Leia os tópicos relacionados para entender como projetar clientes reconectáveis Alguns clientes existentes podem funcionar corretamente sem modificação após reconexão automática.

A reconexão do cliente automática não é suportada pelas classes do WebSphere MQ para Java.

Para evitar que todos os clientes conectados a um gerenciador de filas com falha se reconectem simultaneamente, as tentativas de reconexão são atrasadas por intervalos que são parcialmente fixos e parcialmente aleatórios.

Por padrão, as tentativas de reconexão ocorrem nos intervalos a seguir:

1. A primeira tentativa é feita após um atraso inicial de um segundo, mais um elemento aleatório até 250 milissegundos.
2. A segunda tentativa é feita dois segundos, mais um intervalo aleatório de até 500 milissegundos, após a primeira tentativa falhar.
3. A terceira tentativa é feita quatro segundos, mais um intervalo aleatório de até um segundo, após a segunda tentativa falhar.
4. A quarta tentativa é feita oito segundos, mais um intervalo aleatório de até dois segundos, após a terceira tentativa falhar.
5. A quinta tentativa é feita 16 segundos, mais um intervalo aleatório de até quatro segundos, após a quarta tentativa falhar.
6. A sexta tentativa, e todas as tentativas subsequentes são feitas 25 segundos, mais um intervalo aleatório de até seis segundos e 250 milissegundos, após a tentativa anterior falhar.

Esse processo de reconexão continua até que o cliente seja reconectado com êxito ao gerenciador de filas ou até que o intervalo máximo de reconexão ocorra.

Se você precisar aumentar os valores padrão, para refletir mais precisamente a quantidade de tempo necessária para o gerenciador de filas se recuperar ou o gerenciador de filas em espera para ativar, altere os valores de atraso no MQCLIENT.INI usando o atributo **ReconDelay** .

Conceitos relacionados

reconexão automatizada do cliente

Tarefas relacionadas

Configurando um Cliente Usando um Arquivo de Configuração

Compartilhando uma conexão TCP/IP nas classes IBM WebSphere MQ classes for JMS

Várias instâncias de um canal MQI podem ser feitas para compartilhar uma única conexão TCP/IP.

Os aplicativos que estão em execução dentro do mesmo Java Runtime Environment e que usam as classes IBM WebSphere MQ classes for JMS ou as classes IBM WebSphere MQ se conecte a um gerenciador de filas usando o transporte CLIENT, podem ser feitos para compartilhar a mesma instância do canal.

Há um relacionamento um-a-um entre as instâncias do canal e conexões TCP/IP. Uma conexão TCP/IP é criada para cada instância do canal.

Se um canal for definido com o parâmetro **SHARECNV** configurado para um valor maior que 1, esse número de conversações pode compartilhar uma instância do canal. Para ativar um connection factory ou uma especificação de ativação para usar esta função, configure a propriedade **SHARECONVALLOWED** como YES.

Cada conexão JMS e sessão JMS criada por um aplicativo JMS cria sua própria conversa com o gerenciador de filas.

Quando uma especificação de ativação é inicializada, as classes IBM WebSphere MQ para o adaptador de recursos JMS iniciam uma conversa com o gerenciador de filas para a especificação de ativação usar. Cada sessão do servidor no conjunto de sessões do servidor que está associada com a especificação de ativação também inicia uma conversação com o gerenciador de filas.

O atributo SHARECNV é uma abordagem de melhor esforço para o compartilhamento de conexão. Portanto, quando um valor SHARECNV maior que 0 é usado com as classes IBM WebSphere MQ classes for JMS, não é garantido que um novo pedido de conexão sempre compartilhará uma conexão já estabelecida.

Calculando o número de instâncias do canal

Use as fórmulas a seguir para determinar o número máximo de instâncias do canal que são criados por um aplicativo:

Especificações de ativação

Número de instâncias de canal = (<**maxPoolDepth**> + 1) / <**SHARECNV**>

Em que <**maxPoolDepth**> é o valor da propriedade **maxPoolDepth** e <**SHARECNV**> é o valor da propriedade **SHARECNV** no canal usado pela especificação de ativação.

Outros aplicativos JMS

Número de instâncias do canal = (< conexões JMS> + < sessões JMS>) / <**SHARECNV**>

Em que < JMS connections> é o número de conexões que são criadas pelo aplicativo, em que < JMS sessions> é o número de sessões JMS criadas pelo aplicativo e <**SHARECNV**> é o valor da propriedade **SHARECNV** no canal que é usado pela especificação de ativação.

Examples

Os exemplos a seguir mostram como usar as fórmulas para calcular o número de instâncias do canal que são criadas em um gerenciador de filas por aplicativos usando o adaptador de recursos IBM WebSphere MQ classes for JMS ou IBM WebSphere MQ classes for JMS.

exemplo de aplicativo JMS

Uma conexão de aplicativo JMS se conecta a um gerenciador de filas usando o transporte CLIENT e cria uma conexão JMS e três sessões JMS. O canal que o aplicativo está usando para se conectar ao gerenciador de filas tem a propriedade **SHARECNV** configurada para o valor de 10. Quando o aplicativo está em execução, existem quatro conversas entre o aplicativo e o gerenciador de filas e uma instância do canal. As quatro conversações compartilham entre si a instância do canal.

Exemplo de especificação de ativação

Uma especificação de ativação se conecta a um gerenciador de filas usando o transporte CLIENT. A especificação de ativação é configurada com a propriedade **maxPoolDepth** configurada para 10. O canal que a especificação de ativação está configurada para usar tem a propriedade **SHARECNV** configurada como 10. Quando a especificação de ativação está em execução e o processando 10 mensagens simultaneamente, o número de conversações entre a especificação de ativação e o gerenciador de filas é 11 (10 conversações para as sessões do servidor e um para a especificação de ativação). O número de instâncias do canal que são usadas pela especificação de ativação é 2.

Exemplo de especificação de ativação

Uma especificação de ativação se conecta a um gerenciador de filas usando o transporte CLIENT. A especificação de ativação é configurada com o conjunto de propriedades **maxPoolDepth** para 5. O canal que a especificação de ativação está configurado para usar tem o conjunto de propriedades **SHARECNV** configurado para 0. Quando a especificação de ativação está em execução, e processando 5 mensagens simultaneamente, o número de conversas entre a especificação de ativação e o gerenciador de filas é de 6 (cinco conversas para as sessões do servidor, e uma para a especificação de ativação). O número de instâncias do canal que são usadas pela especificação de ativação é 6, porque a propriedade **SHARECNV** no canal está configurada como 0, cada conversa usa sua própria instância do canal.

Especificando um intervalo de portas para conexões do cliente nas classes do WebSphere MQ para JMS

Use a propriedade LOCALADDRESS para especificar um intervalo de portas ao qual seu aplicativo pode ser ligado.

Quando um WebSphere MQ classes para o aplicativo JMS tenta se conectar a um gerenciador de filas do WebSphere MQ no modo cliente, um firewall pode permitir apenas as conexões que se originam de portas especificadas ou um intervalo de portas. Nesta situação, é possível usar a propriedade LOCALADDRESS de um objeto ConnectionFactory, QueueConnectionFactory ou TopicConnectionFactory para especificar uma porta ou um intervalo de portas, que o aplicativo pode ser ligado.

É possível configurar a propriedade LOCALADDRESS usando a ferramenta de administração JMS do WebSphere MQ ou chamando o método setLocalAddress () em um aplicativo JMS. A seguir há um exemplo da configuração da propriedade de dentro de um aplicativo:

```
mqConnectionFactory.setLocalAddress("192.0.2.0(2000,3000)");
```

Quando o aplicativo se conectar a um gerenciador de filas subsequentemente, o aplicativo será ligado a um endereço IP local e o número da porta no intervalo de 192.0.2.0(2000) para 192.0.2.0(3000).

Em um sistema com mais de uma interface de rede, também é possível usar a propriedade LOCALADDRESS para especificar qual interface de rede deve ser usada para uma conexão.

Para uma conexão em tempo real a um broker, a propriedade LOCALADDRESS será relevante apenas quando multicast for usado. Neste caso, é possível usar a propriedade para especificar qual interface de rede local deve ser usada para uma conexão, mas o valor da propriedade não deve conter um número de porta ou um intervalo de números de porta.

Os erros de conexão poderão ocorrer se você restringir o intervalo de portas. Se ocorrer um erro, uma JMSException será lançada com uma MQException integrada que contém o WebSphere MQ código de razão MQRC_Q_MGR_NOT_AVAILABLE e a mensagem a seguir:

```
A tentativa de conexão do soquete foi recusada devido às restrições de LOCAL_ADDRESS_PROPERTY
```

Um erro pode ocorrer se todas as portas no intervalo especificado estiverem em uso ou se o endereço IP especificado, um nome do host ou número de porta não for válido (um número de porta negativo, por exemplo).

Como as classes do WebSphere MQ para JMS podem criar conexões diferentes daquelas requeridas por um aplicativo, sempre considere especificar um intervalo de porta. Em geral, cada sessão criada por um aplicativo requer uma porta e as classes WebSphere MQ para JMS podem requerer três ou quatro portas adicionais. Se um erro de conexão ocorrer, aumente o intervalo de portas.

O conjunto de conexões, que é usado por padrão nas classes WebSphere MQ para JMS, pode ter um efeito sobre a velocidade na qual as portas podem ser reutilizadas Como resultado, um erro de conexão poderá ocorrer enquanto as portas estiverem sendo liberadas.

Compactação de canal em classes WebSphere MQ para JMS

Um WebSphere MQ classes para aplicativo JMS pode usar WebSphere MQ recursos para compactar um cabeçalho da mensagem ou dados.

Compactar os dados que fluem em um canal do WebSphere MQ pode melhorar o desempenho do canal e reduzir o tráfego de rede. Usando a função fornecida com o WebSphere MQ, é possível compactar os dados que fluem nos canais de mensagem e nos canais MQI Em qualquer tipo de canal, é possível compactar dados de cabeçalho e dados de mensagem de forma independente uns dos outros. Por padrão, nenhum dados é compactado em um canal.

Uma classe WebSphere MQ para o aplicativo JMS especifica as técnicas que podem ser usadas para compactar dados de cabeçalho ou de mensagens em uma conexão criando um objeto java.util.Collection . Cada técnica de compactação é um objeto Integer na coleta e a ordem na qual o aplicativo inclui as técnicas de compactação na coleta é a ordem na qual as técnicas de compactação são negociadas com o gerenciador de filas quando o aplicativo cria a conexão. O aplicativo pode então transmitir a coleta para um objeto ConnectionFactory chamando o método setHdrCompList(), para os dados do cabeçalho ou o método setMsgCompList() para dados da mensagem. Quando o aplicativo estiver pronto, ele poderá criar a conexão.

Os seguintes fragmentos de código ilustram a abordagem descrita. O primeiro fragmento de código mostra como implementar a compactação de dados de cabeçalho:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(WMQConstants.WMQ_COMPHDR_SYSTEM));
.
.
.
((MQConnectionFactory) cf).setHdrCompList(headerComp);
.
.
.
connection = cf.createConnection();
```

O segundo fragmento de código mostra como implementar a compactação dos dados da mensagem:

```
Collection msgComp = new Vector();
msgComp.add(new Integer(WMQConstants.WMQ_COMPMSG_RLE));
msgComp.add(new Integer(WMQConstants.WMQ_COMPMSG_ZLIBHIGH));
.
.
.
((MQConnectionFactory) cf).setMsgCompList(msgComp);
.
.
.
connection = cf.createConnection();
```

No segundo exemplo, as técnicas de compactação são negociadas na ordem RLE, em seguida ZLIBHIGH, quando a conexão é criada. A técnica de compactação selecionada não pode ser mudada durante a existência do objeto Connection. Para usar a compactação em uma conexão, os métodos setHdrCompList() e o setMsgCompList() deverão ser chamados antes de criar o objeto Connection.

Colocando mensagens assincronamente em classes IBM WebSphere MQ para JMS

Normalmente, quando um aplicativo enviar mensagens para um destino, o aplicativo precisará aguardar para o gerenciador de filas confirmar se ele processou a solicitação. É possível melhorar o desempenho do sistema de mensagens em algumas circunstâncias, escolhendo em vez de colocar mensagens assincronamente. Quando um aplicativo efetuar put de uma mensagem de forma assíncrona, o gerenciador de filas não retornará o sucesso ou falha de cada chamada, mas será possível verificar erros periodicamente.

Se um destino retornar o controle ao aplicativo, sem determinar se o gerenciador de filas recebeu a mensagem com segurança, dependerá das seguintes propriedades:

- A Propriedade de Destino JMS PUTSYNCALLOWED (nome abreviado-PAALD).

  PUTASYNCALLOWED controla se os aplicativos JMS podem colocar mensagens de forma assíncrona, se a fila ou o tópico subjacente que o Destino JMS representa, permite essa opção
- A propriedade de fila ou de tópico IBM WebSphere MQ DEFPRESP (tipo de resposta de put padrão).

DEFPRESP especifica se os aplicativos que colocam mensagens na fila ou publicam mensagens para o tópico, podem usar a funcionalidade de colocação assíncrona.

A tabela a seguir mostra os valores possíveis das propriedades PUTASYNCALLOWED e DEFPRESP e quais valores são necessários para que a funcionalidade de colocação assíncrona seja ativada:

Tabela 131. As propriedades PUTASYNCALLOWED e DEFPRESP determinam se as mensagens são colocadas de forma assíncrona.

propriedade da fila do WebSphere MQ	**PUTASYNCALLOWED = NO**	**PUTASYNCALLOWED = YES**	**PUTASYNCALLOWED = AS_DEST or AS_Q_DEF or AS_T_DEF**
DEFPRESP=SYNC	Funcionalidade de colocação assíncrona não ativada	Funcionalidade de colocação assíncrona ativada	Funcionalidade de colocação assíncrona não ativada
DEFPRESP=ASYNC	Funcionalidade de colocação assíncrona não ativada	Funcionalidade de colocação assíncrona ativada	Funcionalidade de colocação assíncrona ativada

Para as mensagens enviadas em uma sessão transacionada, o aplicativo determinará definitivamente se o gerenciador de filas recebeu as mensagens com segurança quando chama `commit()`.

Se um aplicativo enviar mensagens persistentes em uma sessão transacionada e uma ou mais das mensagens não forem recebidas com segurança, a transação falhará ao confirmar e produz uma exceção. No entanto, se um aplicativo enviar mensagens não persistentes em uma sessão transacionada e uma ou mais das mensagens não forem recebidas com segurança, a transação será confirmada com sucesso. O aplicativo não recebe nenhum feedback que as mensagens não persistentes não chegou com segurança.

Para mensagens não persistentes enviadas em uma sessão que não é transacionada, a propriedade SENDCHECKCOUNT do objeto *ConnectionFactory* especifica quantas mensagens devem ser enviadas, antes das classes IBM WebSphere MQ para JMS verificar se o gerenciador de filas recebeu as mensagens com segurança.

Se uma verificação descobrir que uma ou mais mensagens não foram recebidas com segurança e o aplicativo tiver registrado um listener de exceção com a conexão, as classes IBM WebSphere MQ para JMS chamarão o método `onException()` do listener de exceção para transmitir uma exceção JMS para o aplicativo.

A exceção JMS possui um código de erro de JMSWMQ0028 e este código exibe a seguinte mensagem:

```
At least one asynchronous put message failed or gave a warning.
```

A exceção JMS também possui uma exceção vinculada que fornece mais detalhes.. O valor padrão da propriedade SENDCHECKCOUNT é zero, o que significa que nenhum dessas verificações é feita.

Esta otimização é mais benéfica para um aplicativo que se conecta a um gerenciador de filas no modo cliente e precisa enviar uma sequência de mensagens em sucessão rápida, mas não requer feedback imediato do gerenciador de filas para cada mensagem enviada. No entanto, um aplicativo pode ainda usar essa otimização mesmo se ele se conectar a um gerenciador de filas no modo de ligações, mas o benefício de desempenho esperado não será tão grande.

Usando a leitura antecipada com classes do WebSphere MQ para JMS

A funcionalidade de leitura antecipada que é fornecida pelo WebSphere MQ permite que mensagens não persistentes recebidas fora de uma transação sejam enviadas para o IBM WebSphere MQ classes for JMS antes que um aplicativo as solicite. O IBM WebSphere MQ classes for JMS armazena as mensagens em um buffer interno e as transmite ao aplicativo quando o aplicativo solicitá-las.

Os aplicativos IBM WebSphere MQ classes for JMS que usam `MessageConsumers` ou `MessageListeners` para receber mensagens de um destino fora de uma transação podem usar a funcionalidade de leitura antecipada. O uso da leitura antecipada permite que os aplicativos que usam esses objetos se beneficiem do desempenho aprimorado quando receberem mensagens.

Se um aplicativo que usa `MessageConsumers` ou `MessageListeners` pode usar a leitura antecipada depende das seguintes propriedades:

- A Propriedade de Destino JMS READAHEADALLOWED (nome abreviado-RAALD). READAHEADALLOWED controla se os aplicativos JMS podem usar a leitura antecipada ao obter ou procurar mensagens não persistentes fora de uma transação, se a fila ou o tópico subjacente que o Destino JMS representa, permitir essa opção
- O DEFREADA da propriedade de fila ou tópico do IBM WebSphere MQ (leitura antecipada padrão). DEFREADA especifica se os aplicativos que estão recebendo ou procurando mensagens não persistentes fora de uma transação podem usar a leitura antecipada.

A tabela a seguir mostra os valores possíveis para as propriedades READAHEADALLOWED e DEFREADA e quais valores são necessários para a funcionalidade de leitura antecipada a ser ativada:

Tabela 132. As propriedades READAHEADALLOWED e DEFREADA que determinam se a leitura antecipada será usada ao receber ou procurar mensagens não persistentes fora de uma transação.

Propriedade de destino do WebSphere MQ	**READAHEADALLOWED = YES**	**READAHEADALLOWED = NO**	**AS_DEST or AS_Q_DEF or AS_T_DEF**
propriedade da fila do WebSphere MQ			
DEFREADA = NO	Funcionalidade de leitura antecipada ativada	Funcionalidade de leitura antecipada não ativada	Funcionalidade de leitura antecipada não ativada
DEFREADA = YES	Funcionalidade de leitura antecipada ativada	Funcionalidade de leitura antecipada não ativada	Funcionalidade de leitura antecipada ativada
DEFREADA = DISABLED	Funcionalidade de leitura antecipada não ativada	Funcionalidade de leitura antecipada não ativada	Funcionalidade de leitura antecipada não ativada

Se a funcionalidade de leitura antecipada estiver ativada, quando um `MessageConsumer` ou `MessageListener` for criado por um aplicativo, o IBM WebSphere MQ classes for JMS criará um buffer interno para o destino que o `MessageConsumer` ou `MessageListener` estará monitorando. Há um buffer interno de cada `MessageConsumer` ou `MessageListener`. O gerenciador de filas iniciará o envio de mensagens não persistentes para o IBM WebSphere MQ classes for JMS quando o aplicativo chamar um dos seguintes métodos:

- `MessageConsumer.receive()`
- `MessageConsumer.receive(long timeout)`
- `MessageConsumer.receiveNoWait()`
- `Session.setMessageListener(MessageListener listener)`

O IBM WebSphere MQ classes for JMS retorna automaticamente a primeira mensagem de volta ao aplicativo através da chamada de método que o aplicativo fez. As outras mensagens não persistentes são armazenadas pelo IBM WebSphere MQ classes for JMS no buffer interno criado para o destino. Quando o aplicativo solicitar a próxima mensagem para processar, o IBM WebSphere MQ classes for JMS retornará a próxima mensagem no buffer interno.

O IBM WebSphere MQ classes for JMS solicitará mais mensagens não persistentes a partir do gerenciador de filas quando o buffer interno estiver vazio.

O buffer interno usado pelo IBM WebSphere MQ classes for JMS é excluído quando um aplicativo fecha um `MessageConsumer`ou a Sessão JMS à qual um `MessageListener` está associado.

Para `MessageConsumers`, quaisquer mensagens não processadas no buffer interno serão perdidas.

Ao usar `MessageListeners`, o que acontece com as mensagens no buffer interno depende da propriedade de destino JMS READAHEADCLOSEPOLICY (nome abreviado-RACP). O valor padrão da propriedade é DELIVER_ALL, o que significa que a sessão JMS que foi usada para criar o `MessageListener` não será fechada até que todas as mensagens no buffer interno sejam entregues

ao aplicativo. Se a propriedade for configurada como DELIVER_CURRENT, a sessão JMS será fechada depois que a mensagem atual tiver sido processada pelo aplicativo e todas as mensagens restantes no buffer interno forem descartados

Publicações retidas em classes WebSphere MQ para JMS

Um WebSphere MQ classes para o cliente JMS pode ser configurado para usar publicações retidas

Um publicador pode especificar que uma cópia de uma publicação deve ser retida para que ela possa ser enviada aos assinantes futuros que registrarem um interesse no tópico. Isso é feito nas classes do WebSphere MQ para JMS configurando a propriedade de número inteiro JMS_IBM_RETAIN para o valor 1. Constantes foram definidas para esses valores na interface do tipo decom.ibm.msg.client.jms.JmsConstants. Por exemplo, se você tiver criado uma mensagem *msg*, para configurar como uma publicação retida, use o código a seguir:

```
// set as a retained publication
msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION);
```

Agora é possível enviar a mensagem como normal. JMS_IBM_RETAIN também pode ser consultada em uma mensagem recebida. Portanto, é possível consultar se uma mensagem recebida é uma publicação retida.

Suporte XA em classes WebSphere MQ para JMS

O JMS suporta transações compatíveis com XA em ligações e modos do cliente com um gerenciador de transações suportado

Se você precisar da funcionalidade XA em um ambiente de servidor de aplicativos, deverá configurar seu aplicativo de forma apropriada. Consulte a documentação própria de seu servidor de aplicativos para obter informações sobre como configurar os aplicativos para usar as transações distribuídas.

Usando uma conexão em tempo real com um broker do WebSphere Event Broker ou do WebSphere Message Broker

Uma classe do WebSphere MQ para o aplicativo JMS pode usar uma conexão em tempo real com um broker do WebSphere Event Broker ou WebSphere Message Broker para sistema de mensagens de publicação / assinatura. O broker e as classes WebSphere MQ para JMS devem ser configurados para ativar uma conexão em tempo real.

Quando um aplicativo usa uma conexão em tempo real com um broker do WebSphere Event Broker ou do WebSphere Message Broker, o aplicativo e o broker trocam mensagens usando o Transporte em tempo real do WebSphere MQ Dependendo da configuração, as mensagens também podem ser entregues ao aplicativo usando o WebSphere MQ Multicast Transport.

Para obter informações sobre como um aplicativo pode se conectar a um gerenciador de fila do WebSphere MQ e usar o WebSphere MQ Enterprise Transport para trocar mensagens com um broker do WebSphere Event Broker ou do WebSphere Message Broker, consulte a documentação para liberações anteriores de classes WebSphere MQ para JMS. Observe que, para usar o WebSphere MQ Enterprise Transport, um aplicativo deve se conectar a um gerenciador de filas usando um connection factory em execução no modo de migração do provedor de sistemas de mensagens WebSphere MQ .

Configurando um broker do WebSphere Event Broker ou do WebSphere Message Broker para uma conexão em tempo real

Para que as classes do WebSphere MQ para o aplicativo JMS usem uma conexão em tempo real com um broker do WebSphere Event Broker ou do WebSphere Message Broker, deve-se configurar o broker criando e implementando um fluxo de mensagens para ler mensagens da porta TCP/IP na qual o broker está atendendo e publicando as mensagens. Dependendo de seus requisitos, pode ser necessário configurar o broker de maneiras adicionais

Para configurar o broker, você deve criar e implementar um dos fluxos de mensagens a seguir:

- Um fluxo de mensagens que contém um nó de processamento de mensagens Real-timeOptimizedFlow

- Um fluxo de mensagens que contém um nó de processamento de mensagens Real-timeInput e um nó de processamento de mensagem de Publicação..

Deve-se configurar o nó Real-timeOptimizedFlow ou Real-timeInput para atender na porta TCP/IP usada para conexões em tempo real Por padrão, o número da porta para conexões em tempo real é 1506..

Você também deve configurar o broker se tiver qualquer um dos seguintes requisitos:

- Se desejar que o aplicativo se conecte ao broker usando a autenticação Secure Sockets Layer (SSL)
- Se desejar que o aplicativo se conecte ao broker usando o tunelamento HTTP
- Se desejar que as mensagens sejam entregues a um consumidor de mensagens usando multicast

Para obter informações sobre como configurar um broker, consulte a documentação do produto *WebSphere Event Broker* ou *WebSphere Documentação do produto Message Broker*

Configurando as classes do WebSphere MQ para JMS para uma conexão em tempo real com um broker do WebSphere Event Broker ou WebSphere Message Broker

Para que as classes do WebSphere MQ para o aplicativo JMS usem uma conexão em tempo real com um broker de WebSphere Event Broker ou WebSphere Message Broker, WebSphere MQ classes para JMS devem ser configuradas configurando determinadas propriedades do connection factory. Dependendo de seus requisitos, as classes WebSphere MQ para JMS podem precisar ser configuradas de maneiras adicionais.

Para configurar as classes do WebSphere MQ para JMS, as seguintes propriedades do connection factory devem ser configuradas:

- A propriedade TRANSPORT deve ser configurada como DIRECT

  No entanto, para que um aplicativo se conecte usando o tunelamento HTTP, a propriedade TRANSPORT deve ser configurada como DIRECTHTTP. Consulte “Usando o tunelamento HTTP” na página 937.
- A propriedade HOSTNAME deve ser configurada para o nome do host ou o endereço IP do sistema no qual o broker está em execução
- A propriedade PORT deve ser configurada para o número da porta na qual o broker está atendendo conexões em tempo real..

Um aplicativo pode configurar essas propriedades dinamicamente no tempo de execução usando as extensões JMS do IBM ou as extensões JMS do WebSphere MQ . Como alternativa, se o connection factory for um objeto administrado, um administrador poderá configurar essas propriedades usando a ferramenta de administração JMS do WebSphere MQ ou o Explorer do WebSphere MQ

Para obter informações sobre as propriedades e os métodos usados pelos aplicativos para configurar seus valores, consulte Propriedades de IBM WebSphere MQ classes for JMS objetos Para obter informações sobre como usar a ferramenta de administração JMS WebSphere MQ , consulte “Usando a ferramenta de administração JMS do WebSphere MQ” na página 946. Para obter informações sobre como usar o WebSphere MQ Explorer, consulte a ajuda fornecida com o WebSphere MQ Explorer.

Se você tiver qualquer um dos seguintes requisitos, as classes WebSphere MQ para JMS requerem configuração adicional:

- Se desejar que um aplicativo se conecte ao broker usando a autenticação Secure Sockets Layer (SSL)
- Se desejar que um aplicativo se conecte ao broker usando o tunelamento HTTP
- Se desejar que um aplicativo se conecte ao broker por meio de um servidor proxy
- Se desejar que as mensagens sejam entregues a um consumidor de mensagens usando multicast

As seções a seguir descrevem como configurar classes do WebSphere MQ para JMS para cada um desses requisitos.

Usando autenticação Secure Sockets Layer (SSL)

A autenticação SSL pode ser usada em uma conexão em tempo real com um broker Somente a autenticação é suportada para este tipo de conexão Não é possível usar SSL para criptografar e

decriptografar os dados da mensagem que fluem entre o aplicativo e o broker ou para detectar violação dos dados.

Observe a diferença entre essa situação e que quando um aplicativo se conecta a um gerenciador de fila no modo cliente No último caso, é possível usar o suporte SSL do WebSphere MQ para criptografar e decriptografar os dados da mensagem que fluem entre o aplicativo e o gerenciador de fila e para detectar violação dos dados, bem como para fornecer autenticação.

Se desejar proteger os dados da mensagem em uma conexão em tempo real com um broker, será possível usar a função fornecida pelo broker. É possível designar um valor de qualidade de proteção (QoP) para cada tópico com mensagens que você deseja proteger. Portanto, é possível selecionar um nível diferente de proteção de mensagens para cada tópico.. Para obter mais informações sobre a proteção de mensagens fornecida por um broker, consulte a documentação do produto *WebSphere Event Broker* ou *WebSphere Documentação do produto Message Broker*.

Para usar a autenticação SSL em uma conexão em tempo real com um broker, a propriedade DIRECTAUTH do connection factory deve ser configurada como CERTIFICATE

Se desejar usar SSL para autenticação mútua, a propriedade Tipo de protocolo de autenticação do broker deverá especificar a opção R para SSL simétrico. Se desejar usar SSL apenas para autenticar o broker, a propriedade Tipo de protocolo de autenticação do broker deverá especificar a opção S para SSL assimétrico. Mas, nesse caso, o aplicativo deve se conectar ao broker chamando createConnection() com um ID do usuário e senha como parâmetros, como no exemplo a seguir:

```
factory.createConnection("user1", "user1pw");
```

O broker então usa o ID do usuário e a senha, em vez de SSL, para autenticar o aplicativo. Para obter mais informações sobre como configurar o broker para autenticação SSL, consulte a *WebSphere Documentação do produto Event Broker* ou *WebSphere Documentação do produto Message Broker*.

Notes:

1. O valor da propriedade DIRECTAUTH determina se a autenticação SSL é usada em uma conexão em tempo real com um broker, não o valor da propriedade SSLCIPHERSUITE.
2. Quando a autenticação SSL é usada em uma conexão em tempo real com um broker, as propriedades SSLPEERNAME e SSLCRL são usadas para executar as mesmas verificações que aquelas executadas quando um aplicativo se conecta a um gerenciador de filas no modo de cliente
3. As classes do WebSphere MQ para JMS podem usar a mesma configuração do keystore e do armazenamento confiável do Java Secure Socket Extension (JSSE) para fornecer o suporte SSL em uma das situações a seguir:
   - Quando um aplicativo usa uma conexão em tempo real com um broker
   - Quando um aplicativo se conecta a um gerenciador de filas no modo cliente

Usando o tunelamento HTTP

Um WebSphere MQ classes para aplicativo JMS pode se conectar a um broker usando tunelamento HTTP, o que significa que o aplicativo se conecta ao broker usando o protocolo HTTP como se estivesse se conectando a um website.

Para usar o tunelamento HTTP em uma conexão em tempo real com um broker, a propriedade TRANSPORT do connection factory deve ser configurada como DIRECTHTTP..

O tunelamento HTTP não pode ser usado em conjunto com a autenticação SSL, a conexão por meio de um servidor proxy ou a entrega de mensagens usando multicast. A versão suportada do protocolo HTTP é 1.0.. HTTP versão 1.1 não é suportado.

Conectando por meio de um servidor proxy

Uma classe WebSphere MQ para aplicativo JMS pode usar uma conexão em tempo real com um broker conectando-se por meio de um servidor proxy. WebSphere MQ classes para JMS se conecta diretamente

ao servidor proxy e usa o protocolo da Internet definido no RFC 2817 para pedir ao servidor proxy para encaminhar o pedido de conexão para o broker.

Para se conectar a um broker por meio de um servidor proxy, as seguintes propriedades do connection factory devem ser configuradas:

- A propriedade PROXYHOSTNAME deve ser configurada para o nome do host ou endereço IP do sistema no qual o servidor proxy está em execução..
- A propriedade PROXYPORT deve ser configurada para o número da porta na qual o servidor proxy está atendendo

Se a propriedade PROXYHOSTNAME não estiver configurada, ou estiver configurada para a sequência de caracteres vazia, as classes do WebSphere MQ para JMS tentarão se conectar diretamente ao broker usando apenas as propriedades HOSTNAME e PORT e não tentarão se conectar por meio de um servidor proxy

Entregando mensagens usando multicast

Usando uma conexão em tempo real com um broker, as mensagens podem ser entregues a um consumidor de mensagem usando multicast

Para ativar multicast, a propriedade MULTICAST do objeto Tópico deve ser configurada para a opção multicast necessária. Como alternativa, se a propriedade MULTICAST do objeto Topic for configurada como ASCF, a propriedade MULTICAST do connection factory deverá ser configurada como a opção multicast necessária

As classes WebSphere MQ para JMS suportam os protocolos multicast Packet Transfer Layer (PTL) e Pragmatic General Multicast (PGM) e incluem suporte para ambas as implementações do protocolo PGM, PGM/IP e PGM UDP encapsulados. No entanto, o suporte PGM/IP está disponível apenas nas plataformas a seguir:

- AIX (somente 32 bits)
- Linux (plataforma x86)
- Linux (plataforma zSeries , somente 32 bits)
- Solaris SPARC (apenas 32 bits)
- Windows (apenas 32 bits)
- z/OS

WebSphere MQ classes para JMS Application Server Facilities

Este tópico descreve como as classes WebSphere MQ para JMS implementam a classe ConnectionConsumer e a funcionalidade avançada na classe Session. Ele também resume a função de um conjunto de sessões do servidor.

As classes do WebSphere MQ para JMS suportam as Application Server Facilities (ASF) especificadas na *Java Message Service Specification, Versão 1.1* (consulte o website Java da Sun em https://java.sun.com). Esta especificação identifica três funções nesse modelo de programação:

- **O provedor JMS** fornece ConnectionConsumer e funcionalidade avançada de Sessão.
- **O servidor de aplicativos** fornece a funcionalidade ServerSessionPool e ServerSession.
- **O aplicativo cliente** usa a funcionalidade que o provedor JMS e o servidor de aplicativos fornecem

As informações neste tópico não se aplicarão se um aplicativo usar uma conexão em tempo real a um broker.

O JMS ConnectionConsumer

A interface ConnectionConsumer fornece um método de alto desempenho para entregar mensagens simultaneamente para um conjunto de encadeamentos.

A especificação JMS permite que um servidor de aplicativos se integre com uma implementação JMS usando a interface `ConnectionConsumer` . Este recurso fornece processamento simultâneo de mensagens. Geralmente, um servidor de aplicativos cria um conjunto de encadeamentos e a implementação JMS disponibiliza mensagens para esses encadeamentos.. Um servidor de aplicativos JMS (como o WebSphere Application Server) pode usar esse recurso para fornecer funcionalidade de sistema de mensagens de alto nível, como beans acionados por mensagens.

Os aplicativos normais não usam o ConnectionConsumer, mas clientes JMS especialistas podem usá-lo. Para esses clientes, a ConnectionConsumer fornece um método de alto desempenho para entregar mensagens simultaneamente para um conjunto de encadeamentos. Quando uma mensagem chega em uma fila ou em um tópico, o JMS seleciona um encadeamento do conjunto e entrega um lote de mensagens para ele. Para isso, o JMS executa um método MessageListenerassociado `onMessage()` .

É possível obter o mesmo efeito, construindo vários objetos Session e MessageConsumer, cada um com um MessageListener registrado. No entanto, o ConnectionConsumer fornece melhor desempenho, menos uso de recursos e maior flexibilidade. Em particular, menos objetos Session são necessários.

Planejando um aplicativo com ASF

Esta seção informa como planejar um aplicativo, incluindo:

- “Princípios gerais para o sistema de mensagens ponto a ponto usando o ASF” na página 939
- “Princípios gerais para o sistema de mensagens de publicação/assinatura usando o ASF” na página 940
- “Removendo mensagens da fila em ASF” na página 941
- Manipulando mensagens suspeitas no ASF. Consulte o “Manipulando mensagens suspeitas no IBM WebSphere MQ classes for JMS” na página 901.

Princípios gerais para o sistema de mensagens ponto a ponto usando o ASF

Use este tópico para obter informações gerais sobre o sistema de mensagens ponto a ponto usando o ASF.

Quando um aplicativo cria um ConnectionConsumer a partir de um objeto QueueConnection , ele especifica um objeto de fila JMS e uma sequência de seletores O ConnectionConsumer, em seguida, começa a fornecer mensagens para sessões no ServerSessionPool associado. As mensagens chegam na fila e se corresponderem ao seletor, elas serão entregues para sessões no ServerSessionPool associado.

Nos termos do WebSphere MQ , o objeto da fila refere-se a um QLOCAL ou QALIAS no gerenciador de filas locais. Se for um QALIAS, esse QALIAS deverá se referir a um QLOCAL. O WebSphere MQ QLOCAL totalmente resolvido é conhecido como o *QLOCAL subjacente*. Um ConnectionConsumer será considerado como *ativo* se ele não estiver fechado e seu pai QueueConnection for iniciado.

É possível para vários ConnectionConsumers, cada um com diferentes seletores, ser executado no mesmo QLOCAL subjacente. Para manter o desempenho, as mensagens indesejadas não devem se acumular na fila. As mensagens indesejadas são aquelas as quais nenhum ConnectionConsumer ativo possui um seletor correspondente. É possível configurar o QueueConnectionFactory para que essas mensagens indesejadas sejam removidas da fila (para obter detalhes, consulte “Removendo mensagens da fila em ASF” na página 941). É possível configurar esse comportamento em uma de duas maneiras:

- Use a ferramenta de administração JMS para configurar o Factory QueueConnectionpara MRET (NO).
- Em seu programa, use:

```
MQQueueConnectionFactory.setMessageRetention(WMQConstants.WMQ_MRET_NO)
```

Se você não mudar esta configuração, o padrão será reter essas mensagens indesejadas na fila.

Ao configurar o gerenciador de fila do WebSphere MQ , considere os pontos a seguir:

- O QLOCAL subjacente deve estar ativado para a entrada compartilhada. Para fazer isso, use o seguinte comando MQSC:

```
ALTER QLOCAL(your.qlocal.name) SHARE GET(ENABLED)
```

- Seu gerenciador de filas deve ter uma fila de mensagens não entregues ativada. Se um ConnectionConsumer tiver um problema quando ele colocar uma mensagem na fila de mensagens não entregues, a entrega da mensagem a partir do QLOCAL subjacente será interrompida. Para definir uma fila de mensagens não entregues, use:

```
ALTER QMGR DEADQ(your.dead.letter.queue.name)
```

- O usuário que executa o ConnectionConsumer deve ter autoridade para executar MQOPEN com MQOO_SAVE_ALL_CONTEXT e MQOO_PASS_ALL_CONTEXT. Para obter detalhes, consulte a documentação do WebSphere MQ para sua plataforma específica.
- Se mensagens indesejadas forem deixadas na fila, elas irão degradar o desempenho do sistema. Portanto, planeje seus seletores de mensagens para que entre eles, o ConnectionConsumers removerá todas as mensagens da fila.

Para obter detalhes sobre comandos MQSC, consulte o Referência MQSC.

Princípios gerais para o sistema de mensagens de publicação/assinatura usando o ASF

ConnectionConsumers recebem mensagens para um Tópico especificado. Um ConnectionConsumer pode ser durável ou não durável. Deve-se especificar qual fila ou quais filas o ConnectionConsumer usa.

Quando um aplicativo criar um ConnectionConsumer a partir de um objeto TopicConnection, ele especificará um objeto Tópico e uma sequência do seletor. O ConnectionConsumer, em seguida, começa a receber mensagens que correspondem ao seletor nesse tópico, incluindo todas as publicações retidas para o tópico assinado.

Alternativamente, um aplicativo pode criar um ConnectionConsumer durável associado a um nome específico. Este ConnectionConsumer recebe mensagens que foram publicadas no Tópico desde o ConnectionConsumer durável estar ativo pela última vez. Ele recebe todas as mensagens que correspondem ao seletor no Tópico. No entanto, se o ConnectionConsumer estiver usando a leitura antecipada, ele poderá perder as mensagens não persistentes que estão no buffer do cliente quando ele for fechado.

Se as classes do WebSphere MQ para JMS estiverem no modo de migração do provedor de sistemas de mensagens WebSphere MQ , uma fila separada será usada para assinaturas não duráveis do ConnectionConsumer .A opção configurável CCSUB no TopicConnectionFactory especifica a fila a ser usada. Normalmente, o CCSUB especifica uma fila única para uso por todos os ConnectionConsumers que usam o mesmo TopicConnectionFactory. Entretanto, é possível fazer cada ConnectionConsumer gerar uma fila temporária especificando um prefixo de nome da fila seguidos de um asterisco (*).

Se as classes do WebSphere MQ para JMS estiverem no modo de migração do provedor de sistemas de mensagens WebSphere MQ , a propriedade CCDSUB do Tópico especifica a fila a ser usada para assinaturas duráveis. Novamente, esta pode ser uma fila que já existe ou um prefixo de nome de fila seguido por um asterisco (*). Se você especificar uma fila que já existe, todos os ConnectionConsumers duráveis que assinam o Tópico utilizam esta fila. Se você especificar um prefixo de nome da fila seguido de um asterisco (*), uma fila será gerada pela primeira vez que um ConnectionConsumer durável for criado com um nome específico. Esta fila será reutilizada posteriormente quando um ConnectionConsumer durável for criado com o mesmo nome.

Ao configurar o gerenciador de fila do WebSphere MQ , considere os pontos a seguir:

- Seu gerenciador de filas deve ter uma fila de mensagens não entregues ativada. Se um ConnectionConsumer tiver um problema quando ele colocar uma mensagem na fila de mensagens não entregues, a entrega da mensagem a partir do QLOCAL subjacente será interrompida. Para definir uma fila de mensagens não entregues, use:

```
ALTER QMGR DEADQ(your.dead.letter.queue.name)
```

- O usuário que executa o ConnectionConsumer deve ter autoridade para executar MQOPEN com MQOO_SAVE_ALL_CONTEXT e MQOO_PASS_ALL_CONTEXT. Para obter detalhes, consulte a documentação do WebSphere MQ para a sua plataforma
- É possível otimizar o desempenho para um ConnectionConsumer individual criando uma fila separada e dedicada para isso. Este é o custo de uso de recursos extras.

Removendo mensagens da fila em ASF

Quando um aplicativo usa ConnectionConsumers, o JMS pode precisar remover mensagens da fila em várias situações.

Estas situações são as seguintes:

Mensagem formatada incorretamente
Uma mensagem pode chegar que o JMS não pode analisar.

Mensagem suspeita
Uma mensagem pode atingir o limite de restauração, mas o ConnectionConsumer não deve ser recolocado na fila de restauração.

Nenhum ConnectionConsumer interessado
Para o sistema de mensagens ponto a ponto, quando o QueueConnectionFactory for configurado para que ele não retenha mensagens indesejadas, uma mensagem chega que não é desejada por quaisquer dos ConnectionConsumers.

Nessas situações, o ConnectionConsumer tenta remover a mensagem da fila. As opções de disposição no campo de relatório de MQMD da mensagem configura o comportamento exato. Estas opções são:

MQRO_DEAD_LETTER_Q
A mensagem é enfileirada para a fila de mensagens não entregues do gerenciador de filas. Esse é o padrão.

MQRO_DISCARD_MSG
A mensagem será descartada.

O ConnectionConsumer também gera uma mensagem de relatório e isso também depende do campo de relatório MQMD da mensagem. Esta mensagem é enviada para o ReplyToQ da mensagem no ReplyToQmgr. Se houver um erro enquanto a mensagem de relatório estiver sendo enviada, a mensagem será enviada para a fila de mensagens não entregues. As opções de relatório de exceção no campo de relatório do MQMD da mensagem configura os detalhes da mensagem de relatório. Estas opções são:

MQRO_EXCEPTION
Uma mensagem de relatório que contém o MQMD da mensagem original é gerada. Ela não contém nenhum dado do corpo da mensagem.

MQRO_EXCEPTION_WITH_DATA
Uma mensagem de relatório que contém o MQMD, quaisquer cabeçalhos MQ e 100 bytes de dados do corpo é gerada.

MQRO_EXCEPTION_WITH_FULL_DATA
Uma mensagem de relatório que contém todos os dados da mensagem original é gerada.

padrão
Nenhuma mensagem de relatório é gerada.

Quando mensagens de relatório são geradas, as seguintes opções são favorecidas:

- MQRO_NEW_MSG_ID
- MQRO_PASS_MSG_ID
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- MQRO_PASS_CORREL_ID

Se uma mensagem suspeita não puder ser enfileirada novamente, talvez porque a fila de mensagens não entregues esteja cheia ou a autorização está especificada incorretamente, o que acontece depende da persistência da mensagem. Se a mensagem for não persistente, a mensagem será descartada e nenhuma mensagem de relatório será gerada. Se a mensagem for persistente, a entrega de mensagens para

todos os consumidores de conexão atendendo nesse destino parará. Estes consumidores de conexão devem ser fechados e o problema resolvido antes que possam ser recriados e a entrega de mensagens reiniciada.

É importante definir uma fila de mensagens não entregues e verificá-la regularmente para assegurar que nenhum problema ocorra. Particularmente, certifique-se de que a fila de mensagens não entregues não tenha atingido sua profundidade máxima e que seu tamanho máximo de mensagem é grande o suficiente para todas as mensagens.

Quando uma mensagem é reenfileirada para a fila de devoluções, ela é precedida por um cabeçalho de devoluções (MQDLH) WebSphere MQ . Consulte MQDLH – Cabeçalho de mensagens não entregues para obter detalhes sobre o formato do MQDLH. É possível identificar mensagens que um ConnectionConsumer colocou na fila de mensagens não entregues ou mensagens de relatório que um ConnectionConsumer tenha gerado pelos seguintes campos:

- PutApplType é `MQAT_JAVA (0x1C)`
- PutApplO nome é "`MQ JMS ConnectionConsumer`"

Estes campos estão no MQDLH de mensagens na fila de mensagens não entregues e no MQMD de mensagens de relatório. O campo de feedback do MQMD e o campo Reason do MQDLH contêm um código que descreve o erro. Para obter detalhes sobre esses códigos, consulte “Códigos de razão e de feedback em ASF” na página 943. Outros campos estarão conforme descrito em MQDLH – Cabeçalho de mensagens não entregues.

Manipulando mensagens suspeitas no ASF

No Application Server Facilities, a manipulação de mensagens suspeitas é manipulada de forma um pouco diferente de qualquer outro lugar nas classes do WebSphere MQ para JMS.

Para obter informações sobre a manipulação de mensagens suspeitas nas classes do WebSphere MQ para JMS, consulte “Manipulando mensagens suspeitas no IBM WebSphere MQ classes for JMS” na página 901

Ao usar o Application Server Facilities (ASF), o ConnectionConsumer, em vez de o MessageConsumer, processa mensagens suspeitas. O ConnectionConsumer recoloca mensagens na fila de acordo com as propriedades BackoutThreshold e BackoutRequeueQName da fila.

Quando um aplicativo usa ConnectionConsumers, as circunstâncias nas quais uma mensagem é restaurada dependem da sessão que o servidor de aplicativos fornece:

- Quando a sessão é não transacionada, com AUTO_ACKNOWLEDGE ou DUPS_OK_ACKNOWLEDGE, uma mensagem é restaurada somente após um erro do sistema ou se o aplicativo for finalizado inesperadamente.
- Quando a sessão é não transacionada com CLIENT_ACKNOWLEDGE, mensagens não reconhecidas podem ser restauradas pelo servidor de aplicativos chamando Session.recover().

  Geralmente, a implementação do cliente do MessageListener ou do servidor de aplicativos chama Message.acknowledge(). Message.acknowledge() reconhece todas as mensagens entregues na sessão até o momento.
- Quando a sessão é transacionada, as mensagens não reconhecidas podem ser restauradas pelo servidor de aplicativos chamando Session.rollback().
- Se o servidor de aplicativos fornecer uma XASession, as mensagens serão confirmadas ou restauradas, dependendo de uma transação distribuída. O servidor de aplicativos assume a responsabilidade por concluir a transação.

O provedor JMS integrado no WebSphere Application Server, versão 5.0 e versão 5.1 manipula mensagens suspeitas de uma maneira diferente daquela descrita para as classes WebSphere MQ para JMS. Para obter informações sobre como o provedor JMS integrado manipula mensagens suspeitas, consulte a documentação relevante do produto WebSphere Application Server.

Manipulação de Erros

Esta seção abrange diversos aspectos de manipulação de erros, incluindo “Recuperação de condições de erro no ASF” na página 943 e “Códigos de razão e de feedback em ASF” na página 943.

Recuperação de condições de erro no ASF

Se um ConnectionConsumer receber um erro grave, a entrega da mensagem para todos os ConnectionConsumers com um interesse no mesmo QLOCAL será interrompida. Quando isso ocorrer, qualquer ExceptionListener registrado com o Connection afetado será notificado. Há duas maneiras nas quais um aplicativo pode se recuperar destas condições de erro.

Geralmente, um grave erro desta natureza ocorrerá se o ConnectionConsumer não puder reenfileirar uma mensagem na fila de mensagens não entregues ou se ocorrer um erro ao ler mensagens a partir de QLOCAL.

Como qualquer ExceptionListener registrado com o Connection afetado é notificado, é possível usá-las para identificar a causa do problema. Em alguns casos, o administrador do sistema deve intervir para resolver o problema.

Use uma das técnicas a seguir para se recuperar destas condições de erro:

- Chame `close()` em todos os ConnectionConsumers afetados. O aplicativo poderá criar novos ConnectionConsumers somente após todos os ConnectionConsumers afetados serem fechados e quaisquer problemas do sistema resolvidos.
- Chame `stop()` em todas as Connections afetadas. Quando todas as Connections forem interrompidas e quaisquer problemas no sistema resolvidos, o aplicativo poderá efetuar `start()` de suas Connections com sucesso.

Códigos de razão e de feedback em ASF

Use códigos de razão e de feedback para determinar a causa de um erro. Códigos de razão comuns gerados pelo ConnectionConsumer são fornecidos aqui.

Para determinar a causa de um erro, use as seguintes informações:

- O código de feedback em todas as mensagens de relatório
- O código de razão no MQDLH de todas as mensagens na fila de mensagens não entregues

ConnectionConsumers gera os códigos de razão a seguir.

MQRC_BACKOUT_THRESHOLD_REACHED (0x93A; 2362)

Causa

A mensagem alcançou o limite de restauração definido no QLOCAL, mas nenhuma fila de restauração está definida.

Nas plataformas em que não é possível definir a Fila de Restauração, a mensagem atingiu o limite de restauração definido pelo JMS de 20..

Ação

Se isso não for desejado, defina a fila de restauração para o QLOCAL relevante. Além disso, procure pela causa das múltiplas restaurações.

MQRC_MSG_NOT_MATCHED (0x93B; 2363)

Causa

No o sistema de mensagens ponto a ponto, há uma mensagem que não corresponde a nenhum dos seletores para os ConnectionConsumers que monitoram a fila. Para manter o desempenho, a mensagem é enfileirada novamente na fila de mensagens não entregues.

Ação

Para evitar essa situação, assegure-se de que ConnectionConsumers usando a fila forneçam um conjunto de seletores que lidam com todas as mensagens ou configure QueueConnectionFactory para reter as mensagens.

Como alternativa, investigue a origem da mensagem.

MQRC_JMS_FORMAT_ERROR (0x93C; 2364)

Causa
O JMS não pode interpretar a mensagem na fila.

Ação
Investigue a origem da mensagem. O JMS normalmente entrega mensagens de um formato inesperado como um BytesMessage ou TextMessage Ocasionalmente, isso falhará se a mensagem for muito mal formatada.

Outros códigos que aparecem nesses campos são causados por uma tentativa fracassada de enfileirar novamente a mensagem em uma fila de restauração. Nesta situação, o código descreve o motivo pelo qual o reenfileiramento falhou. Para diagnosticar a causa desses erros, consulte Códigos de razão da API.

Se a mensagem de relatório não puder ser colocada na ReplyToQ, ela será colocada na fila de mensagens não entregues. Nesta situação, o campo de feedback do MQMD estará concluído conforme descrito neste tópico. O campo razão no MQDLH explica o motivo pelo qual a mensagem de relatório não pôde ser colocada na ReplyToQ.

A função de um conjunto de sessões do servidor em AFS

Este tópico resume a função de um conjunto de sessões do servidor.

Figura 165 na página 945 resume os princípios da funcionalidade ServerSessionPool e ServerSession.

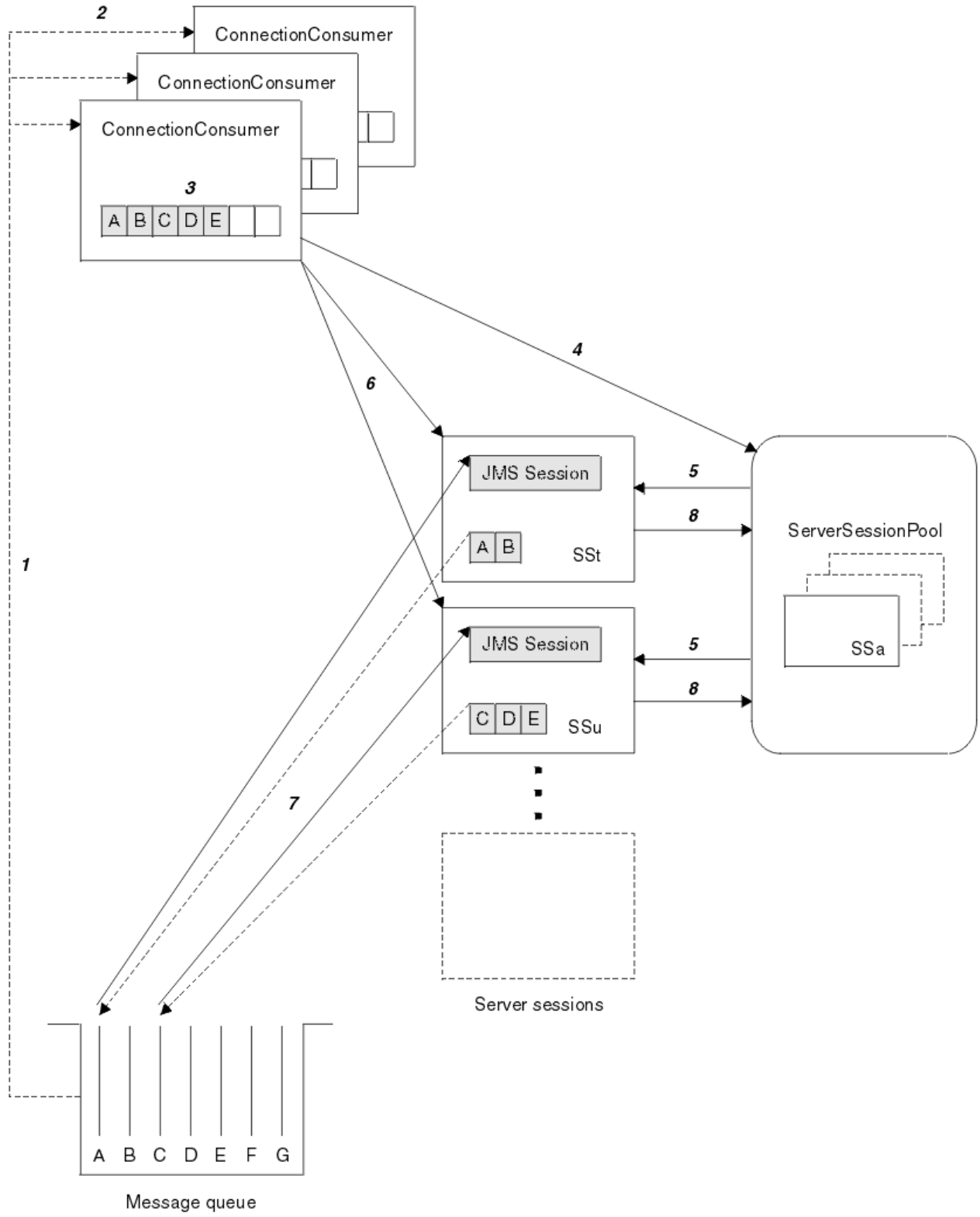


Figura 165. Funcionalidade ServerSessionPool e ServerSession

1. O ConnectionConsumers obtém referências de mensagem a partir da fila.
2. Cada ConnectionConsumer seleciona referências de mensagens específicas.
3. O buffer ConnectionConsumer contém as referências de mensagens selecionadas.
4. As solicitações ConnectionConsumer um ou mais ServerSessions a partir do ServerSessionPool.

5. ServerSessions são alocadas a partir do ServerSessionPool.
6. O ConnectionConsumer designa referências de mensagens ao ServerSessions e inicia os encadeamentos em execução ServerSession.
7. Cada ServerSession recupera suas mensagens referenciadas a partir da fila. Ele os transmite para o método onMessage do MessageListener que está associado à Sessão JMS.
8. Após concluir seu processamento, o ServerSession será retornado ao conjunto.

Um servidor de aplicativos geralmente fornece a funcionalidade ServerSessionPool e ServerSession.

Usando a ferramenta de administração JMS do WebSphere MQ

Use a ferramenta de administração para definir as propriedades de oito tipos de classes WebSphere MQ para objeto JMS e para armazená-las em um namespace JNDI. Os aplicativos podem, então, usar JNDI para recuperar esses objetos administrados do namespace.

Os objetos JMS de classes do WebSphere MQ que você pode administrar usando a ferramenta são:

- MQConnectionFactory
- MQQueueConnectionFactory
- MQTopicConnectionFactory
- MQQueue
- MQTopic
- MQXAConnectionFactory
- MQXAQueueConnectionFactory
- MQXATopicConnectionFactory

Para obter detalhes sobre esses objetos, consulte “Administrando Objetos JMS” na página 951

Os tipos de propriedades e valores necessários para usar essa ferramenta são listados em Propriedades de IBM WebSphere MQ classes for JMS objetos.

A ferramenta também permite que os administradores manipulem subcontextos de namespace de diretório dentro da JNDI. Consulte “Manipulando subcontexto com a ferramenta de administração JMS do WebSphere MQ” na página 950.

Também é possível criar e configurar objetos administrados pelo JMS com o WebSphere MQ Explorer.

Chamando a ferramenta de administração do IBM WebSphere MQ classes for JMS

A ferramenta de administração tem uma interface de linha de comandos. É possível usar esta ferramenta interativamente ou usá-la para iniciar um processo em lote.

O modo interativo fornece um prompt de comandos no qual é possível digitar comandos de administração. No modo de lote, o comando para iniciar a ferramenta inclui o nome de um arquivo que contém um script de comando de administração.

Modo interativo

Para iniciar a ferramenta no modo interativo, insira o comando:

```
JMSAdmin [-t] [-v] [-cfg config_filename]
```

em que:

-t
Ativa o rastreio (o padrão é rastreio desligado)

O arquivo de rastreio é gerado em "%MQ_JAVA_DATA_PATH%"\errors (Windows) ou /var/mqm/trace (UNIX). O nome do arquivo de rastreio está no formato:

mqjms_*PID*.trc

em que *PID* é o ID do processo da JVM.

-v
Produz saída detalhada (o padrão é saída terse)

-cfg config_filename
Nomeia um arquivo de configuração alternativo. Se esse parâmetro for omitido, o arquivo de configuração padrão, JMSAdmin.config, será usado (Consulte "Configurando a Ferramenta de Administração JMS" na página 947)

Um prompt de comandos é exibido, o que indica que a ferramenta está pronta para aceitar comandos de administração. Esse prompt aparece inicialmente como:

```
InitCtx>
```

indicando que o contexto atual (ou seja, o contexto JNDI ao qual todas as operações de nomenclatura e diretório atualmente se referem) é o contexto inicial definido no parâmetro de configuração PROVIDER_URL (consulte "Configurando a Ferramenta de Administração JMS" na página 947).

À medida que você percorre o espaço de nomes de diretório, o prompt muda para refletir isso, de forma que o prompt sempre exibe o contexto atual.

modo Batch

Para iniciar a ferramenta no modo em lote, insira o comando:

```
JMSAdmin <test.scp
```

em que *test.scp* é um arquivo de script que contém comandos de administração (consulte "Comandos de administração na ferramenta de administração JMS do WebSphere MQ" na página 949). O último comando no arquivo deve ser o comando END.

Configurando a Ferramenta de Administração JMS

A ferramenta de Administração JMS do WebSphere MQ usa um arquivo de configuração para configurar valores de determinadas propriedades. Um arquivo de amostra é fornecido e é possível customizá-lo para seu sistema.

O arquivo de configuração é um arquivo de texto simples que consiste em um conjunto de pares de valores de chave, separados pelo sinal de igual (=). Isso é mostrado no seguinte exemplo:

```
#Set the service provider
    INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
#Set the initial context
    PROVIDER_URL=ldap://polaris/o=ibm_us,c=us
#Set the authentication type
    SECURITY_AUTHENTICATION=none
```

(Um # na primeira coluna da linha indica um comentário ou uma linha que não é usada.)

Um arquivo de configuração de amostra é fornecido com o WebSphere MQ O arquivo é chamado de JMSAdmin.confige está localizado no diretório <MQ_JAVA_INSTALL_PATH>/bin Edite esse arquivo para se adequar à configuração do sistema.

Configure a ferramenta de administração com valores para as seguintes propriedades:

INITIAL_CONTEXT_FACTORY
O provedor de serviços que a ferramenta usa. Os valores suportados para essa propriedade são os seguintes:

- com.sun.jndi.ldap.LdapCtxFactory (para LDAP)
- com.sun.jndi.fscontext.RefFSContextFactory (para contexto de sistema de arquivos)

Também é possível usar um factory InitialContextque não está na lista anterior.. Consulte "Usando um Factory InitialContextnão listado com a ferramenta de administração JMS do WebSphere MQ" na página 948 para obter mais detalhes.

PROVIDER_URL
A URL do contexto inicial da sessão; a raiz de todas operações JNDI realizadas pela ferramenta. Dois formatos dessa propriedade são suportados:

- ldap://hostname/contextname
- file:[drive:]/pathname

O formato da URL LDAP pode variar, dependendo do seu provedor LDAP. Veja a documentação do LDAP para obter mais informações.

SECURITY_AUTHENTICATION
Se JNDI passa credenciais de segurança para seu provedor de serviços. Essa propriedade é usada apenas quando um provedor de serviços LDAP é usado.. Essa propriedade pode ter um dos três valores:

- nenhum (autenticação anônima)
- simples (autenticação simples)
- CRAM-MD5 (mecanismo de autenticaçãoCRAM-MD5)

Se um valor válido não for fornecido, o padrão da propriedade será nenhum. Consulte "Configurando a Segurança para a Ferramenta de Administração JMS" na página 949 para obter mais detalhes sobre segurança com a ferramenta de administração.

Essas propriedades são definidas em um arquivo de configuração Ao chamar a ferramenta, é possível especificar essa configuração usando o parâmetro da comandos `-cfg` , conforme descrito em "Chamando a ferramenta de administração do IBM WebSphere MQ classes for JMS" na página 946. Se você não especificar um nome de arquivo de configuração, a ferramenta tentará carregar o arquivo de configuração padrão (`JMSAdmin.config`) Ele procura esse arquivo primeiro no diretório atual e, em seguida, no diretório `<MQ_JAVA_INSTALL_PATH>/bin` , em que `<MQ_JAVA_INSTALL_PATH>` é o caminho para suas classes do WebSphere MQ para instalação do JMS.

Usando um Factory InitialContextnão listado com a ferramenta de administração JMS do WebSphere MQ

Dois valores de InitialContextFactory são suportados. É possível usar outros contexto JNDI configurando parâmetros no arquivo de configuração de administração JMS.

É possível usar a ferramenta de administração para conectar a contextos JNDI diferentes daqueles listados em "Configurando a Ferramenta de Administração JMS" na página 947 usando três parâmetros definidos no arquivo de configuração JMSAdmin.

Para usar um InitialContextFactory diferente:

1. Configure a propriedade INITIAL_CONTEXT_FACTORY para o nome de classe necessário
2. Defina o comportamento do factory InitialContextusando as propriedades USE_INITIAL_DIR_CONTEXT, NAME_PREFIX e NAME_READABILITY_MARKER..

As configurações para essas propriedades são descritas nos comentários do arquivo de configuração de amostra.

Não é necessário definir as três propriedades listada aqui se você usar um dos valores INITIAL_CONTEXT_FACTORY suportados No entanto, é possível fornecer a eles valores para substituir os padrões do sistema.. Se você omitir uma ou mais das três propriedades InitialContextFactory, a ferramenta de administração fornecerá padrões adequados com base nos valores das outras propriedades.

Configurando a Segurança para a Ferramenta de Administração JMS

Use a propriedade SECURITY_AUTHENTICATION para determinar se as credenciais de segurança são passadas ao provedor de serviços.

A propriedade SECURITY_AUTHENTICATION é descrita em "Configurando a Ferramenta de Administração JMS" na página 947.. O seu efeito é o seguinte:

- Se você configurar esse parâmetro para `nenhum`, a JNDI não transmitirá nenhuma credencial de segurança para o provedor de serviços e *autenticação anônima* será executada.
- Se você configurar o parâmetro para `simple` ou `CRAM-MD5`, as credenciais de segurança serão transmitidas por JNDI para o provedor de serviços subjacente. Essas credenciais de segurança estão na forma de um DN do Usuário (Nome Distinto do Usuário) e senha.

Se as credenciais de segurança forem necessárias, você será solicitado a informá-las quando a ferramenta for inicializada. Evite isso definindo as propriedades PROVIDER_USERDN e PROVIDER_PASSWORD no arquivo de configuração JMSAdmin

Nota: Se você não usar essas propriedades, o texto digitado, *, incluindo a senha*, será repetido na tela. Isso pode ter implicações de segurança.

A ferramenta não faz autenticação em si; a tarefa é delegada ao servidor LDAP. O administrador do servidor LDAP deve configurar e manter os privilégios de acesso para diferentes partes do diretório. Veja a documentação do LDAP para obter mais informações. Se a autenticação falhar, a ferramenta exibirá uma mensagem de erro apropriada e será encerrada.

Informações mais detalhadas sobre segurança e JNDI estão na documentação no website Java da Sun (https://java.sun.com).

Comandos de administração na ferramenta de administração JMS do WebSphere MQ

A ferramenta de administração aceita comandos que consistem em um verbo de administração e seus parâmetros apropriados.

Quando o prompt de comandos é exibido, a ferramenta está pronta para aceitar comandos Os comandos de administração são geralmente do seguinte formato:

```
verb [param]*
```

em que **verb** é um dos verbos de administração listados na Tabela 133 na página 949. Todos os comandos válidos contidos em um verbo, que aparecem no início do comando no formato padrão ou abreviado.

Os parâmetros que um verbo pode usar dependem do verbo. Por exemplo, o verbo `END` não pode usar parâmetros, mas o verbo `DEFINE` pode usar qualquer número de parâmetros. Os detalhes dos verbos que tomam pelo menos um parâmetro são discutidos nos tópicos relacionados.

Tabela 133. Verbos de Administração

Verb	Formato curto	Descrição
ALTER	ALT	Mudar pelo menos uma das propriedades de um objeto administrado
DEFINE	DEF	Criar e armazenar um objeto administrado ou criar um subcontexto
DISPLAY	DIS	Exibir as propriedades de um ou mais objetos administrados armazenados ou o conteúdo do contexto atual
EXCLUIR	EXCL	Remover um ou mais objetos administrados do espaço de nomes ou remover um subcontexto vazio

Tabela 133. Verbos de Administração (continuação)		
Verb	**Formato curto**	**Descrição**
ALTERAÇÃO	TRC	Alterar o contexto atual, permitindo que o usuário atravesse o espaço de nomes de diretório em qualquer parte abaixo do contexto inicial (liberação de segurança pendente)
CÓPIA	CP	Fazer uma cópia de um objeto administrado armazenado, armazenando-o com um nome alternativo
MOVER	MV	Alterar o nome com o qual um objeto administrado é armazenado
END		Fechar a ferramenta de administração

Os nomes dos verbos não fazem distinção entre maiúsculas e minúsculas.

Geralmente, para finalizar comandos, pressione a tecla de retorno de linha. No entanto, você pode substituir isso digitando o sinal de mais (+) diretamente antes do retorno de linha. Isso permite digitar comandos de várias linhas, como mostra o exemplo a seguir:

```
DEFINE Q(BookingsInputQueue) +
       QMGR(QM.POLARIS.TEST) +
       QUEUE(BOOKINGS.INPUT.QUEUE) +
       PORT(1415) +
       CCSID(437)
```

As linhas que começam com qualquer um dos seguintes caracteres são tratadas como comentários e são ignoradas: * # /.

Manipulando subcontexto com a ferramenta de administração JMS do WebSphere MQ

Use os verbos **CHANGE**, **DEFINE**, **DISPLAY** e **DELETE** para manipular subcontextos de namespace de diretório.

O uso desses verbos é descrito em Tabela 134 na página 950

Tabela 134. Sintaxe e Descrição de Comandos Utilizados para Manipular Subcontextos	
Sintaxe do comando	**Descrição**
DEFINE CTX(ctxName)	Tenta criar um subcontexto filho do contexto atual, com o nome ctxName. Falha se houver uma violação de segurança, se o subcontexto já existir ou se o nome fornecido for inválido.
DISPLAY CTX	Exibe o conteúdo do contexto atual. Objetos administrados são anotados com a, subcontexto com `[D]`. O tipo Java de cada objeto também é exibido..
DELETE CTX(ctxName)	Tenta excluir o contexto filho do contexto atual que possui o nome ctxName. Falhará se o contexto não for localizado, não for vazio ou se houver uma violação de segurança.

Tabela 134. Sintaxe e Descrição de Comandos Utilizados para Manipular Subcontextos (continuação)	
Sintaxe do comando	**Descrição**
CHANGE CTX(ctxName)	Altera o contexto atual, para que agora se refira ao contexto filho que possui o nome ctxName. Um dos dois valores especiais de ctxName pode ser fornecido: **=UP** move para o pai do contexto atual **=INIT** move diretamente para o contexto inicial Falhará se o contexto especificado não existir ou se houver uma violação de segurança.

Administrando Objetos JMS

Esta seção descreve os oito tipos de objeto que a ferramenta de administração pode manipular Inclui detalhes sobre cada uma de suas propriedades configuráveis e os verbos que podem manipulá-las.

Também é possível criar e configurar objetos administrados pelo JMS com o WebSphere MQ Explorer.

tipos de objeto JMS

A tabela mostra os oito tipos de objetos administrados.

A coluna Palavra-chave mostra as cadeias que podem ser substituídas por *TYPE* nos comandos mostrados em Tabela 136 na página 952

Tabela 135. Os tipos de objeto JMS que são manipulados pelo Administration Tool		
Tipo de objeto	**Palavra-chave**	**Descrição**
MQConnectionFactory	CF	A implementação do WebSphere MQ da interface JMS ConnectionFactory . Isso representa um objeto de factory para criar conexões nos domínios ponto a ponto e de publicação/assinatura.
MQQueueConnectionFactory	QCF	A implementação do WebSphere MQ da interface JMS QueueConnectionFactory. Representa um objeto de factory para criar conexões no domínio ponto a ponto.
MQTopicConnectionFactory	TCF	A implementação do WebSphere MQ da interface JMS TopicConnectionFactory. Representa um objeto de factory para criar conexões no domínio de publicação/assinatura.
MQQueue	Q	A implementação do WebSphere MQ da interface Fila JMS. Isso representa um destino para mensagens no domínio ponto-a-ponto.
MQTopic	T	A implementação do WebSphere MQ da interface de Tópico do JMS Isso representa um destino para mensagens no domínio de publicação/assinatura.

Tabela 135. Os tipos de objeto JMS que são manipulados pelo Administration Tool (continuação)		
Tipo de objeto	**Palavra-chave**	**Descrição**
MQXAConnectionFactory[“1” na página 952]	XACF	A implementação do WebSphere MQ da interface JMS XAConnectionFactory . Isso representa um objeto de factory para criar conexões nos domínios ponto a ponto e de publicação / assinatura e em que as conexões usam as versões XA de classes JMS.
MQXAQueueConnectionFactory[“1” na página 952]	XAQCF	A implementação do WebSphere MQ da interface JMS XAQueueConnectionFactory. Isso representa um objeto de factory para criar conexões no domínio ponto a ponto que usam as versões XA de classes JMS
MQXATopicConnectionFactory[“1” na página 952]	XATCF	A implementação do WebSphere MQ da interface JMS XATopicConnectionFactory. Isso representa um objeto de factory para criar conexões no domínio de publicação / assinatura que usam as versões XA de classes JMS
Nota: 1. Essas classes são fornecidas para uso por fornecedores de servidores de aplicativos. É improvável que elas sejam diretamente úteis para programadores de aplicativos.		

Verbos usados com objetos JMS

É possível usar os verbos `ALTER`, `DEFINE`, `DISPLAY`, `DELETE`, `COPY` e `MOVE` para manipular objetos administrados no namespace do diretório.

Tabela 136 na página 952 resume o uso desses verbos. Substitua *TYPE* pela palavra-chave que representa o objeto administrado necessário, conforme listado em Tabela 135 na página 951..

Tabela 136. Sintaxe e Descrição de Comandos Utilizados para Manipular Objetos Administrados	
Sintaxe do comando	**Descrição**
ALTER *TYPE*(nome) [propriedade] *	Tenta atualizar as propriedades do objeto administrado com aquelas fornecidas. Falha se houver uma violação de segurança, se o objeto especificado não puder ser localizado ou se as novas propriedades fornecidas não forem válidas.
DEFINE *TYPE*(nome) [propriedade] *	Tenta criar um objeto administrado do tipo *TYPE* com as propriedades fornecidas e armazena-o com o nome `name` no contexto atual. Falha se existir uma violação de segurança, se o nome fornecido não for válido ou se um objeto com esse nome existir, ou se as propriedades fornecidas não forem válidas.
DISPLAY *TYPE*(nome)	Exibe as propriedades do objeto administrado do tipo *TYPE*, ligado ao nome `name` no contexto atual. Falhará se o objeto não existir ou se houver uma violação de segurança.
DELETE *TYPE*(nome)	Tenta remover o objeto administrado do tipo *TYPE*, com o nome `name`, do contexto atual. Falhará se o objeto não existir ou se houver uma violação de segurança.

Tabela 136. Sintaxe e Descrição de Comandos Utilizados para Manipular Objetos Administrados (continuação)	
Sintaxe do comando	**Descrição**
COPY *TYPE*(nameA) *TYPE*(nameB)	Faz uma cópia do objeto administrado do tipo *TYPE*, com o nome nameA, nomeando a cópia como nameB Isso tudo ocorrerá no escopo do contexto atual. Falha se o objeto a ser copiado não existir, se um objeto com o nome nameB existir ou se houver uma violação de segurança.
MOVE *TYPE*(nameA) *TYPE*(nameB)	Move (renomeia) o objeto administrado do tipo *TYPE*, com o nome nameA, para nameB. Isso tudo ocorrerá no escopo do contexto atual. Falha se o objeto a ser movido não existir, se um objeto de nome nameB existir ou se houver uma violação de segurança.

Criando objetos com a ferramenta de administração JMS do WebSphere MQ

Crie objetos e armazene-os em um namespace JNDI usando o comando DEFINE,

Use a sintaxe de comando a seguir:

```
DEFINE TYPE(name) [property]*
```

Ou seja, o verbo DEFINE , seguido por uma referência do objeto administrado *TYPE*(name) , seguido por zero ou mais *propriedades* (consulte Propriedades de IBM WebSphere MQ classes for JMS objetos)...

Considerações de Nomenclatura LDAP para Objetos JMS

Para armazenar os objetos em um ambiente LDAP, dê a eles nomes que em conformidade com determinadas convenções. A ferramenta de administração pode ajudar a obedecer às convenções de nomenclatura incluindo um prefixo padrão.

Uma convenção de nomenclatura é que os nomes de objeto e subcontexto devem incluir um prefixo, como cn= (nome comum) ou ou= (unidade organizacional).

A ferramenta de administração simplifica o uso de fornecedores de serviços LDAP, permitindo que você se refira a nomes de objeto e de contexto sem um prefixo. Se você não fornecer um prefixo, a ferramenta incluirá automaticamente um prefixo padrão ao nome que você fornecer. Para LDAP, é cn=.

É possível alterar o prefixo padrão configurando a propriedade NAME_PREFIX no arquivo de configuração do JMSAdmin, conforme descrito em “Usando um Factory InitialContextnão listado com a ferramenta de administração JMS do WebSphere MQ” na página 948

Isso é mostrado no exemplo a seguir:

```
InitCtx> DEFINE Q(testQueue)

InitCtx> DISPLAY CTX

    Contents of InitCtx

      a  cn=testQueue              com.ibm.mq.jms.MQQueue

     1 Object(s)
       0 Context(s)
       1 Binding(s), 1 Administered
```

Embora o nome do objeto fornecido (testQueue) não tenha um prefixo, a ferramenta inclui automaticamente um para assegurar conformidade com a convenção de nomenclatura LDAP. Da mesma forma, o envio do comando DISPLAY Q(testQueue) também faz com que esse prefixo seja incluído

Pode ser necessário configurar o servidor LDAP para armazenar objetos Java Para obter informações para ajudar com essa configuração, consulte a documentação do servidor LDAP.

Condições de Erro de Amostra Criando um Objeto JMS

Várias condições de erro comuns podem surgir quando você cria um objeto.

Os seguintes são exemplos dessas condições de erro:

CipherSpec mapeado para CipherSuite

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) SSLCIPHERSUITE(RC4_MD5_US)
  WARNING: Converting CipherSpec RC4_MD5_US to
  CipherSuite SSL_RSA_WITH_RC4_128_MD5
```

Propriedade inválida para objeto

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) PRIORITY(4)
  Unable to create a valid object, please check the parameters supplied
  Invalid property for a QCF: PRI
```

Tipo inválido para o valor da propriedade

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) CCSID(english)
  Unable to create a valid object, please check the parameters supplied
  Invalid value for CCS property: English
```

Conflito da propriedade - cliente/ligações

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) HOSTNAME(polaris.hursley.ibm.com)
  Unable to create a valid object, please check the parameters supplied
  Invalid property in this context: Client-bindings attribute clash
```

Conflito de propriedade – inicialização de saída

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) SECEXITINIT(initStr)
  Unable to create a valid object, please check the parameters supplied
  Invalid property in this context: ExitInit string supplied
  without Exit string
```

Valor da propriedade fora do intervalo válido

```
InitCtx/cn=Trash> DEFINE Q(testQ) PRIORITY(12)
  Unable to create a valid object, please check the parameters supplied
  Invalid value for PRI property: 12
```

Propriedade desconhecida

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) PIZZA(ham and mushroom)
  Unable to create a valid object, please check the parameters supplied
  Unknown property: PIZZA
```

A seguir estão exemplos de condições de erro que podem surgir no Windows ao consultar objetos administrados por JNDI de um aplicativo JMS.

1. Se você estiver usando o provedor JNDI do WebSphere , com.ibm.websphere.naming.WsnInitialContextFactory, deverá usar uma barra (/) para acessar objetos administrados definidos em subcontextos; por exemplo, jms /MyQueueNome. Se você usar uma barra invertida (\), uma InvalidNameException será emitida.
2. Se você estiver usando o provedor JNDI Sun, com.sun.jndi.fscontext.RefFSContextFactory, deverá usar uma barra invertida (\) para acessar objetos administrados definidos em subcontextos; por exemplo, ctx1\\fred. Se você usar uma barra (/), uma NameNotFoundException será emitida.

Usando o WebSphere MQ Explorer para configuração JMS

Use a interface gráfica com o usuário do IBM WebSphere MQ Explorer para criar objetos JMS a partir de objetos WebSphere MQ e objetos WebSphere MQ a partir de objetos JMS, bem como para administrar e monitorar outros objetos WebSphere MQ .

Antes de começar

Antes de criar e configurar objetos administrados JMS com o WebSphere MQ Explorer, inclua um contexto inicial para definir a raiz do namespace JNDI no qual os objetos JMS são armazenados no serviço de nomenclatura e diretório. Para obter mais informações, consulte a assistência ao usuário do IBM WebSphere MQ Explorer para objetos administrados pelo JMS

Sobre esta tarefa

É possível executar as tarefas a seguir com o IBM WebSphere MQ Explorer, contextualmente a partir de um objeto existente no IBM WebSphere MQ Explorer ou a partir de um assistente de criação de novo objeto. Consulte a ajuda do WebSphere MQ Explorer para obter exemplos da assistência do usuário do WebSphere MQ Explorer para algumas tarefas típicas.

Procedimento

- Crie um Connection Factory JMS a partir de qualquer um dos seguintes objetos WebSphere MQ :
  a) Um gerenciador de filas do WebSphere MQ , seja no computador local ou em um sistema remoto.
  b) Um canal do WebSphere MQ
  c) Um listener do WebSphere MQ
- Inclua um gerenciador de filas do WebSphere MQ no WebSphere MQ Explorer usando um Connection Factory JMS
- Criar uma fila JMS a partir de uma fila do WebSphere MQ
- Crie uma fila do WebSphere MQ a partir de uma fila JMS
- Criar um tópico JMS a partir de um tópico do WebSphere MQ , que pode ser um objeto do WebSphere MQ ou um tópico dinâmico
- Criar um tópico do WebSphere MQ a partir de um tópico JMS

Usando o pacote de cabeçalhos do WebSphere MQ

O pacote de Cabeçalhos do WebSphere MQ fornece um conjunto de interfaces e classes auxiliares que podem ser usadas para manipular os cabeçalhos do WebSphere MQ de uma mensagem Geralmente, você usa o pacote de cabeçalhos WebSphere MQ porque deseja executar serviços administrativos usando o servidor de comandos (usando mensagens Programmable Command Format (PCF)).

Sobre esta tarefa

O pacote de cabeçalhos do WebSphere MQ está localizado nos pacotes `com.ibm.mq.headers` e `com.ibm.mq.pcf` . É possível usar esse recurso para as duas APIs alternativas que o WebSphere MQ fornece para uso em aplicativos Java:

- WebSphere MQ classes para Java (também referido como WebSphere MQ Cabeçalhos Base Java).
- WebSphere MQ classes para Java Message Service (WebSphere MQ classes para JMS, também referido como WebSphere MQ JMS).

WebSphere MQ Aplicativos Java base geralmente manipulam objetos MQMessage e as classes de suporte de Cabeçalhos podem interagir diretamente com esses objetos, pois eles entendem nativamente as interfaces Java base do WebSphere MQ .

No WebSphere MQ JMS, a carga útil para uma mensagem geralmente é uma Sequência ou um objeto de matriz de bytes, que pode ser manipulado com fluxos DataInput e DataOutput . O pacote de cabeçalhos WebSphere MQ pode ser usado para interagir com esses fluxos de dados e é adequado para manipular quaisquer mensagens do MQ enviadas e recebidas por aplicativos JMS do WebSphere MQ .

Portanto, embora o pacote de Cabeçalhos do WebSphere MQ contenha referências ao pacote Java Base do WebSphere MQ , ele também deve ser usado em aplicativos JMS do WebSphere MQ e é adequado para uso em ambientes do Java Platform, Enterprise Edition (Java EE)

Uma maneira típica na qual você pode usar o pacote de cabeçalhos WebSphere MQ é manipular mensagens de administração no Programmable Command Format (PCF), por exemplo, por qualquer uma das seguintes razões:

- Para acessar detalhes sobre um recurso do WebSphere MQ .
- Para monitorar a profundidade de uma fila.
- Para inibir o acesso a uma fila.

Usando mensagens PCF com a API JMS do WebSphere MQ , esse tipo de administração de recursos centralizados no aplicativo pode ser executado a partir de aplicativos Java EE sem precisar recorrer ao uso da API Java do WebSphere MQ Base.

Procedimento

- Para usar o pacote de cabeçalhos WebSphere MQ para manipular cabeçalhos de mensagens para classes WebSphere MQ para Java, consulte “Usando com classes do WebSphere MQ para Java” na página 956.
- Para usar o pacote de cabeçalhos WebSphere MQ para manipular cabeçalhos de mensagens para JMS, consulte “Usando com classes WebSphere MQ para JMS” na página 957.

Usando com classes do WebSphere MQ para Java

As classes do WebSphere MQ para aplicativos Java geralmente manipulam objetos MQMessage e as classes de suporte de Cabeçalhos podem interagir diretamente com esses objetos, pois elas entendem nativamente as classes do WebSphere MQ para interfaces Java.

Sobre esta tarefa

WebSphere MQ fornece alguns aplicativos de amostra que demonstram como usar o pacote de cabeçalhos do WebSphere MQ com a API Java base do WebSphere MQ (classes WebSphere MQ para Java).

As amostras mostram duas coisas:

- Como criar uma mensagem PCF para executar uma ação administrativa e analisar a mensagem de resposta.
- Como enviar essa mensagem PCF usando as classes WebSphere MQ para Java.

Dependendo de qual plataforma você estiver usando, essas amostras serão instaladas no diretório `pcf` no diretório `samples` ou `tools` da instalação do WebSphere MQ (consulte “Diretórios de Instalação para Classes WebSphere MQ para Java” na página 664).

Procedimento

1. Crie uma mensagem PCF para executar uma ação administrativa e analise a mensagem de resposta.
2. Envie esta mensagem PCF usando as classes do WebSphere MQ para Java

Conceitos relacionados

“Manipulando cabeçalhos da mensagem do WebSphere MQ com classes do WebSphere MQ para Java” na página 686
Classes Java são fornecidas representando diferentes tipos de cabeçalho da mensagem. Duas classes auxiliares também são fornecidas.

“Manipulando mensagens PCF com classes WebSphere MQ para Java” na página 691
As classes Java são fornecidas para criar e analisar mensagens estruturadas por PCF e para facilitar o envio de solicitações PCF e a coleta de respostas PCF.

Usando com classes WebSphere MQ para JMS

Para usar os cabeçalhos do WebSphere MQ com as classes do WebSphere MQ para JMS, execute as mesmas etapas essenciais que para as classes do WebSphere MQ para Java. A mensagem PCF pode ser criada e a resposta analisada exatamente da mesma maneira usando o pacote de cabeçalhos do WebSphere MQ e o mesmo código de amostra para classes do WebSphere MQ para Java.

Sobre esta tarefa

Para enviar uma mensagem PCF usando a API do WebSphere MQ , a carga útil da mensagem deve ser gravada em uma Mensagem de Bytes JMS e enviada usando as APIs JMS padrão. A única consideração é que a mensagem não deve conter um JMS RFH2 ou quaisquer outros cabeçalhos com valores específicos no MQMD.

Para enviar uma mensagem PCF, conclua as etapas a seguir. A maneira na qual a mensagem PCF é criada e as informações são extraídas da mensagem de resposta é a mesma para as classes WebSphere MQ para Java (consulte “Usando com classes do WebSphere MQ para Java” na página 956).

Procedimento

1. Crie um Destino de Fila JMS que represente o SYSTEM.ADMIN.COMMAND.QUEUE.

   WebSphere MQ Os aplicativos JMS enviam as mensagens PCF para o SYSTEM.ADMIN.COMMAND.QUEUE, e precisa de acesso a um objeto de Destino JMS que representa essa fila O destino deve ter as seguintes propriedades do conjunto:

   ```
   WMQ_MQMD_WRITE_ENABLED = YES
   WMQ_MESSAGE_BODY = MQ
   ```

   Se você estiver usando o WebSphere Application Server, deverá definir essas propriedades como propriedades customizadas no Destino.

   Para criar o destino programaticamente a partir de um aplicativo, use o código a seguir:

   ```
   Queue q1 = session.createQueue("SYSTEM.ADMIN.COMMAND.QUEUE");
   ((MQQueue) q1).setIntProperty(WMQConstants.WMQ_MESSAGE_BODY,
       WMQConstants.WMQ_MESSAGE_BODY_MQ);
   ((MQQueue) q1).setMQMDWriteEnabled(true);
   ```

2. Converta uma mensagem PCF em uma mensagem de Bytes JMS contendo os valores de MQMD corretos

   Uma mensagem de Bytes JMS precisa ser criada e a Mensagem PCF gravada nela. Uma fila de resposta precisa ser criada, mas ela não precisa ter nenhuma configuração específica.

   O fragmento de código de amostra a seguir mostra como criar uma Mensagem de Bytes JMS e gravar um objeto com.ibm.mq.headers,pcf.PCFMessage nele.. O objeto PCFMessage (pcfCmd) foi construído anteriormente usando o pacote de cabeçalhos WebSphere MQ . (Observe que o pacote para carregar o PCFMessage é com.ibm.mq.headers.pcf.PCFMessage).

   ```
    // create the JMS Bytes Message
   final BytesMessage msg = session.createBytesMessage();

   // Create the wrapping streams to put the bytes into the message payload
   ByteArrayOutputStream baos = new ByteArrayOutputStream();
   DataOutput dataOutput = new DataOutputStream(baos);

   // Set the JMSReplyTo so the answer comes back
   msg.setJMSReplyTo(new MQQueue("adminResp"));

   // write the pcf into the stream
   pcfCmd.write(dataOutput);
   baos.flush();
   msg.writeBytes(baos.toByteArray());

   // we have taken control of the MD, so need to set all
   // flags in the MD that we require - main one is the format
   msg.setJMSPriority(4);
   msg.setIntProperty(WMQConstants.JMS_IBM_MQMD_PERSISTENCE,
   ```

```
  CMQC.MQPER_NOT_PERSISTENT);
msg.setIntProperty(WMQConstants.JMS_IBM_MQMD_EXPIRY, 300);
msg.setIntProperty(WMQConstants.JMS_IBM_MQMD_REPORT,
    CMQC.MQRO_PASS_CORREL_ID);
msg.setStringProperty(WMQConstants.JMS_IBM_MQMD_FORMAT, "MQADMIN");

// and send the message
sender.send(msg);
```

3. Envie a mensagem e receba a resposta usando as APIs do JMS padrão
4. Converta a mensagem de resposta em uma mensagem PCF para processamento.

   Para recuperar a mensagem de resposta e processá-la como uma mensagem PCF, use o código a seguir:

```
 // Get the message back
BytesMessage msg = (BytesMessage) consumer.receive();

// get the size of the bytes message & read into an array
int bodySize = (int) msg.getBodyLength();
byte[] data = new byte[bodySize];
msg.readBytes(data);

// Read into Stream and DataInput Stream
ByteArrayInputStream bais = new ByteArrayInputStream(data);
DataInput dataInput = new DataInputStream(bais);

// Pass to PCF Message to process
PCFMessage response = new PCFMessage(dataInput);
```

Conceitos relacionados

“Mensagens JMS” na página 818
As mensagens JMS são compostas de um cabeçalho, propriedades e um corpo. O JMS define cinco tipos de corpo da mensagem

Usando serviços da Web no WebSphere MQ

É possível desenvolver aplicativos IBM WebSphere MQ para serviços da web usando o transporte IBM WebSphere MQ para SOAP ou a ponte IBM WebSphere MQ para HTTP (Protocolo de Transporte de Hipertexto)

O transporte IBM WebSphere MQ para SOAP fornece um transporte JMS para SOAP.. O transporte IBM WebSphere MQ para SOAP também é integrado a outros ambientes, como Microsoft Windows Communication Foundation, WebSphere Application Server e CICS Transaction Server.

Para obter mais informações sobre o transporte do IBM WebSphere MQ para SOAP, consulte “Transporte do WebSphere MQ para SOAP” na página 959

Com a ponte IBM WebSphere MQ para HTTP, os aplicativos clientes podem trocar mensagens com IBM WebSphere MQ sem a necessidade de instalar um cliente MQI do WebSphere MQ . É possível chamar o WebSphere MQ de qualquer plataforma ou idioma com recursos HTTP.

Para obter mais informações sobre a ponte IBM WebSphere MQ para HTTP, consulte “WebSphere MQ ponte para HTTP” na página 1035.

Conceitos relacionados

“Conceitos de desenvolvimento de aplicativos” na página 8
É possível usar uma opção de linguagens orientadas a objeto ou processuais para gravar aplicativos IBM WebSphere MQ. Use os links neste tópico para obter informações sobre conceitos do IBM WebSphere MQ que são úteis para desenvolvedores de aplicativos.

“Decidindo qual linguagem de programação usar” na página 80
Use estas informações para descobrir sobre linguagens de programação e estruturas suportadas pelo IBM WebSphere MQe algumas considerações para usá-las.

“Projetando aplicativos IBM WebSphere MQ” na página 91
Quando você tiver decidido como seus aplicativos podem aproveitar as plataformas e ambientes disponíveis para você, será necessário decidir como usar os recursos oferecidos pelo WebSphere MQ.

“Programas de amostra do WebSphere MQ” na página 98
Use esta coleção de tópicos para aprender sobre programas de amostra do WebSphere MQ em diferentes plataformas

“Gravando um Aplicativo de Enfileiramento” na página 197
Use essas informações para aprender como escrever aplicativos de enfileiramento, conectar e desconectar de um gerenciador de filas, publicar/assinar e abrir e fechar objetos.

“Gravando aplicativos clientes” na página 356
O que você precisa saber para gravar os aplicativos clientes no WebSphere MQ

“Escrevendo aplicativos de publicar/assinar” na página 281
Inicie a gravação de aplicativos de publicação / assinatura do WebSphere MQ

“Construindo um aplicativo IBM WebSphere MQ” na página 435
Use estas informações para saber como construir um aplicativo IBM WebSphere MQ em diferentes plataformas.

“Manipulando erros do programa” na página 555
Estas informações explicarão os erros associados às chamadas MQI de aplicativos, ao realizar uma chamada ou quando sua mensagem for entregue para seu destino final.

Transporte do WebSphere MQ para SOAP

O transporte WebSphere MQ para SOAP fornece um transporte JMS para SOAP. O transporte do WebSphere MQ para SOAP também é integrado a outros ambientes, como Microsoft Windows Communication Foundation, WebSphere Application Server e CICS Transaction Server.

Introdução ao IBM WebSphere MQ Transport for SOAP

O transporte IBM WebSphere MQ para SOAP fornece um transporte JMS para SOAP.. O emissor e o listener SOAP do WebSphere MQ fornecem um meio de chamar serviços da Web

O listener SOAP WebSphere MQ suporta serviços hospedados pelo .NET Framework 1, .NET Framework 2 e Axis 1.4. O emissor SOAP WebSphere MQ suporta clientes de serviços da Web em execução no .NET Framework 1, .NET Framework 2, Axis 1.4 e Axis2. Os clientes podem ser um servidor WebSphere MQ ou aplicativo cliente. O transporte IBM WebSphere MQ para SOAP também é integrado a outros ambientes, como Microsoft Windows Communication Foundation, WebSphere Application Server e CICS Transaction Server.

A integração no Microsoft Windows Communication Foundation faz parte do suporte do IBM WebSphere MQ para .NET Framework 3.

O transporte IBM WebSphere MQ para SOAP é um conjunto de protocolos e ferramentas para transportar mensagens SOAP usando JMS sobre IBM WebSphere MQ.. Ele é empacotado de maneiras diferentes para ambientes de aplicativos diferentes, conforme mostrado na Tabela 137 na página 959.

Tabela 137. Ambientes de aplicativos do IBM WebSphere MQ Transport for SOAP

	Integrado com componentes adicionais do WebSphere MQ	**Integrado a uma estrutura**
Fornecido como parte da instalação do WebSphere MQ ..	.NET Framework 1 . Axis 1.4 do NET Framework 2	Windows Communication Foundation (.NET Framework 3) Axis2 (Somente Cliente)
Fornecido em outro pacote de software		WebSphere Application Server CICS Transaction Server 4.1 WebSphere ESB WebSphere Process Server for Multiplatforms

A integração do IBM WebSphere MQ Transport for SOAP em uma estrutura de aplicativo simplifica o desenvolvimento e a implementação de serviços da web para IBM WebSphere MQ.

Com componentes SOAP do IBM WebSphere MQ adicionais, é possível interagir diretamente com os componentes SOAP do WebSphere MQ para desenvolver e implementar serviços Use as ferramentas do IBM WebSphere MQ SOAP para configurar e implementar os serviços da web e os clientes de serviço da web para IBM WebSphere MQ.

Nos ambientes integrados, o desenvolvimento e a implementação é mais simples. É possível usar as mesmas ferramentas para desenvolvimento e implementação, como você faria para desenvolver e implementar um serviço da web SOAP HTTP. Você ainda deve configurar as filas, canais e gerenciadores de filas do IBM WebSphere MQ que você requer usando as ferramentas do WebSphere MQ .

É possível combinar e corresponder com clientes e servidores do IBM WebSphere MQ SOAP a partir de qualquer um desses ambientes.

Benefícios

O transporte do WebSphere MQ para SOAP oferece aos usuários existentes do IBM WebSphere MQ os seguintes benefícios principais:

Usando a rede do IBM WebSphere MQ para se conectar aos serviços da web existentes.

Os serviços podem ser aqueles gravados ou serviços fornecidos como interfaces para outros aplicativos de software empacotados implementados.

O benefício vem do uso da rede existente do WebSphere MQ para conectar serviços da web. O transporte do IBM WebSphere MQ tem a vantagem de ser um serviço de sistema de mensagens na fila confiável e gerenciado.

Gravando novos aplicativos ou convertendo seus aplicativos existentes, para usar SOAP em vez de interfaces do IBM WebSphere MQ.

Geralmente, os aplicativos requerem que um adaptador específico do WebSphere MQ seja desenvolvido para integrar com outro aplicativo Os adaptadores possuem duas partes: a parte do conector, que coloca e obtém mensagens para e a partir do transporte e a parte do adaptador que converte dados para e a partir de formatos específicos do aplicativo. A integração de cada par de aplicativos é um novo desafio.

O benefício de SOAP vem da padronização em SOAP para definir as interfaces de aplicativo e, em seguida, ter uma opção de transportes. Não é necessário gravar adaptadores específicos do aplicativo e é possível escolher se deseja usar o IBM WebSphere MQ ou HTTP como o conector. Qual transporte você escolhe, depende de quais qualidades de serviço e conectividade você requer.

Para usuários SOAP sobre HTTP existentes, o benefício do transporte WebSphere MQ para SOAP vem do uso de um transporte assíncrono gerenciado e confiável. Os benefícios são duplos:

Um modelo de programação realmente assíncrono para disponibilidade e desempenho.
Ao usar uma interface de cliente assíncrona, os aplicativos de serviço e cliente não precisarão estar disponíveis ao mesmo tempo. As solicitações enviadas pelo cliente serão armazenadas até que o serviço esteja disponível para processá-las.

Uma rede gerenciada pronta construída projetada para ser confiável e disponível.

Ao escolher o IBM WebSphere MQ como um transporte, você estará obtendo a vantagem de usar uma rede gerenciada que fornece o sistema de mensagens confiável.

Em contraste, transportes, como HTTP e FTP sobre TCP/IP não são gerenciados. Uma rede não gerenciada é ideal para conexões imprevisíveis: há menos tarefas de gerenciamento.

Resumo

O IBM WebSphere MQ Transport for SOAP fornece os componentes a seguir:

- A ligação de transporte SOAP/JMS é usada em documentos WSDL para ligar um serviço SOAP a um transporte JMS. A implementação do WebSphere MQ da ligação SOAP/JMS usa um URI que assume uma das duas formas:

Transporte do WebSphere MQ para SOAP

```
jms:/queue?&Name=Value&Name=Value...
```

Formato de ligação do WebSphere MQ para recomendação de candidato do W3C

```
jms:queue:qName?connectionFactory=connectQueueManager(qMgrName)&Name=Value&Name=Value...
```

- O mapeamento de uma mensagem SOAP para uma mensagem WebSphere MQ .
- Dois listeners SOAP do IBM WebSphere MQ para receber solicitações SOAP, um para Java e um para .NET Framework 1 ou .NET Framework 2. Os listeners usam .NET ou Axis 1.4 para processar a solicitação SOAP.
- Dois emissores do IBM WebSphere MQ SOAP criam solicitações SOAP do IBM WebSphere MQ. os clientes de serviços da web se registram com um emissor para processar as solicitações SOAP `jms:`.
- Integração com Windows Communication Foundation (WCF), às vezes conhecido como .NET 3, para enviar e receber mensagens do WebSphere MQ Transport for SOAP.
- Integração do cliente com Axis2, às vezes conhecido como JAX-WS, para enviar mensagens JMS do WebSphere MQ Transport for SOAP ou W3C SOAP.
- O comando **amqwdeployWMQService**, que cria o desenvolvimento, os componentes de tempo de execução e os scripts para implementar um serviço da web usando o IBM WebSphere MQ Transport for SOAP.
- Código de cliente e de serviço Java e .NET de amostra
- Um script para configurar o caminho de classe e outros scripts de utilitário.

Nos ambientes integrados, o emissor e o listener são integrados em cada ambiente, pois são extensões para as ferramentas de desenvolvimento e implementação.

Integração de SOAP e WebSphere MQ

O transporte do WebSphere MQ para SOAP estende SOAP e ferramentas de serviços da web e tempo de execução, com WebSphere MQ como um transporte alternativo para HTTP para SOAP. Não é necessário modificar os serviços da Web existentes para usar o transporte WebSphere MQ para SOAP como um transporte. O transporte usa um formato URI customizado para SOAP/JMS. O formato URI W3C para SOAP/JMS é suportado de uma maneira limitada pelos clientes Axis2 .

Uma linha de código adicional deve ser incluída nos clientes nos ambientes do .NET Framework 1, .NET Framework 2 e Axis 1.4 . Nenhum código adicional é necessário nos clientes Axis 2 e Windows Communication Foundation (WCF). O listener SOAP WebSphere MQ executa serviços nos ambientes .NET Framework 1, .NET Framework 2 e Axis 1.4 . O transporte do WebSphere MQ para SOAP é integrado em alguns outros ambientes de servidor de aplicativos, incluindo WCF, CICS e WebSphere Application Server

O que é SOAP?

SOAP[9]descreve o formato padronizado das mensagens e protocolos de interação que os aplicativos usam para trocar solicitações, respostas e datagramas. SOAP é independente do transporte usado para transferir as mensagens e do ambiente de aplicativos que envia e recebe as mensagens. O W3C define o SOAP Versão 1.2 sucintamente:

O SOAP Versão 1.2 fornece a definição das informações baseadas em XML que podem ser usadas para trocar informações estruturadas e digitadas entre peers em um ambiente distribuído descentralizado.[10].

Para usar SOAP ele deve ser ligado a um transporte, como HTTP, e-mail ou WebSphere MQ.

Uma estrutura de ligação de protocolo SOAP é o conjunto de regras para transportar uma mensagem SOAP na parte superior de outro protocolo, como HTTP. SOAP Versão 1.2 Parte 2: Auxiliares (Segunda edição) descreve a ligação HTTP SOAP.

[9] Historicamente, o acrônimo era Simple Object Access Protocol.
[10] W3C: SOAP Versão 1.2 Parte 0

A recomendação do candidato W3C , de 4 de junho de 2009, SOAP sobre Java Message Service 1.0, descreve a recomendação para a ligação JMS SOAP Como JMS é uma especificação de API, e não um protocolo de transporte, a recomendação SOAP JMS não descreve o formato de ligação de mensagens JMS SOAP Ele descreve os protocolos de interação SOAP e a ligação da API JMS Consequentemente, ao usar a recomendação SOAP JMS, você ainda deve utilizar a mesma implementação JMS para o cliente SOAP e o servidor SOAP Ele permite que um aplicativo JMS SOAP seja executado em qualquer implementação do JMS Uma implementação JMS pode ser conectada a um servidor de aplicativos J2EE , se tanto o servidor quanto a implementação JMS estiverem em conformidade com a especificação JCA WebSphere MQ JMS está em conformidade com a especificação JCA e pode ser conectado a um servidor de aplicativos compatível.

O transporte do WebSphere MQ para ligação SOAP é como o padrão W3C proposto, mas não é o mesmo. Seu uso é descrito no tópico Configurações SOAP de MQRFH2. Ao contrário da recomendação do candidato do W3C, a ligação SOAP não está formalmente especificada. Com efeito, é a ligação HTTP e o endereço de serviço toma o formulário, `jms:/queue?name=value&name=value...`, em vez `http://authority/path?query#fragment`. `jms:` não é um esquema de URI oficialmente registrado pelo IANA.

O que é um serviço da web?

SOAP permite que os programas gravados em linguagens diferentes, em execução em diferentes plataformas, se comuniquem usando vários protocolos de transporte. SOAP é a especificação do protocolo. Um serviço da web é um aplicativo que fornece um serviço através de uma interface SOAP que pode ser acessado usando protocolos da internet.

Um objetivo importante do SOAP é fornecer serviços que os clientes possam usar facilmente. Depois de ter projetado um cliente para usar um serviço, será possível programar a chamada para solicitar o serviço sem fazer referência à documentação externa. As interfaces de serviço são descritas em XML, em um documento WSDL. A consulta, `http://authority/path?wsdl`, retorna a descrição WSDL de um serviço SOAP.

Sugestão: Ao implementar um serviço da Web para usar o WebSphere MQ, também implemente o serviço para HTTP para que a consulta WSDL padrão funcione..

Desenvolvendo serviços da web

Os serviços da web têm um cliente e uma parte do serviço. O serviço é gravado primeiro, iniciando a partir da descrição de interface em WSDL ou seguindo as regras para gravar a classe de serviço. Os kits de ferramentas de serviço da web têm utilitários para gerar WSDL a partir de uma definição de interface de uma classe; por exemplo, **`java2wsdl`** ou **`disco`** Eles também possuem ferramentas para gerar estruturas básicas de classe a partir de descrições de interface WSDL; por exemplo **`wsdl2java`**, **`wsimport`** ou **`wsdl`**. O primeiro é conhecido como desenvolvimento ascendente e o segundo como desenvolvimento descendente.

O comando **`amqwdeployWMQService`** no transporte do WebSphere MQ para SOAP usa essas ferramentas para gerar WSDL, stubs de cliente e proxies de cliente

Os serviços da web são geralmente gravados usando um ambiente de desenvolvimento integrado destinado a um ambiente de servidor de aplicativos específico:

Eclipse IDE para Desenvolvedores Java EE
Cria serviços da Web para o Axis 2 Suporta JAX-RPC e JAX-WS

Rational Application Developer V7.5
Cria serviços da web para o WebSphere Application Server V7 e versões anteriores e também para Axis. Suporta JAX-RPC e JAX-WS.

WebSphere Integration Developer V6.2
Cria serviços da web para WebSphere Process Server e WebSphere ESB Suporta JAX-RPC e JAX-WS.

Visual Studio 2008 (Versão 9)
Cria serviços da Web para .NET Framework 3.5 e anterior (Windows Communication Foundation)

Visual Studio 2005 (Versão 8)
Cria serviços da web para .NET Framework 2 e anterior

É possível usar qualquer uma dessas ferramentas em combinação com o transporte WebSphere MQ para SOAP. Após ter desenvolvido um serviço para usar com HTTP, use a ferramenta **amqwdeployWMQService** para implementar os serviços para usar o WebSphere MQ como um transporte. É possível gravar um novo cliente usando a saída da ferramenta ou modificar seus clientes existentes para usar o transporte WebSphere MQ para SOAP.

Se o transporte do WebSphere MQ para SOAP estiver integrado no ambiente de aplicativos, então não será necessário usar a ferramenta **amqwdeployWMQService** ou modificar o código do cliente A camada SOAP do cliente direciona solicitações do cliente que possuem um URI com o prefixo `jms:` para o transporte WebSphere MQ para SOAP. A camada SOAP do servidor chama o transporte WebSphere MQ para SOAP aguardar `jms:` solicitações SOAP e retorna respostas para o transporte WebSphere MQ para SOAP.

Geralmente, os serviços .NET foram desenvolvidos de baixo para cima usando anotações de serviço da Web no código e os serviços Java de cima para baixo, usando definições de interface WSDL A diferença em abordagens é limitada, pois o Java Standard Edition Versão 6 suporta JAX-WS 2.0e usa anotações para qualificar a definição de interfaces de serviço Agora é tão fácil desenvolver os serviços Java de baixo para cima quanto de cima para baixo. Qual abordagem escolhida é uma questão de método de desenvolvimento.

O cliente de serviço da web estará gravado após o serviço, usando a definição de serviço WSDL e stubs do cliente gerado e proxies. Em alguns aplicativos, a definição de serviço não será conhecida quando o cliente for gravado. O cliente recupera o WSDL de serviço e cria solicitações de serviço dinamicamente. Mais comumente a definição de serviço é conhecida, mas o endereço no qual o serviço é implementado, não é. O kit de ferramentas de serviços da web gera interfaces para o cliente a ser usado para fazer solicitações de serviço. O cliente fornecerá o endereço de serviço quando for necessário. No terceiro caso, o WSDL contém todas as informações que um cliente precisa. O WSDL contém a interface e o endereço do serviço. O código gerado pelo kit de ferramentas de serviço da web tem todas as informações necessárias pelo cliente para fazer solicitações de um serviço.

É possível usar qualquer um destes três estilos com o transporte WebSphere MQ para SOAP.

Ambientes de aplicativo de serviço da web

Os kits de ferramentas de serviço da web requerem um mapeamento a partir da definição WSDL de um serviço aos fluxos de bytes transferidos em solicitações e respostas SOAP. O fluxo de bytes é definido pela especificação SOAP e está contido no envelope SOAP. O envelope SOAP é mostrado em Figura 166 na página 963.

```
<?xml version='1.0'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header> <!-- optional -->
<!-- headers... -->
</soap:Header>
<soap:Body>
<!-- payload or fault message -->
</soap:Body>
</soap:Envelope>
```

Figura 166. Envelope SOAP

O mapeamento a partir do envelope SOAP para a ligação entre linguagens e novamente é a peça padronizada e o proprietário da peça. O mapeamento é fundamental para a arquitetura .NET, e é fornecido como parte do Common Language Runtime (CLR). O mapeamento é padronizado em Java por especificações JAX.. Como os mapeamentos Java são padronizados, os clientes e serviços de serviço da web Java são móveis entre diferentes ambientes de aplicativos baseados em Java. JAX-RPC (às vezes chamado de JAX-WS 1.0) é o mapeamento mais usado nos dias de hoje. Ele é suportado por Axis 1.4. JAX-WS (às vezes chamado de JAX-WS 2.0) é um padrão muito melhorado e provavelmente substituirá o JAX-RPC rapidamente. O JAX-WS é suportado por Axis 2.0. WebSphere MQ 7.0.1 não suporta JAX-WS e Axis 2.

O transporte WebSphere MQ para SOAP não altera o conteúdo do envelope SOAP e o conteúdo não afeta o transporte. As ligações de linguagem afetam o transporte do WebSphere MQ para SOAP. WebSphere MQ 7.0.1 suporta .NET Framework 1, .NET Framework 2 e Axis 1.4 usando o código e os utilitários enviados com o transporte WebSphere MQ para SOAP. O suporte para o transporte WebSphere para SOAP no .NET Framework 3 e 3.5 é implementado usando o canal customizado do WebSphere MQ para o Windows Communication Foundation.

Outros ambientes de desenvolvimento e tempo de execução SOAP podem enviar suporte para o transporte WebSphere MQ para SOAP e suportar diferentes idiomas. Por exemplo, serviços da web em execução no CICS suportam linguagens como COBOL e PL/1.

Nota: O mapeamento usado não faz diferença para a interoperabilidade de serviços da web. É possível combinar e corresponder clientes e serviços gravados usando mapeamentos .NET, JAX-RPC e JAX-WS..

O que é o transporte do WebSphere MQ para SOAP

WebSphere MQ transporte para SOAP é uma ligação SOAP e um kit de ferramentas de serviços da web. Juntos, eles permitem que aplicativos troquem mensagens SOAP usando WebSphere MQ em vez de HTTP. Figura 167 na página 964 mostra o WebSphere MQ como uma alternativa para HTTP como um transporte SOAP

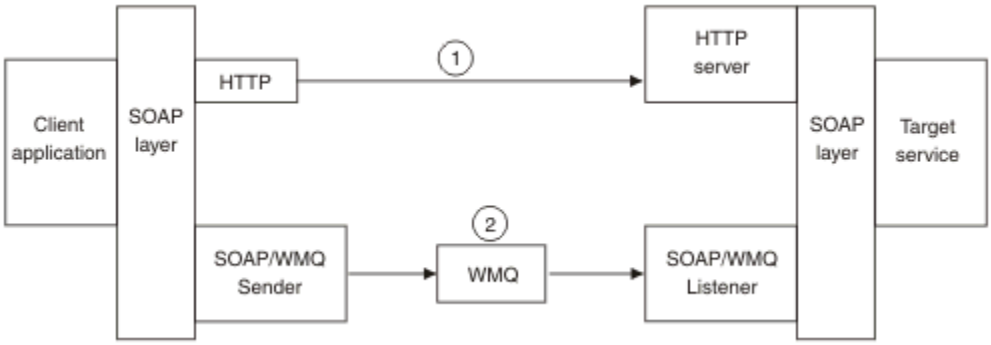


Figura 167. Visão Geral do Transporte WebSphere MQ para SOAP

SOAP sobre HTTP é mostrado como (1) no diagrama. A camada SOAP do cliente converte uma solicitação em uma mensagem SOAP e o componente HTTP envia sobre TCP/IP. O componente do servidor HTTP atende as solicitações de HTTP, geralmente na porta 80 do TCP/IP. Se a solicitação for para um serviço SOAP, o componente do servidor HTTP chamará a camada SOAP para converter a solicitação SOAP em chamada de método. Ele então retorna a resposta.

SOAP sobre WebSphere MQ é mostrado como (2). O aplicativo cliente registra o componente emissor SOAP WebSphere MQ como um manipulador para o protocolo `jms:` com a camada SOAP. A camada SOAP passa mensagens SOAP endereçadas a `jms:` para o WebSphere MQ emissor SOAP. O emissor usa o URI na mensagem para colocar a mensagem na fila de solicitações com as qualidades de serviço necessárias. O listener SOAP WebSphere MQ correspondente aguarda mensagens em sua fila de solicitações e chama a camada SOAP para processar solicitações e retornar respostas.

O emissor e o listener SOAP são programas normais do WebSphere MQ Eles podem ser conectados no mesmo gerenciador de filas, como em Figura 168 na página 965 ou conectados a gerenciadores de filas diferentes; consulte Figura 169 na página 966. O cliente pode ser conectado por uma conexão do cliente.

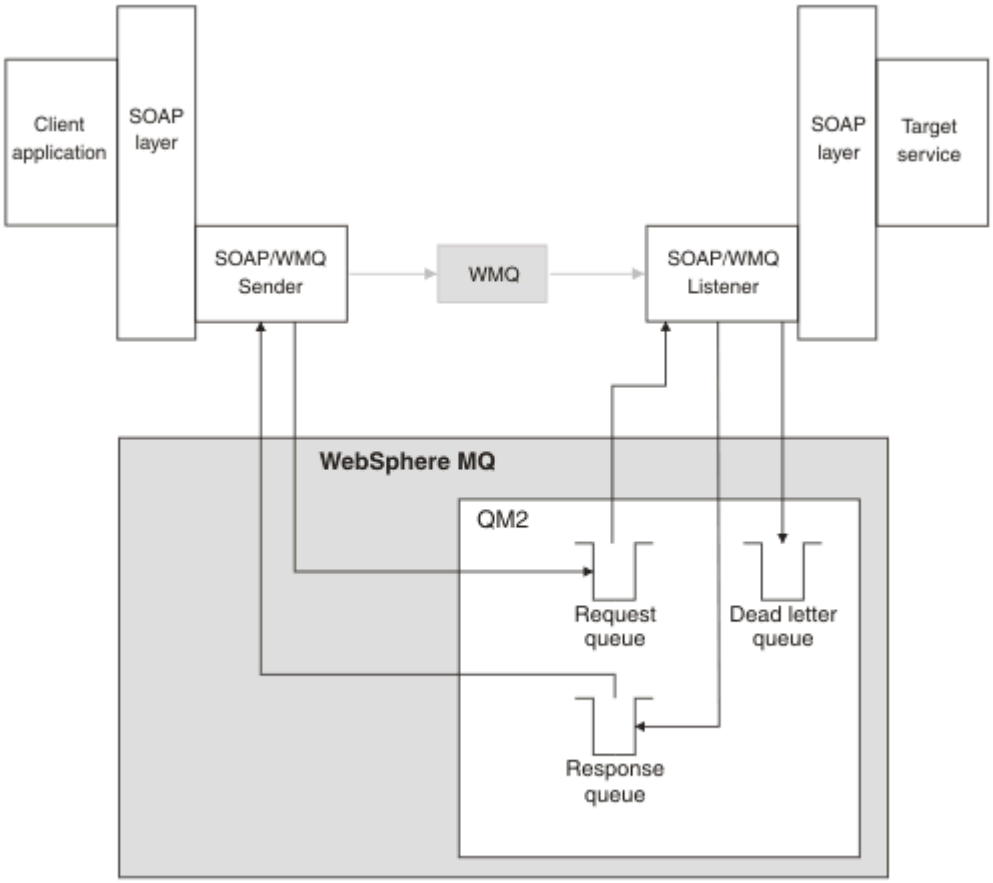


Figura 168. Filas usadas pelo SOAP/WebSphere MQ (gerenciador de filas único)

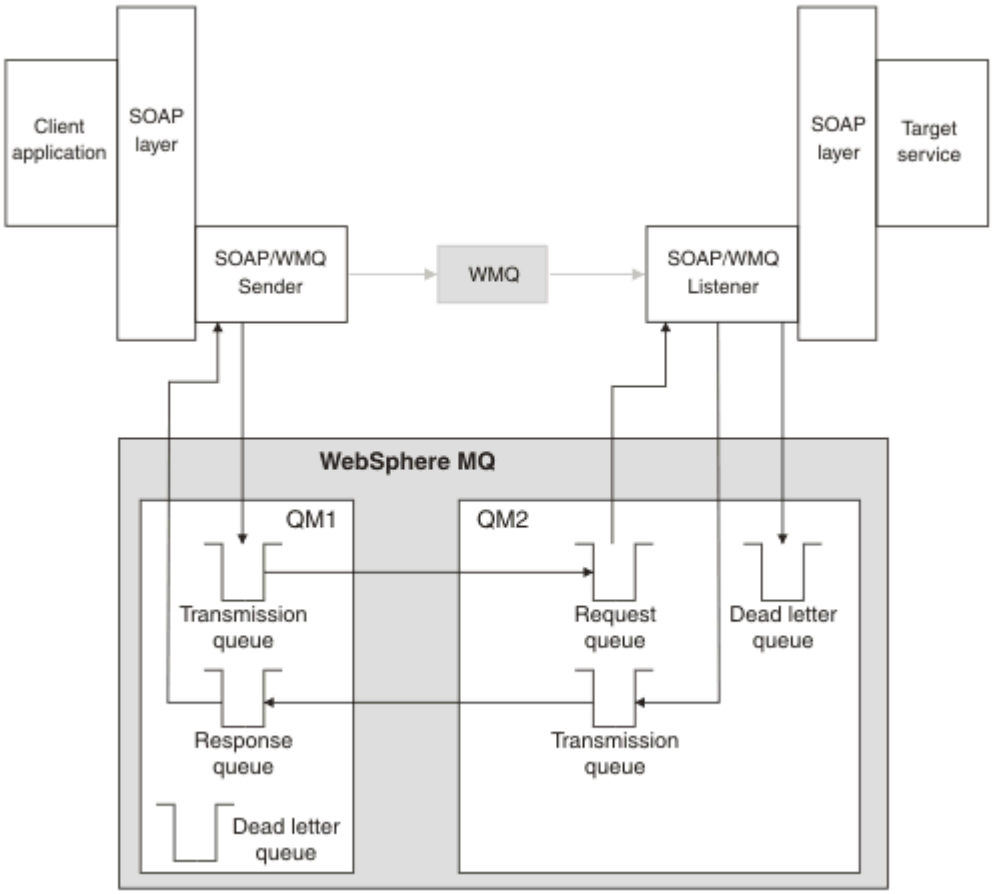


Figura 169. Filas usadas pelo SOAP/WebSphere MQ (gerenciadores de filas separados).

W3C recomendação de candidato para ligar SOAP a JMS

A recomendação de candidato do W3C define a ligação SOAP sobre JMS; SOAP sobre Java Message Service 1.0. O Esquema de URI para Java(tm) Message Service 1.0 também é útil para seus exemplos[11].

Algumas estruturas de aplicativos, como o WebSphere Application Server v7, têm suporte para a recomendação de candidato W3C Envie solicitações SOAP formatadas com um URI compatível com a recomendação de candidato W3C usando o cliente Axis2 ; consulte W3C URI SOAP sobre JMS para o cliente WebSphere MQ Axis 2 . O cliente Axis2 envia um pedido SOAP formatado com um transporte W3C ou WebSphere MQ para SOAP com base no URI no pedido SOAP.

O suporte a clientes do Axis2 para a recomendação W3C é introduzido no fix pack 7.0.1.3. Suporte para outros clientes e para os listeners SOAP fornecidos pelo WebSphere MQ não é fornecido.

Conceitos relacionados

Implementação do transporte do WebSphere para SOAP no .NET Framework 1, .NET 2 e Axis 1.4
Talvez você queira gravar seu próprio emissor e listener SOAP do WebSphere MQ . Use a implementação do transporte do WebSphere MQ para SOAP no .NET Framework 1, .NET Framework 2 e Axis 1.4 como um guia.

Transporte do WebSphere MQ para SOAP e sistema de mensagens confiável de serviços da web
O sistema de mensagens confiável dos serviços da web é um protocolo para trocar confiável de solicitações do serviço da web e respostas através de uma conexão não confiável. É adequado para resolver problemas de interrupção de conexão de curta duração.

[11] Procure *Esquema de URI para JMS*, nas referências de especificação W3C, para o rascunho mais recente.

Implementação do transporte do WebSphere para SOAP no .NET Framework 1, .NET 2 e Axis 1.4

Talvez você queira gravar seu próprio emissor e listener SOAP do WebSphere MQ . Use a implementação do transporte do WebSphere MQ para SOAP no .NET Framework 1, .NET Framework 2 e Axis 1.4 como um guia.

1. Um programa cliente usa a estrutura de serviços da web apropriada da mesma maneira que faria para o transporte HTTP. Ele também deve registrar o prefixo `jms:`. O prefixo é registrado usando o método Java `com.ibm.mq.soap.Register.extension()` ou o método CLR `IBM.WMQSOAP.Register.Extension()` .
2. A estrutura Axis 1.4 ou .NET Framework 1 ou 2 serializar a chamada em uma mensagem de solicitação SOAP exatamente como para SOAP/HTTP.
3. Um serviço do WebSphere MQ é identificado por um URI prefixado com `jms:` Quando a estrutura identifica o URI `jms:` , ele chama o código do emissor de transporte do WebSphere MQ ; `com.ibm.mq.soap.transport.jms.WMQSender` (para o Eixo 1.4) ou `IBM.WMQSOAP.MQWebRequest` (para .NET1 e 2). Se a estrutura encontrar um URI com um prefixo `http:`, ela chamará o emissor padrão de SOAP sobre HTTP.
4. A mensagem SOAP é transportado pelo emissor SOAP do WebSphere MQ usando a fila de solicitações O **`SimpleJavaListener`** (para Java) ou o **`amqwSOAPNETListener`** (para .NET) recebe a mensagem de solicitação

   Os listeners SOAP WebSphere MQ são processos independentes e multiencadeados com um número customizável de encadeamentos.
5. O listener SOAP do WebSphere MQ lê a solicitação SOAP recebida e a transmite para a infraestrutura de serviço da web apropriada
6. A infraestrutura de serviço da web analisa a mensagem de solicitação SOAP e invoca o serviço. O procedimento é o mesmo para uma mensagem que chegou em um transporte de HTTP.
7. A infraestrutura formata a resposta em uma mensagem de resposta SOAP e a retorna para o listener SOAP WebSphere MQ .
8. O listener coloca a mensagem na fila de resposta e a mensagem é transferida para o emissor SOAP WebSphere MQ . O emissor a transmite para a infraestrutura de serviço da web do cliente.
9. A infraestrutura do cliente analisa a mensagem SOAP de resposta e manipula o resultado de volta para o aplicativo cliente.

Cada contexto de aplicativos é atendido por uma fila de pedidos separada do WebSphere MQ

O contexto de aplicativos é controlado no Axis 1.4 assegurando que o listener SOAP e o serviço WebSphere MQ sejam executados no diretório apropriado. Axis 1.4 configura o CLASSPATH correto para o diretório.

O contexto de aplicativos é controlado em .NET pelo listener SOAP do WebSphere MQ executando o serviço em um contexto criado por uma chamada para `ApplicationHost.CreateApplicationHost`. A chamada especifica o diretório de execução de destino. Cada serviço, então, opera no diretório no qual ele foi implementado.

`amqwdeployWMQService` gera as filas de solicitações e respostas. Ele também gera a infraestrutura necessária para manipular as filas e implementar serviços para o Axis 1.4.

Conceitos relacionados

Integração de SOAP e WebSphere MQ

Transporte do WebSphere MQ para SOAP e sistema de mensagens confiável de serviços da web
O sistema de mensagens confiável dos serviços da web é um protocolo para trocar confiável de solicitações do serviço da web e respostas através de uma conexão não confiável. É adequado para resolver problemas de interrupção de conexão de curta duração.

Transporte do WebSphere MQ para SOAP e sistema de mensagens confiável de serviços da web

O sistema de mensagens confiável dos serviços da web é um protocolo para trocar confiável de solicitações do serviço da web e respostas através de uma conexão não confiável. É adequado para resolver problemas de interrupção de conexão de curta duração.

WebSphere MQ para SOAP tira vantagem de usar uma rede gerenciada e confiável do WebSphere MQ para transmitir mensagens SOAP. Transportes como HTTP e FTP não são gerenciados. As redes não gerenciadas são ideais para conexões imprevisíveis, em que as dificuldades e os custos de gerenciamento de conexões excedem os benefícios de não perder solicitações e respostas.

Para superar o problema de perder arquivos quando as conexões são quebradas em redes não gerenciadas, serviços como o FTP gerenciado constroem uma camada de gerenciamento sobre o FTP. A camada de gerenciamento assume a responsabilidade de verificar se os arquivos foram transferidos com sucesso a partir de usuários e retransmite arquivos ausentes, se necessário. Para usar o FTP gerenciado, deve-se ter o software de gerenciamento instalado em ambas as extremidades da conexão.

O sistema de mensagens confiável de serviços da web (WSRM) assume uma abordagem diferente para solucionar o problema de conexões não confiáveis. Seu objetivo é transferir as solicitações e respostas confiáveis do serviço da web, sem ambas as extremidades da conexão terem que usar o mesmo software. Qualquer software, implementando o protocolo do sistema de mensagens confiável de serviços da web, pode trocar mensagens confiavelmente com outro software.

Quando uma conexão falhar, um emissor e um receptor deverão preservar o contexto da transferência de mensagem WSRM, usando o URI gerado como uma chave. O emissor e receptor tentam estabelecer uma nova conexão. Se uma nova conexão for estabelecida com sucesso, a transferência será concluída. A especificação WSRM não especifica como o contexto é preservado ou quando uma nova conexão é tentada.

Você pode decidir que apenas indisponibilidades de curta duração são de interesse. Para mais indisponibilidades, você pode estar preparado para descartar as transferências que não puderam ser reiniciadas após uma hora. Da mesma forma, você pode estar preparado para descartar as transferências, se o cliente ou serviço falhar. Deixar o usuário responsável por garantir as transferências, coloca menos demandas no gerenciamento de coordenação de cliente e serviço.

Se as indisponibilidades de rede forem de longa duração, mais de 30 minutos mais ou menos, ou se o cliente ou o servidor falhar, haverá uma probabilidade maior de que algumas conexões nunca serão restabelecidas. Não é mais possível contar com o WSRM restaurando a transferência de mensagem automaticamente em um modo não gerenciado. É necessário considerar o gerenciando das conexões WSRM com falha, que significa o desenvolvimento de software para gerenciar a rede de clientes e serviços.

O uso do WSRM para superar indisponibilidades curtas pode reduzir significativamente como lidar com mensagens perdidas em uma rede remota. Se você não tiver para garantir a entrega de mensagens, os benefícios de reduzir a perda de mensagens poderá justificar o custo adicional de desenvolvimento de uma implementação do WSRM.

SOAP sobre JMS fornece entrega de mensagem assegurada e lida com interrupções de longa duração do cliente, do servidor e da rede. Se você estiver buscando uma qualidade de serviço mais confiável para SOAP do que HTTP, qual solução você escolher: WebSphere MQ transporte para SOAP ou WSRM? A resposta depende de vários fatores. Alguns dos fatores a serem considerados são listados:

1. Se as incertezas for devido a uma falha de conexão.
2. Quanto tempo as falhas de conexão são desejadas.
3. Se for possível gerenciar ambos, o cliente e o lado do servidor da conexão.
4. Se o usuário ou um administrador for responsável basicamente para a entrega de mensagens.

Conceitos relacionados

Integração de SOAP e WebSphere MQ

Implementação do transporte do WebSphere para SOAP no .NET Framework 1, .NET 2 e Axis 1.4

Talvez você queira gravar seu próprio emissor e listener SOAP do WebSphere MQ . Use a implementação do transporte do WebSphere MQ para SOAP no .NET Framework 1, .NET Framework 2 e Axis 1.4 como um guia.

Instalando e verificando serviços da web do WebSphere MQ

Use as instruções nesses tópicos para instalar e verificar o transporte WebSphere MQ para SOAP.

Instalando o transporte da web do WebSphere MQ para SOAP

Use estas instruções para instalar o transporte da web do WebSphere MQ para SOAP. A instalação cria ferramentas para executar clientes ou serviços de serviço da Web usando o WebSphere MQ como o transporte SOAP As ferramentas são usadas nos ambientes SOAP .NET Framework 1, .NET 2, Axis 1.4ou Axis2 .

Antes de começar

Verifique os produtos com pré-requisitos em Requisitos do sistema para IBM WebSphere MQ. O processo de instalação não verifica a presença e a disponibilidade do software obrigatório. Deve-se verificar se os pré-requisitos estão instalados.

O WebSphere MQ fornece uma cópia do tempo de execução do Axis 1.4 Use esta versão com o WebSphere MQ em vez de qualquer outra que você possa ter instalado A IBM não fornece suporte técnico para o eixo Apache . Entre em contato com a Apache Software Foundation se tiver problemas técnicos com ele.

Para executar serviços da Web no ambiente SOAP do .NET Framework 3, o WebSphere MQ usa o Windows Communication Foundation O canal customizado do WebSphere MQ para o Windows Communication Foundation executa clientes e serviços da web usando o WebSphere MQ como um transporte para mensagens SOAP.

Sobre esta tarefa

É possível instalar o transporte da Web do WebSphere MQ para SOAP como um cliente MQI do WebSphere MQ ou aplicativo do Servidor. Se você já tiver instalado o WebSphere MQ como um cliente ou servidor em seu computador, verifique se instalou os componentes listados.

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

Execute as etapas de instalação a seguir.

Procedimento

1. Selecione o componente "Java e. Net Messaging e Serviços da Web" para instalação
2. No Solaris e HP-UX, selecione o componente "Java Runtime Environment" para instalação.
3. Selecione o kit de ferramentas de desenvolvimento para instalação.
4. Instale e verifique o WebSphere MQ conforme descrito na Iniciação rápida para sua plataforma.
5. Copie o tempo de execução do Apache Axis 1.4 , `axis.jar` do diretório `prereqs/axis` na mídia de instalação do WebSphere MQ . Copie-o para o diretório de instalação descrito em Tabela 138 na página 970, Tabela 139 na página 970 ou Tabela 140 na página 970.

   Windows

   ```
   Copy D:\PreReqs\axis\axis.jar MQ_INSTALLATION_PATH\java\lib\soap
   ```

   AIX

   ```
   cp -f PreReqs/axis/axis.jar MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chown mqm:mqm MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chmod 444 MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   ```

Diretórios de instalação do HP-UX, Solaris e Linux (todas as plataformas)

```
cp -f PreReqs/axis/axis.jar MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
chown mqm:mqm MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
chmod 444 MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
```

6. No Windows 2003, execute **Aspnet_regiis.exe** para atualizar os mapas de script para apontar para a versão do Common Language Run Time que você está usando.

   Procure o utilitário **Aspnet_regiis.exe** em %SystemRoot%Microsoft.NET/Framework/*version-number*
7. Configure a variável de ambiente WMQSOAP_HOME para apontar para o diretório de instalação do WebSphere MQ .

Resultados

Tabela 138. diretórios de instalação do Windows

Local	**Conteúdos**
MQ_INSTALLATION_PATH\programs\bin	Arquivos binários, de comando, DLL e executáveis
MQ_INSTALLATION_PATH\programs\java\lib	.jar files
MQ_INSTALLATION_PATH\programs\java\lib\soap	SOAP .jar files
MQ_INSTALLATION_PATH\programs\soap\samples	Amostras e IVT

Tabela 139. AIX diretórios de instalação

Local	**Conteúdos**
MQ_INSTALLATION_PATH/bin	Shell scripts
MQ_INSTALLATION_PATH/java/lib	.jar files
MQ_INSTALLATION_PATH/java/lib/soap	axis.jar e outros arquivos JAX-RPC .jar
MQ_INSTALLATION_PATH/samp/soap	Amostras e IVT

Tabela 140. Diretórios de instalação do HP-UX, Solaris e Linux (todas as plataformas)

Local	**Conteúdos**
MQ_INSTALLATION_PATH/bin	Shell scripts
MQ_INSTALLATION_PATH/java/lib	.jar files
MQ_INSTALLATION_PATH/java/lib/soap	axis.jar e outros arquivos JAX-RPC .jar
MQ_INSTALLATION_PATH/samp/soap	Amostras e IVT

Como proceder a seguir

1. Somente para .NET, você deve registrar o transporte WebSphere MQ para arquivos SOAP com o Cache de Conjunto Global. Se .NET já estiver instalado ao instalar o WebSphere MQ, o registro será executado automaticamente na instalação. Se você instalar .NET após WebSphere MQ, o registro será executado automaticamente quando o IVT for executado pela primeira vez.

   É possível executar o **amqiregisterdotnet.cmd** para executar o registro dos conjuntos .NET Também é possível executar **amqiregisterdotnet.cmd** para forçar o novo registro em qualquer

estágio. Uma vez realizado, esse registro sobrevive a reinicializações do sistema e o novo registro subsequente normalmente não é necessário.

2. Execute o Teste de verificação de instalação, conforme descrito em "Verificando o IBM WebSphere MQ Transport for SOAP" na página 971.
3. Se pretende desenvolver o cliente Axis2, deve-se fazer download do Axis2 1.4.1 a partir da Apache;, consulte "Desenvolvendo um Cliente JAX-WS para WebSphere Transport for SOAP Usando Eclipse" na página 994.

Verificando o IBM WebSphere MQ Transport for SOAP

Verifique se o IBM WebSphere MQ Transport for SOAP está usando o comando **runivt**. O comando executa vários aplicativos de demonstração e assegura que o ambiente está corretamente configurado após a instalação.

Antes de começar

Antes de executar o comando **runivt**, assegure-se de que você tenha os seguintes ambientes de tempo de execução:

- Para executar somente no Axis: deve-se ter um Java SDK (dentro do SOE) disponível no sistema. Também deve-se incluir o local dos comandos `java.exe` e `javac.exe` na variável de ambiente **PATH** dos sistemas.
- Para executar um teste apenas no .NET (suportado apenas no Windows): deve-se ter um Java SDK e os compiladores e ferramentas .NET em seu sistema.. Para fazer isso, acesse um prompt de comandos do Visual Studio ou o prompt de comandos do SDK Microsoft Windows , em seguida, inclua o local dos arquivos `java.exe` e `javac.exe` na variável de ambiente **PATH** .
- Para executar todos os testes disponíveis: para as plataformas Windows , o ambiente deve ser configurado conforme descrito na execução de teste .NET Em plataformas do UNIX and Linux, o ambiente deve ser configurado conforme descrito na execução de teste para apenas Axis.

Sobre esta tarefa

Em vez de executar o teste de verificação no .NET e no Axis, talvez você queira executar o teste apenas no Axis ou somente no .NET

Se você tiver problemas com os testes e desejar iniciar novamente:

1. Pare o gerenciador de filas `WMQSOAP.DEMO.QM` usando a opção `immediate`.
2. Pare o listener que foi iniciado em uma janela diferente.
3. Exclua o gerenciador de filas.
4. Exclua o diretório de amostras temporário que você criou e inicie novamente.

Em plataformas UNIX and Linux , você deve executar o comando usando uma sessão do sistema X Windows .

O comando **runivt** muda o conteúdo do diretório `soap/samples`. Para manter a imagem de instalação inalterado, copie o diretório de amostras para um local temporário e execute o teste de verificação do local temporário.

É possível executar a verificação da instalação, quantas vezes desejar.

Execute as etapas a seguir para verificar a instalação do IBM WebSphere MQ transport for SOAP on .NET Framework 1, .NET Framework 2 e Axis 1.4:

Procedimento

1. Copie a árvore de diretórios `./tools/soap/samples` para um local temporário.
2. Inicie uma janela de comandos com o diretório temporário como o diretório atual.
3. Use o comando **runivt** para iniciar o teste de instalação. O script `runivt` compila uma classe de teste, o cliente de amostra e serviços antes de implementá-los e executá-los. Para a classe de teste,

o cliente de amostra e os serviços a serem executados, conclua as etapas de instalação descritas em Instalando o WebSphere(r) MQ Web Transport for SOAP e assegure-se de que o prompt de comandos usado para executar o comando `runivt` tenha o conjunto de ambiente de tempo de execução necessário.. Use qualquer um dos seguintes métodos para executar o comando **runivt**:

- Execute um teste apenas no Axis: `runivt Axis`.
- Execute um teste somente no .NET (suportado apenas no Windows): `runivt DotNet`.
- Execute todos os testes disponíveis:`runivt`.

Para obter mais informações sobre os parâmetros e sintaxe do comando `runivt`, consulte **runivt**: teste de verificação de instalação do IBM WebSphere MQ Transport for SOAP. Os testes que podem ser executados são listados no arquivo `ivttests.txt` nas plataformas Windows e `ivttests_unix.txt` nas UNIX and Linux

Referências relacionadas

runivt: WebSphere MQ transporte para teste de verificação de instalação SOAP

Desenvolvendo Serviços da web para o transporte WebSphere MQ para SOAP

Use seu ambiente de desenvolvimento de serviço da Web normal para desenvolver serviços para uso com o transporte do WebSphere MQ para SOAP

Antes de começar

1. Se você estiver planejando usar as ferramentas de linha de comandos fornecidas com o transporte do WebSphere MQ para SOAP:
   a. Crie um diretório de implementação para o serviço.
   b. Inicie uma janela de comandos no diretório.
   c. Para .NET, csc.exe e wsdl.exe devem estar no caminho e ser da mesma versão do .NET Framework.
   d. Para Java,
      i) Execute o comando **amqwsetcp** para configurar o caminho de classe.
      ii) Um IBM JRE e um JDK no mesmo nível de versão devem estar no caminho atual O nível de versão deve ser pelo menos 5.0.
      iii) Customize o caminho de classe para incluir as localizações de quaisquer bibliotecas adicionais `.jar` e diretórios que contêm pacotes .java, inclusive para o serviço que você está desenvolvendo. Coloque o diretório atual "`.`" no caminho de classe..
      iv) Crie um diretório, relativo ao diretório atual da janela de comandos, correspondente ao nome do pacote do serviço que você está desenvolvendo.
2. Como alternativa, use as ferramentas do ambiente de trabalho que suportam o desenvolvimento de serviços da web. As tarefas de desenvolvimento de exemplo usam Microsoft Visual Studio 2008, Eclipse IDE para Java EE Developers e WebSphere Application Server Community Edition.

Sobre esta tarefa

Os serviços da Web existentes não precisam de modificação para trabalhar com o transporte do WebSphere para SOAP As ferramentas fornecidas com o transporte WebSphere MQ para SOAP implementam um serviço da Web e o executam usando um listener SOAP do WebSphere MQ . As ferramentas também geram stubs de cliente WSDL, .NET e classes de proxy `.java` para desenvolver o transporte do WebSphere MQ para clientes SOAP

Siga estas etapas para criar um serviço e prepará-lo para a implementação e a geração de clientes. Siga as etapas nas tarefas relacionadas para criar um serviço usando Eclipse ou Microsoft Visual Studio 2008.

Procedimento

1. Desenvolva o serviço usando o seu ambiente de desenvolvimento normal.

2. Teste o serviço usando um cliente de serviços da web HTTP
3. Siga estas etapas para preparar o diretório de implementação:
   - Para Java
     a. Copie o arquivo `.java` definindo a interface de serviço no diretório de implementação.
     b. Copie quaisquer arquivos `.class` para o serviço no diretório correspondente ao nome do pacote.
     c. Verifique se o caminho de classe pode localizar todas as classes que são necessárias: compile o arquivo `.java` do serviço usando **`javac`**.
   - Para .NET
     a. Copie o arquivo `.asmx` definindo o serviço no diretório de implementação e
     b. Se estiver usando o modelo code-behind, copie quaisquer arquivos `.dll` em um diretório *`deployment directory`*`\bin` .

Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse

Desenvolva um serviço da web Axis 1.4 para executar usando o WebSphere MQ como seu provedor de serviços. Use seu ambiente de desenvolvimento de serviço da Web normal para criar um serviço para implementação para o Axis 1.4

Antes de começar

Leve em conta os requisitos para implementar um servidor da Web no listener SOAP WebSphere MQ para Axis 1.4.

- O listener SOAP do WebSphere MQ para Axis 1.4 requer um IBM JRE na versão 5.0 ou superior. O JRE e JDK usados para desenvolvimento devem estar no mesmo nível de versão.
- O listener SOAP do WebSphere MQ para Axis 1.4 requer o `axis.jar` instalado com o WebSphere MQ. Altere o caminho de construção em seu ambiente de desenvolvimento para fazer referência ao arquivo `axis.jar` instalado com o WebSphere MQ, em vez dos arquivos `axis.jar` instalados com o ambiente de desenvolvimento.
- Até, e incluindo, WebSphere MQ V7.0.1, o WSDL gerado para o serviço implementado é RPC/codificado. A partir da V7.1, também é possível solicitar o WSDL de estilo RPC/literal O WSDL gerado é usado apenas para implementação É possível definir o serviço usando o WSDL compatível com WS-I

Sobre esta tarefa

Crie o serviço usando seu ambiente normal de desenvolvimento de serviços da Web (Web Services Development Environment)

Nesta tarefa, usamos o IDE Eclipse Java EE de software livre livremente disponível para Web Developers, conhecido como Galileo. Para o servidor de aplicativos, usamos WebSphere Application Server Community Edition v2.1 (Community Edition), com base no Geronimo. Consulte as tarefas relacionadas para obter informações sobre como obter, instalar e configurar o IDE e o servidor

Teste o serviço usando HTTP como um transporte usando o Web Services Explorer fornecido no IDE antes. Como alternativa, gere um proxy de cliente HTTP e teste o serviço usando seu próprio código do cliente..

Você pode seguir estas etapas para desenvolver um serviço da Web ascendente. Use o programa de amostra `StockQuoteAxis.java` como um exemplo

Procedimento

1. Inicie o Eclipse IDE para Java EE Developers com uma nova área de trabalho.
2. Configure a área de trabalho para usar Java50,

WebSphere Application Server Community Edition 2.1.4 não funciona com Java60.

a) **Janela** > **Preferências** > **Java** > **JREs instalados** > **Incluir ...** > **VM padrão** > **Avançar** > **Diretório ...**

b) Navegue para o diretório de instalação do **Java50** > **OK** > **Concluir**

c) Verifique o **Java50** JRE> OK

3. Inclua o ambiente de tempo de execução Community Edition e inicie o Community Edition.

a) **Janela** > **Preferências** > **Servidor** > **Ambientes de Tempo de Execução** > **Incluir ...**

b) Selecione **IBM WASCE v2.1** na lista **Novo ambiente de tempo de execução do servidor** > Marque **Criar um novo servidor local** > **Avançar** ..

Se **IBM WASCE 2.1** não estiver na lista, será necessário concluir duas outras tarefas:

i) Instale o WebSphere Application Server Community Edition.

ii) Instale a atualização do Eclipse para Community Edition.

Localize os detalhes em WebSphere Application Server Community Edition

c) Procure o Diretório de Instalação do Servidor de Aplicativos> **OK** > **Concluir** > **OK**.

d) Clique com o botão direito em **IBM WASCE v2.1 server** na visualização de servidores> **Iniciar**

Sugestão: É possível gerenciar o WASCE no Eclipse: clique com o botão direito do mouse em **IBM WASCE v2.1 server** > **Ativar Console do WASCE** . O padrão **Username** e **Password** são `system` e `manager`

4. Configure o servidor e o tempo de execução para os Serviços da web

a) **Janela** > **Preferências** > **Serviços da Web** > **Servidor e Tempo de Execução**

b) Selecione o **IBM WASCE v2.1 Server** como o servidor

c) Deixe o **Apache Axis** como o tempo de execução do serviço da Web

5. Crie um Projeto da Web Dinâmico

a) **Arquivo** > **Novo** > **Projeto Dinâmico da Web**

Nomeie o projeto como `StockQuoteAxis`

b) Marque **Incluir projeto em um EAR** > **Novo ...** .

c) Na página **Projeto do Aplicativo EAR** , digite **Project name** `StockQuoteAxisEAR` > **Concluir**

Responda **OK** em resposta à caixa de diálogo, sugerindo que você alterne para a perspectiva Java EE ou permaneça na perspectiva Java para seguir essas instruções exatamente

d) O **IBM WASCE 2.1 server** é selecionado como o tempo de execução de destino Aceite e os outros padrões> **Concluir**.

Responda **OK** em resposta à caixa de diálogo, sugerindo que você alterne para a perspectiva Java EE ou permaneça na perspectiva Java para seguir essas instruções exatamente

6. Importe o programa de amostra `StockQuoteAxis.java`

a) Abra o projeto da web **StockQuoteAxis** > Clique com o botão direito na pasta **src** > **Importar ...**

b) Selecione **Geral** > **Sistema de Arquivos** > **Próximo**

c) Navegue até *MQ_INSTALLATION_PATH*`\tools\soap\samples\java\server` > check **StockQuoteAxis.java** > **Finish**

MQ_INSTALLATION_PATH representa o diretório de alto nível onde o WebSphere MQ está instalado. Você deve destacar o diretório do servidor para ver os arquivos nele contidos.

7. Corrija o erro de compilação movendo o `StockQuoteAxis.java` para seu pacote correto

a) Abra `StockQuoteAxis.java` e clique com o botão direito no problema> **Correção Rápida**

b) Clique duplo em **Mover 'StockQuoteAxis.java' para o pacote 'soap.server'** > **Salvar**.

8. Crie um serviço da web do `StockQuoteAxis.java` .

a) Clique com o botão direito em `StockQuoteAxis.java` > **Serviços da web** > **Criar serviço da web** > **Avançar**.

Aceite a configuração padrão para o serviço:

Tipo de Serviço da Web
Serviço Java beanWeb de baixo para cima

Implementação de serviço..
soap.server.StockQuoteAxis

Servidor
`IBM WASCE v2.1 server`

Tempo de execução do serviço da web
Eixo do Apache

Projeto de serviço
Eixo StockQuote

Projeto de EAR de serviço
StockQuoteAxisEAR

Configuração
Nenhuma geração do cliente

9. Selecione os métodos para acessar e o estilo de serviço da Web> **Avançar**.

   Inicie o servidor, se solicitado.

   a) Deixe todos os métodos selecionados

   b) Selecione o estilo de documento / literal (agrupado)

10. **Concluir**

    Quando o serviço tiver sido implementado, consulte a pasta `WebContent\wsdl` no projeto da Web `StockQuoteAxis` e localize o arquivo `StockQuoteAxis.wsdl` gerado

11. Teste o serviço usando HTTP com o Web Services Explorer.

    a) Clique com o botão direito em `StockQuoteAxis.wsdl` > **Testar com o Web Services Explorer**.

    b) Clique na operação **getQuote** nas ações de **StockQuoteAxisSoapLigação** na janela **Web Services Explorer**

    c) Digite `ibm` no campo de entrada **symbol** > **Go**

12. Teste o serviço usando o transporte WebSphere MQ para SOAP.

    Para implementar o serviço, use o **`SimpleJavaListener`**, que está em `com.ibm.mq.soap.jar` É necessário incluir as bibliotecas Java e SOAP do WebSphere MQ no caminho de compilação

    a) Clique com o botão direito no projeto da web **StockQuoteAxis** > **Caminho de Construção** > **Configurar Caminho de Construção ...**

    b) Clique na guia **Bibliotecas** > **Incluir Jars Externos ...**.. Procure por *`MQ_INSTALLATION_PATH`*`\java\lib` . e selecione todos os arquivos `.jar` > **Abrir** > **Incluir Jars Externos ...** Procure *`WMQ Install directory`*`\java\lib\soap` e selecione todos os arquivos `.jar` > **Abrir** > **OK**.

    `MQ_INSTALLATION_PATH` representa o diretório de alto nível onde o WebSphere MQ está instalado.

    c) No Explorador de Projetos, clique com o botão direito em `StockQuoteAxis\Java Resources\Libraries\com.ibm.mq.soap.jar\com.ibm.mq.soap.transport.jms\SimpleJavaListener.class\ SimpleJavaListener` > **Executar como ...**> **Configurações de Execução ...**

    Sugestão:

    Se não houver nenhuma configuração para o `SimpleJavaListener` , clique no ícone **Nova configuração** na página **Criar, gerenciar e executar configurações** do assistente **Executar configurações** ,

    O **`SimpleJavaListener`** não possui nenhum comando para pará-lo, Para monitorar ou parar **`SimpleJavaListener`**, abra a **Perspectiva de depuração** no Eclipse.

d) Abra a aba **(x) = Argumentos** . Na área de entrada **Argumentos do programa** , digite os parâmetros para **`SimpleJavaListener`**

Para este exemplo, digite

```
-u "jms:/queue?destination=REQUESTAXIS@QM1&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
&targetService=StockQuoteAxis
&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE" -n 10
```

Nota: O serviço de destino é `StockQuoteAxis` para corresponder ao nome do serviço de destino criado no descritor de implementação de serviço, `StockQuoteAxis\WebContent\WEB-INF\server-config.wsdd` amqwdeployWMQService cria um serviço de destino chamado `soap.server.StockQuoteAxis`.. Neste exemplo, você está usando o mesmo `StockQuoteAxis.class` e `service-config.wsdd` como o servidor HTTP

e) Na mesma guia, configure **Diretório ativo** para fazer referência ao arquivo `server-config.wsdd` : `${workspace_loc:StockQuoteAxis/WebContent/WEB-INF}`

a) **Executar**

Erros são gravados no console. Se o console permanecer em branco, **`SimpleJavaListener`** foi iniciado ok.

b) Para testar a implementação, execute um cliente Axis StockQuote, desenvolvido na tarefa "Desenvolvendo um cliente JAX-RPC para o transporte WebSphere para SOAP usando Eclipse" na página 986.

Exemplo: programa de amostra StockQuoteAxis

O serviço da web Java de amostra, `StockQuoteAxis.java`, é instalado em *`WMQ install directory`*`\tools\soap\samples\java\server` `StockQuoteAxis.java`, Figura 170 na página 977, tem quatro métodos:

1. `float getQuote(String symbol)`
2. `void getQuoteOneWay(String symbol)`.
3. `int asyncQuote(int delay)`
4. `float getQuoteTran(String symbol)`

```
package soap.server;
import java.lang.Thread;
import java.io.FileWriter;
public class StockQuoteAxis {
    public float getQuote(String symbol) throws Exception {
        return ((float) 55.25);
    }
    public void getQuoteOneWay(String symbol) throws Exception {
        try {
            // Write the results for this service to a file
            FileWriter f = new FileWriter("getQuoteOneWay.txt", true);
            f.write("One way service result via proxy is: 44.44\n");
            f.close();
        } catch (Exception ee) {
            System.out.println("Error writing result file in getQuoteOneWay");
            ee.printStackTrace();
        }
    }
    public int asyncQuote(int delay) {
        try {
            Thread.sleep(delay);
        } catch (Exception e) {
            System.out.println("Exception in asyncQuote during sleep");
        }
        return delay;
    }

    public float getQuoteTran(String symbol) throws Exception {
        if (symbol.equalsIgnoreCase("ROLLBACK")) {
            System.out.println("Rollback was requested,
                exiting from service by calling System.exit().");
            System.exit(0);
        }
        return ((float) 55.25);
    }
}
```

Figura 170. Eixo StockQuote

Como proceder a seguir

Implemente o serviço usando WebSphere MQ Transport for SOAP, em vez de HTTP usando o comando **amqwdeployWMQService**.

O comando tem uma opção `axisDeploy`, que implementa o serviço criando um descritor de implementação Apache Axis 1.4 . O listener SOAP do WebSphere MQ executa o serviço O listener SOAP é chamado de Listener SimpleJavae é fornecido com o transporte WebSphere MQ para SOAP

Tarefas relacionadas

Desenvolvendo um serviço .NET 1 ou 2 para o transporte WebSphere MQ para SOAP usando o Microsoft Visual Studio 2008
Desenvolva o serviço da web de cotação SampleStockpara .NET 1 ou .NET 2 usando o Microsoft Visual Studio 2008

Desenvolvendo um Serviço da Web EJB JAX-WS para SOAP W3C sobre JMS
Um serviço da Web vinculado à recomendação de candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos JEE Essa tarefa é a etapa 2 de conectar um cliente de serviço da Web Axis2 e um serviço da Web implementado no WebSphere Application Server usando o protocolo W3C SOAP sobre JMS.

Desenvolvendo um serviço .NET 1 ou 2 para o transporte WebSphere MQ para SOAP usando o Microsoft Visual Studio 2008

Desenvolva o serviço da web de cotação SampleStockpara .NET 1 ou .NET 2 usando o Microsoft Visual Studio 2008

Sobre esta tarefa

Crie o serviço StockQuote com uma implementação code-behind usando o Visual Studio 2008.

Procedimento

1. Crie um modelo para o serviço e verifique se ele é executado em HTTP.
   a) Inicie o Visual Studio 2008 > **Arquivo** > **Novo** > **Projeto...**. Selecione **C#** Tipo de projeto, **, NET Framework 2**e **ASP.NET Aplicativo de Serviço da Web** Digite o **Nome:** e o **Nome da solução:** `StockQuoteDotNet` > **OK**
   b) Clique com o botão direito em **Service1.asmx** no **Solution Explorer** > **Renomear** > `StockQuote.asmx`.
   c) Altere o fragmento de código `public class Service1` para `public class StockQuote`
   d) Clique com o botão direito em **StockQuote.asmx** no **Explorador de Soluções** > **Abrir com ...** > **Editor XML**. Altere `Class="StockQuoteDotNet.Service1"` para `Class="StockQuoteDotNet.StockQuote"`
   e) Altere o fragmento de código `[WebService(Namespace = "http://tempuri.org/")]` para `[WebService(Namespace = "http://stock.samples/")]`
   f) Remova a linha de código `[ToolboxItem(false)]`.
   g) Verifique se tudo está correto até agora: **Depurar** > **Iniciar depuração (F5)**. Verifique a saída no Explorer.
2. Inclua os métodos da amostra `SQDNNonInline.asmx.cs` e teste o serviço em HTTP.
   a) Abra o *MQ_INSTALLATION_PATH* `\tools\soap\samples\dotnet\SQDNNonInline.asmx.cs` e substitua o método `HelloWorld` com os quatro métodos `Quote`; consulte Figura 171 na página 979.*MQ_INSTALLATION_PATH* representa o diretório onde o WebSphere MQ está instalado.
   b) **Construir** > **Reconstruir** a solução > Clique com o botão direito em um dos **Encadeamentos** com erro > **Resolver** > Usando **System.Threading**.
   c) Pressione F5 para iniciar a depuração.

      O serviço não tem conformante com o WS-I Basic Profile v1.1. Você tem a opção de mudar a anotação WebMethod de `[SoapRpcMethod]` para `[SoapDocumentMethod]` ou remover a anotação `[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]`.
   d) Pressione F5 para verificar a implementação usando HTTP.
3. Gere WSDL, clientes e execute o serviço utilizando o transporte WebSphere MQ para SOAP.
   a) Abra uma janela de comando na árvore de diretórios do projeto, em que o `StockQuote.asmx` é armazenado.
   b) (Opcional) Use amqwdeployWMQService para gerar artefatos. O gerenciador de filas deve ser iniciado:

      ```
      amqswdeployWMQService -f StockQuote.asmx
      -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
      &connectionFactory=()
      &destination=REQUESTDOTNET@QM1
      &targetService=StockQuote.asmx"
      StockQuote.asmx StockQuote.wsdl
      ```

      Todos os artefatos são criados na árvore de diretórios `./generated`.
   c) (Opcional) Gere apenas o WSDL para chamar o serviço usando o transporte WebSphere MQ para SOAP.

      ```
      amqswsdl -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
      &connectionFactory=()
      &destination=REQUESTDOTNET@QM1
      &targetService=StockQuote.asmx"
      StockQuote.asmx StockQuote.wsdl
      ```
   a) Execute o listener do .NET Use `.\generated\server\startWMQNListener.cmd` ou digite o comando:

```
amqSOAPNETListener -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=()
&destination=REQUESTDOTNET@QM1
&targetService=StockQuote.asmx"
```

4. Teste o serviço usando um cliente gerado a partir do WSDL ou usando os clientes gerados pelo **amqwdeployWMQService**.

Código de Amostra

O serviço da web .NET de amostra, `StockQuoteDotNet`, é instalado em *MQ_INSTALLATION_PATH*`\tools\soap\samples\dotnet` *MQ_INSTALLATION_PATH* é o diretório onde o WebSphere MQ está instalado. A ligação de serviço da web das amostras publicadas diferem um pouco da ligação usada na tarefa. A tarefa usa os padrões usados no Visual Studio 2008.

Há dois exemplos de serviços da Web .NET Framework 1 e .NET Framework 2. `StockQuoteDotNet.asmx`, é um serviço sequencial.. `SQDNNoninline.asmx`, é um serviço da Web de código por trás implementado pelo `SQDNNoninline.asmx.cs`

`StockQuoteDotNet` tem quatro métodos:

1. `float getQuote(String symbol)`
2. `void getQuoteOneWay(String symbol).`
3. `int asyncQuote(int delay)`
4. `float getQuoteDOC(String symbol)`

```
<%@ WebService Language="C#" Class="StockQuoteDotNet" %>
using System;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.Services.Description;
using System.Threading;
[WebService (Namespace="http://stock.samples")]
public class StockQuoteDotNet {
     [WebMethod] [ SoapRpcMethod(OneWay=true) ]
     public void getQuoteOneWay(String symbol) {
          if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(5000);
          System.Console.WriteLine("getQuoteOneWay was invoked.");
     }
     [WebMethod] [SoapRpcMethod]
     public float getQuote(String symbol) {
          if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(10000);
          return 88.88F;
     }
     [WebMethod] [SoapRpcMethod]
     public int asyncQuote(int delay) {
          Thread.Sleep(delay);
          return delay;
     }
     [WebMethod]
     public float getQuoteDOC(String symbol) {
          return 77.77F;
     }
}
```

Figura 171. Serviço sequencial: `StockQuoteDotNet.asmx`

```
<%@ WebService Language="C#" Codebehind="SQDNNonInline.asmx.cs" Class="SQDNNonInline" %>
```

Figura 172. Code-behind: design `SQDNNonInline.asmx`

```
using System;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.Services.Description;
using System.Threading;

[WebService(Namespace = "http://stock.samples")]
public class SQDNNonInline : System.Web.Services.Protocols.SoapHttpClientProtocol
{

    [WebMethod]
    [SoapRpcMethod(OneWay = true)]
    public void getNonInlineQuoteOneWay(String symbol)
    {
        if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(5000);
        System.Console.WriteLine("getNonInlineQuoteOneWay was invoked.");
    }

    [WebMethod]
    [SoapRpcMethod]
    public float getNonInlineQuote(String symbol)
    {
        if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(10000);
        return 88.88F;
    }

    [WebMethod]
    [SoapRpcMethod]
    public int asyncNonInlineQuote(int delay)
    {
        Thread.Sleep(delay);
        return delay;
    }
    [WebMethod]
    public float getNonInlineQuoteDOC(String symbol)
    {
        return 77.77F;
    }
}
```

Figura 173. Code-behind: implementação: SQDNNonInline.asmx.cs

Tarefas relacionadas

Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse
Desenvolva um serviço da web Axis 1.4 para executar usando o WebSphere MQ como seu provedor de serviços. Use seu ambiente de desenvolvimento de serviço da Web normal para criar um serviço para implementação para o Axis 1.4

Desenvolvendo um Serviço da Web EJB JAX-WS para SOAP W3C sobre JMS
Um serviço da Web vinculado à recomendação de candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos JEE Essa tarefa é a etapa 2 de conectar um cliente de serviço da Web Axis2 e um serviço da Web implementado no WebSphere Application Server usando o protocolo W3C SOAP sobre JMS.

Desenvolvendo um Serviço da Web EJB JAX-WS para SOAP W3C sobre JMS

Um serviço da Web vinculado à recomendação de candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos JEE Essa tarefa é a etapa 2 de conectar um cliente de serviço da Web Axis2 e um serviço da Web implementado no WebSphere Application Server usando o protocolo W3C SOAP sobre JMS.

Antes de começar

Use Rational Application Developer para criar o serviço da Web EJB. O assistente de serviço da web no Rational Application Developer tem uma opção de criar um serviço da web usando a recomendação de candidato W3C para a ligação SOAP sobre JMS O Rational Application Developer 7.54 é necessário O exercício usado Rational Application Developer incluído no Rational Software Architect para WebSphere Software v7.5.5.1,

O EJB é implementado no WebSphere Application Server do Rational Application Developer como parte desta tarefa. Você deve concluir o "Configurando o WebSphere Application Server para usar o W3C SOAP sobre JMS" na página 1018

Para criar o WSDL realmente usado na tarefa, deve-se primeiro concluir a tarefa, "Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse" na página 973 Em seguida, é possível importar o WSDL do projeto da Web Dinâmico na área de trabalho do Eclipse Galileo ou do serviço da Web HTTP em execução implementado no WASCE.

WebSphere Application Server ainda pode estar em execução como resultado de "Configure recursos do WebSphere Application Server" na página 1020. Se não estiver, é possível iniciá-lo a partir da visualização Servidores no RAD.

Sobre esta tarefa

Nesta tarefa, você reimplementa o serviço `StockQuoteAxis` da execução como um serviço do Eixo JAX-RPC executado pelo **`SimpleJavaListener`** usando o transporte WebSphere MQ para SOAP, para ser um serviço JAX-WS em execução no servidor de aplicativos WebSphere usando o protocolo W3C SOAP sobre JMS.

Há duas partes para migrar o serviço do **`SimpleJavaListener`** para o WebSphere Application Server:

1. Gere a interface de serviço da Web a partir do WSDL para o serviço usando o assistente de serviço da Web EJB de Top-down no Rational Application Developer
2. Implementando o serviço importando o WebSphere MQ SOAP de amostra `StockQuoteAxis.java`

Uma abordagem alternativa teria sido gerar o serviço de forma ascendente, a partir do `StockQuoteAxis.java`. No entanto, para ter certeza de que a interface para o serviço migrado seja idêntica, a abordagem descendente é melhor, pois usa o mesmo WSDL.

O serviço da Web é desenvolvido para o contêiner EJB e não para o contêiner Web porque o suporte JMS faz parte do contêiner EJB.

Procedimento

1. Inicie o Rational Application Developere verifique se o WebSphere Application Server está em execução
   a) Inicie o Rational Application Developer em uma nova área de trabalho
   b) Abra a perspectiva do Java EE
   c) Abra a guia **Servidores** e marque WebSphere Application Server está em execução.
      - Se não houver nenhum WebSphere Application Server v7.0 na visualização, clique com o botão direito na visualização> **Novo** > **Servidor**. Siga as opções no assistente para criar uma instância do WebSphere Application Server v7.0 ..
      - Se o servidor estiver presente, mas não iniciado, clique na ponta da seta para iniciá-lo.
      - Para verificar as propriedades e obter acesso rápido aos logs do servidor, clique com o botão direito em **WebSphere Application Server v7.0 no host local** > **Propriedades** > **WebSphere Application Server**.
      - Para administrar o servidor, use um navegador externo e abra a URL, `http://localhost:9061/ibm/console/unsecureLogon.jspou` clique com o botão direito em **WebSphere Application Server v7.0 no host local** > **Executar console administrativo**.
      - A configuração padrão é publicar automaticamente. Muitas pessoas preferem implementar atualizações no servidor manualmente. Clique duas vezes em **WebSphere Application Server v7.0 no host local**e expanda a opção **Publicando** na janela**Visão Geral** Clique em **Nunca publicar automaticamente**.
      - Outro padrão que você pode desejar alterar é limpar a caixa de seleção **Finalizar o servidor no encerramento do ambiente de trabalho** na janela **Visão geral**.
2. Criar os projetos JEE

Você deve criar um Enterprise Application Project (EAR) e um Enterprise Java Bean (EJB) Project.

a) **Arquivo** > **Novo** > **Projeto do aplicativo corporativo**. Nomeie o projeto como `W3CJMSEAR` > **Concluir**

Os padrões devem identificar WebSphere Application Server v7.0 como o tempo de execução de destino e a versão EAR 5.0. A configuração padrão deve ser selecionada.

b) **Arquivo** > **Novo** > **Projeto EJB**. Nomeie o projeto como `W3CJMSEJB` Selecione `W3CEARJMS` como o **Nome do Projeto EAR** > **Avançar**.

A versão do módulo EJB padrão é 3.0 e a configuração padrão será usada novamente.

c) Desmarque a caixa de seleção **Criar um módulo JAR do Cliente EJB** > **Concluir**

3. Gere e implemente o serviço da web EJB a partir do WSDL `StockQuoteAxis`.

a) **Executar** > **Ativar o Web Services Explorer**.

b) Selecione a página WSDL usando os ícones na janela**Web Services Explorer** > clique em **WSDL principal** no Navigator..

c) Na janela **Ações**, digite ou procure a URL do WSDL para `StockQuoteAxis.wsdl`.

Se você tiver WASCE em execução com o `StockQuoteAxis` implementado como um serviço HTTP, a URL será:

```
http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl
```

Se tiver o WSDL no sistema de arquivos, a URL poderá ser:

```
File:\Dirpath\StockQuoteAxis\WebContent\wsdl\StockQuoteAxis.wsdl
```

d) Clique na linha que contém a URL importada na árvore do Navegador.

É a linha imediatamente após **WSDL principal**, se for o primeiro WSDL importado para o Web Services Explorer.

e) Na janela **Ações**, clique em **Ativar assistente de serviço da web** > **Estrutura básica de serviço da web** > **Ir**

f) No assistente de serviço da web, selecione **Serviço da web EJB descendente**

Selecione ou verifique a Configuração usando informações da Tabela 141 na página 982. Marque **Sobrescrever arquivos sem aviso** > **Avançar**.

Tabela 141. Configuração de serviço da web EJB descendente

Campo	**Value**
Servidor	`WebSphere Application Server v7.0`
Tempo de execução de serviço da web	`IBM WebSphere JAX-WS`
Projeto de serviço	`W3CJMSEJB`
Projeto de EAR de serviço	`W3CJMSEAR`
Configuração:	`No client generation`

g) Na página com subtítulo, **Especifique opções para criar um WebSphere EJB JAX-WS Top Down Web Service**, marque a caixa **Alternar para ligação JMS**. Além disso, marque **Ativar Estilo de Wrapper**, **Copiar WSDL para o projeto** e **Gerar Descritor de Implementação de Serviço da web** > **Avançar**

h) Na página intitulada **WebSphere JAX-WS**, marque **Usar protocolo de interoperabilidade SOAP/JMS** e forneça valores de Tabela 142 na página 983, deixando outros campos em branco> **Avançar**

Tabela 142. WebSphere Configuração de Ligação JMS JAX-WS	
Campo	**Value**
Destino JMS	queue
Nome JNDI de destino:	requestaxis
Connection factory de JMS	qm1
Nome para resposta	W3CJMSEAR
Configuração:	replyaxis

a) Na página com o título **Configuração do projeto de roteador do WebSphere JAX-WS**, digite qm1as no campo **Nome JNDI de ActivationSpec** > **Avançar**.

O RAD leva cerca de 30 segundos a um minuto para gerar e implementar o projeto.

b) Ignore as opções na página **Publicação de Serviço da Web** > **Concluir**

4. Verifique o WSDL gerado.

Você solicitou que o WSDL específico do serviço fosse gerado e salvo no projeto.

a) No navegador Enterprise Explorer, abra a pasta **W3CJMSEJB** > **ejbmodule** > **META-INF** > **wsdl**. Clique duas vezes em StockQuoteAxis.wsdl para abri-lo no editor do WSDL.

Inspecione a ligação; você vê a URL do JMS:

```
jms:jndi:requestaxis?jndiConnectionFactoryName=qm1&targetService=StockQuoteAxis
```

5. Etapa opcional: ligue o EJB a SOAP sobre HTTP usando JAX-WS.

Fornecer duas ligações para o EJB fornece aos clientes a opção de ligações SOAP para chamar o serviço da web. Também fornece aos clientes o meio para consultar o servidor da web para obter seu WSDL, usando HTTP.

As etapas para ligar um EJB a SOAP sobre HTTP não estão incluídas como parte da tarefa.

6. Implemente e reimplemente StockQuoteAxis usando a amostra StockQuoteAxis.java

a) No navegador Enterprise Explorer, abra a pasta **W3CJMSEJB** > **Serviços** Clique duas vezes StockQuoteAxisService para abrir a classe de implementação em um editor Java.

b) Abra o programa de amostra StockQuoteAxis.java na pasta *WebSphere MQ Installation directory*\tools\soap\samples\java\server > Selecione todos os métodos, mas não o nome da classe> **Copiar**

c) Em StockQuoteAxisSoapBindingImpl.java, selecione todos os métodos, mas não o nome da classe, e cole os métodos a partir de StockQuoteAxis.java.

d) Inclua uma instrução de impressão na saída para o console do WebSphere Application Server quando o serviço for chamado
Mude o método getQuote(String symbol):

```
public float getQuote(String symbol) {
    System.out.println("StockQuoteAxisSoapBindingImpl called with symbol: "
                       + symbol);
    return ((float) 55.25);
```

e) Corrija as importações: **Origem** > **Organizar importações** > **Salvar**.

f) Corrija os três erros devido à implementação não corresponder à interface.

Os erros se devem a três dos métodos em StockQuoteAxis.java lançarem exceções e o WSDL para o serviço não conter quaisquer mensagens de falha. O problema é diagnosticado como sendo uma incompatibilidade entre as assinaturas do método e as anotações de serviço da web do método.

Anote os métodos com @WebFault e gere o WSDL novamente ou mantenha a interface inalterada e remova as exceções.

Para manter a interface igual, remova os três `throws exception` das assinaturas de método > **Salvar**.

Como proceder a seguir

“Implementando em um cliente Axis2 usando W3C SOAP sobre JMS” na página 1029

Tarefas relacionadas

Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse
Desenvolva um serviço da web Axis 1.4 para executar usando o WebSphere MQ como seu provedor de serviços. Use seu ambiente de desenvolvimento de serviço da Web normal para criar um serviço para implementação para o Axis 1.4

Desenvolvendo um serviço .NET 1 ou 2 para o transporte WebSphere MQ para SOAP usando o Microsoft Visual Studio 2008
Desenvolva o serviço da web de cotação SampleStockpara .NET 1 ou .NET 2 usando o Microsoft Visual Studio 2008

Desenvolvendo clientes de serviço da web do WebSphere MQ para transporte WebSphere MQ para SOAP

Use seu ambiente de desenvolvimento normal para desenvolver clientes de serviços da Web para uso com o transporte do WebSphere MQ para SOAP

Antes de começar

Crie o serviço. É possível usar um dos exemplos em “Desenvolvendo Serviços da web para o transporte WebSphere MQ para SOAP” na página 972.

Faça escolhas sobre como você vai desenvolver, implementar e usar os clientes e o local para obter o WSDL para a geração do cliente.

Decida sobre sua abordagem para desenvolver clientes e serviços para o transporte do WebSphere MQ para SOAP.

Há duas abordagens.

1. Use ferramentas de desenvolvimento padrão, desenvolva um serviço e cliente HTTP e, em seguida, use a URL para o transporte WebSphere MQ para SOAP.
2. Use as ferramentas e amostras fornecidas com o transporte WebSphere MQ para SOAP.

Se você tomar a rota HTTP, poderá executar o serviço em um servidor HTTP e também executá-la usando o transporte WebSphere MQ para SOAP.. Para executá-lo usando o transporte WebSphere MQ para SOAP, configure o listener WebSphere MQ apropriado para SOAP e configure os caminhos e descritores de implementação para executar o serviço. As ferramentas fornecidas pelo transporte do WebSphere MQ para SOAP fazem a configuração para você Como alternativa, é possível configurar o ambiente para executar os listeners.

As ferramentas fornecidas com o transporte WebSphere MQ para SOAP são úteis para iniciar e aprender como implementar o transporte. Para o trabalho de produção, há benefícios em usar as ferramentas padrão e implementar o mesmo serviço acessível para diferentes transportes SOAP.

Decida sobre o tipo de cliente para desenvolver

Deve-se decidir qual tipo de cliente de serviço da web desenvolver. A opção depende se você conhece a interface de serviço e o endereço do serviço.

Se a interface for conhecida, use as ferramentas Axis ou .NET para gerar classes de cliente de proxy a partir da interface de serviços As classes do cliente proxy tornam mais fácil gravar um cliente para chamar o serviço. Se o local do serviço for conhecido ao desenvolver o cliente, então use a interface de proxy estático. Se o local do serviço for mudado, por exemplo, se o serviço for reimplementado em um servidor de produção, use a interface de proxy dinâmico.

Se a interface de serviço não for conhecida no momento em que você desenvolver um cliente no Axis, será possível criar um cliente Dynamic Invocation Interface (DII) para o Axis 1.4. Um cliente DII usa uma interface genérica para chamar qualquer serviço. Para transmitir os parâmetros para um serviço específico corretamente, é necessário construir a interface com o serviço específico programaticamente. Construa a interface programaticamente no cliente ou ao carregar o WSDL para o serviço no cliente. No Axis2, é possível criar um cliente Dispatch. O cliente Dispatch usa um modelo de documento para descrever a solicitação do cliente, enquanto que um cliente DII usa um modelo de chamada. Ambos trabalham na construção da solicitação dinamicamente.

Obtenha o WSDL para o serviço

Exceto para o caso da interface de serviço que está sendo construída programaticamente, deve-se primeiro obter o WSDL de serviço para criar um cliente de serviço da web. O WSDL do serviço é obtido a partir de três origens diferentes:

1. Diretamente da implementação do serviço da Web usando uma ferramenta como **java2wsdl** (Axis) ou **disco** (.NET).
2. Consultando o serviço da web usando a URL: *Web service http url*?wsdl
3. A partir de um arquivo, em um sistema de arquivos ou de um registro, como UDDI ou WebSphere Service Registry and Repository.

Nota: Se o serviço não estiver acessível usando HTTP, então a consulta WSDL não funcionará. O próprio serviço pode estar disponível apenas usando o transporte WebSphere MQ para SOAP.

O WSDL gerado pelo **amqwdeployWMQService** não é o mesmo que WSDL gerado usando **java2wsdl** ou **disco**. O WSDL gerado também é diferente para quaisquer WSDL você pode ter iniciado para criar o serviço "Top Down". No Axis, o descritor de implementação server-config.wsdd mapeia a mensagem SOAP produzida por um cliente para uma operação e serviço. **amqwdeployWMQService** gera um descritor de implementação diferente do Eclipse.

O WSDL usado para construir clientes depende de como o serviço é implementado:

Implementados usando o amqwdeployWMQService
Use o WSDL gerado por **amqwdeployWMQService** Especifique o sinalizador -w e selecione o WSDL rpcLiteral. Para compatibilidade, é possível selecionar o WSDL rpcEncoded. rpcEncoded WSDL funciona apenas com clientes .NET e Axis 1.4 .

Implementação manual usando SimpleJavaListener
Use um dos seguintes arquivos WSDL:

1. WSDL usado para definir o serviço ou armazenado em um repositório.
2. WSDL gerado do serviço por **java2wsdl** .
3. WSDL consultado usando a URL *Web service http url* ?wsdl, se disponível de um servidor HTTP. Você pode executar uma ferramenta como o Web Services Explorer para importar a definição de serviço diretamente no Eclipse.

Pode ser necessário mudar o URI para o serviço. Altere-o do endereço do serviço HTTP para o URI para o transporte do WebSphere MQ para SOAP.

Implementação manual usando amqSOAPNETListener.
Use um dos seguintes arquivos WSDL:

1. WSDL usado para definir o serviço ou armazenado em um repositório.
2. WSDL obtido da classe de serviço .NET (.asmx). usando **disco**..
3. WSDL consultado usando a URL *Web service http url* ?wsdl, se disponível. Você pode executar uma ferramenta como o Web Services Explorer para importar a definição de serviço diretamente no Eclipse.
4. WSDL obtido executando **amqswsdl** com relação à classe de serviço .NET (.asmx).

Pode ser necessário mudar o URI para o serviço. Altere-o do endereço do serviço HTTP para o URI para o transporte do WebSphere MQ para SOAP.

Implementado no Windows Communication Foundation
Obtenha o WSDL de serviço usando a URL *Web service http url*?wsdl O serviço deve ser definido com a configuração de comportamento serviceMetaData como parte da definição de serviço.

Implementação para uma plataforma de servidor diferente.
Siga a orientação fornecida com a plataforma sobre como obter o serviço correto WSDL.

Sobre esta tarefa

Desenvolva clientes usando ferramentas de desenvolvimento padrão. As tarefas a seguir ilustram como construir clientes para .NET 1 e 2, Axis 1.4 (JAX-RPC) e Axis2 (JAX-WS). Para o Windows Communication Foundation, consulte os links de tarefa relacionados

Desenvolvendo um cliente JAX-RPC para o transporte WebSphere para SOAP usando Eclipse

Desenvolva um cliente de serviço da web Axis 1.4 para executar usando o transporte WebSphere MQ para SOAP.

Antes de começar

Deve-se ter o serviço disponível. Se você estiver seguindo a tarefa como um exercício prático, use a área de trabalho e o serviço criado na tarefa, "Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse" na página 973. É necessário ter um servidor de aplicativos em execução no eclipse que suporta serviços da web Axis 1.4. Nessa tarefa, usamos o WebSphere Application Server Community Edition Versão 2.1.4livremente disponível. Ele é configurado como parte da tarefa "Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse" na página 973 Você também pode usar o Tomcat 6, que é um servidor de aplicativos de software livre menor.

Sobre esta tarefa

A tarefa mostra o desenvolvimento de três tipos de cliente para o serviço de amostra StockQuoteAxis usando Eclipse em execução no Windows. Os clientes são um cliente estático e um dinâmico desenvolvidos usando o proxy de cliente e um cliente DII.

Duas abordagens alternativas para gerar os proxies de cliente a partir do WSDL são ilustradas:

1. Gerando proxies do cliente usando o **amqwdeployWMQService**
2. Importar WSDL para o Eclipse e usar o assistente de serviço da web para gerar os proxies de cliente.

Procedimento

1. Inicie o Eclipse IDE para desenvolvedores do Java EE .
2. Crie um projeto Java chamado StockQuoteAxisClient:
   a) Alterne para a perspectiva Java> **Arquivo** > **Novo** > **Projeto Java**. No campo **Project name** do tipo **Criar um Projeto Java** , StockQuoteAxisEclipseClient. Certifique-se de que o ambiente de execução seja **J2SE1-1.4** ou **J2SE-1.5** > **Avançar**
   b) Na página **Configurações Java**, selecione a guia **Bibliotecas** > **Incluir JARs externos...**
   c) Procure *MQ_INSTALLATION_PATH*/java/lib e selecione todos os arquivos .jar > **Abrir**.
   O *MQ_INSTALLATION_PATH* é o diretório no qual o WebSphere MQ está instalado.
   d) Procure *MQ_INSTALLATION_PATH*/java/lib/soap e selecione todos os arquivos .jar > **Abrir**. Você deve ter instalado axis.jar a partir da mídia de instalação do WebSphere MQ nesse diretório.
   O *MQ_INSTALLATION_PATH* é o diretório no qual o WebSphere MQ está instalado.
   e) A guia **Biblioteca** agora referencia todos os arquivos .jar necessários para construir o cliente> **Concluir**.

3. Siga uma dessas duas abordagens para criar proxies no Eclipse para o serviço da web StockQuoteAxis de amostra:

   - Gerar os proxies do cliente usando **amqwdeployWMQService** .

     a. Crie um gerenciador de filas. Para a tarefa, crie QM1 como o gerenciador de filas padrão.

     b. Crie um diretório de trabalho, `samples`. Copie o programa de amostra `StockQuoteAxis.java` para o `samples/soap/server`

     c. Modifique `amqwsetcp.cmd` em *MQ_INSTALLATION_PATH*`/bin` para incluir o diretório atual no caminho de classe.*MQ_INSTALLATION_PATH* é o diretório onde o WebSphere MQ está instalado.

     d. Abra uma janela de comando em `samples` e execute o comando **amqwsetcp** modificado

     e. Crie WSDL para o serviço StockQuoteAxis executando o comando,

     ```
     amqwdeployWMQService -f soap/server/StockQuoteAxis.java -c genAxisWsdl
     -u "jms:/queue?destination=REQUESTAXIS
     &initialContextFactory=com.ibm.mq.jms.Nojndi
     &connectionFactory=(connectQueueManager(QM1)binding(auto))"
     ```

     Não se esqueça: Use "`/`"em vez de "`.`" ou "`\`" ao usar comandos Java.

     Sugestão: Em vez de importar os proxies gerados no Eclipse, é possível pode importar o WSDL gerado a partir de `.samples/generated`. As proxies resultantes diferem de duas maneiras:

     i) Os nomes de pacotes são diferentes, o que é possível refatorar.

     ii) Os proxies gerados Eclipse incluem uma classe auxiliar adicional, `StockQuoteAxisProxy.java`

     f. Crie os proxies de cliente para o serviço StockQuoteAxis executando o comando:

     ```
     amqwdeployWMQService -f soap/server/StockQuoteAxis.java -c genProxiestoAxis
     -u "jms:/queue?destination=REQUESTAXIS
     &initialContextFactory=com.ibm.mq.jms.Nojndi
     &connectionFactory=(connectQueueManager(QM1)binding(auto))"
     ```

     g. Importe os proxies de cliente para StockQuoteAxisClient:

     i) Clique com o botão direito em **StockQuoteAxisClient\src** > Selecionar **Sistema de Arquivos** > **Avançar** > **Procurar ...** > localizar a pasta `.\samples\generated\client\remote\soap\server` > **OK**.

     ii) Verifique o **servidor** na página**Importar** > **Concluir**

     h. Refatore o nome do pacote para `soap.server`.

     i) Clique com o botão direito no pacote que contém os proxies de cliente > **Refatorar** > **Renomear**. Digite **New name**: `soap.server` > deixe os padrões selecionados para as outras opções> **OK** . Todos os erros são corrigidos.

   - Gere os proxies de cliente usando o Eclipse.

     Você tem uma opção de maneiras de obter o WSDL para o serviço. Neste exemplo, o serviço foi implementado no WebSphere Application Server Community Edition e você obtém o WSDL do servidor da Web. A implementação é descrita na tarefa "Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse" na página 973,

     a. No Eclipse, alterne para a perspectiva da web e verifique se o WebSphere Application Server Community Edition v2.1 Server está em execução e o StockQuoteAxis está implementado e sincronizado.

     b. Importe o WSDL para o Web Services Explorer:

     i) Clique no ícone **Web Services Explorer** na barra de ação ou clique em **Executar** > **Ativar o Web Services Explorer**.

     ii) Clique no ícone de página WSDL no Web Services Explorer para alternar para a página WSDL.

iii) Clique em **WSDL Principal** na janela do Navegador do Web Services Explorer.

iv) Digite a URL do serviço da web, seguida por ?WSDL A URL para o Eixo StockQuote, implementado na tarefa "Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse" na página 973, é:

```
http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl
```

c. Gere os proxies de cliente:

i) No navegador Web Services Explorer, clique em **http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl** .

ii) Na janela **Ações**, clique em **Ativar assistente de serviço da web** > deixe **Cliente de serviço da web** selecionado > **Ir**.

iii) Na primeira página do assistente, clique no link do projeto **Cliente** na configuração > Selecione o projeto do cliente **StockQuoteAxisClient** > **OK**.

Sugestão: A janela do assistente pode perder o foco. É necessário trazê-la de volta ao foco manualmente.

iv) O tempo de execução do serviço da web deve ser o Apache Axis para gerar um cliente JAX-RPC.

v) Clique em **Concluir**.

vi) Altere a URL estática do serviço para apontar para o transporte do WebSphere MQ para o endereço SOAP para o serviço do Eixo StockQuote Você pode optar por ignorar essa etapa, até que você tenha testado o cliente com um servidor HTTP.

a) Abra `StockQuoteAxisServiceLocator.java` e localize a declaração para `StockQuoteAxis_address`.

b) Mude a URL para

```
"jms:/queue?destination=REQUESTAXIS
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
&amp;connectionFactory=(connectQueueManager(QM1)binding(auto))"
```

Sugestão: O Eclipse transforma & automaticamente em `&`, e o inverso, quando você copia e cola sequências para o código `.java`.

d. Crie três classes de cliente Java, cada uma com um método principal:

i) Crie um pacote. Clique com o botão direito em **StockQuoteAxisClient/src** > **Novo pacote**. Dê a ele o nome `soap.client` > **Concluir**.

ii) Selecione **soap.client** > **Novo** > **Classe**. Dê à classe o nome `SQAStaticClient` > Marque **public static void main(string [] args)** > Concluir

iii) Repita o procedimento para criar `SQADynamicClient.java` e `SQADIIClient.java`

e. Escreva o código do cliente.

Figura 177 na página 992 a Figura 181 na página 994 fornecem exemplos dos três estilos de código do cliente. Os exemplos usam uma URL HTTP para testar o cliente usando o serviço StockQuoteAxis implementado em um servidor HTTP. Para executar os clientes no serviço do Eixo StockQuoteimplementado usando o transporte do WebSphere MQ para SOAP, altere a URL para:

```
"jms:/queue?destination=REQUESTAXIS
connectionFactory=(connectQueueManager(QM1)binding(auto))
initialContextFactory=com.ibm.mq.jms.Nojndi
targetService=soap.server.StockQuoteAxis.java
replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE"
```

– Figura 177 na página 992 e Figura 179 na página 993 usam o proxy gerado pelo Eclipse, que tem a classe auxiliar StockQuoteAxisproxy adicional que facilita um pouco a codificação.

– Figura 178 na página 993 e Figura 180 na página 993 usam o proxy gerado por **amqwdeployWMQService** .

– Figura 181 na página 994 não usa nenhuma classe de proxy.

Cada um dos clientes chama `com.ibm.mq.soap.Register.extension()` para vincular ao transporte WebSphere MQ para SOAP. A extensão é registrada no descritor de implementação do cliente. A implementação do cliente no Axis 1.4 é descrita em "Implementando um cliente de serviço da web para o Axis 1.4 para usar o transporte do IBM WebSphere MQ para SOAP" na página 1024.

f. Execute os clientes enviando a solicitação SOAP para StockQuoteAxis hospedado pelo servidor WebSphere Application Server Community Edition configurado na área de trabalho.

i) Verifique se o servidor está em execução, StockQuoteAxis está implementado e sincronizado.

ii) Selecione ou abra o cliente que deseja testar > Clique em **Executar** na barra de ação. Como alternativa, clique no ícone Executar verde ou clique no cliente no navegador > **Executar como** > **Executar configurações...**. Configure os parâmetros que você precisa para executar o cliente.

g. Execute o cliente usando o transporte WebSphere MQ para SOAP.

O procedimento usa **amqwdeployWMQService** para implementar o serviço e só funciona com o cliente que usa o WSDL ou proxies construídos pelo **amqwdeployWMQService** Para executar o cliente usando o WSDL original ou proxies construídos pelo Eclipse, implemente o serviço com seu descritor de implementação construído pelo Eclipse. Inicie manualmente o **SimpleJavaListener** usando o nome da porta de serviço como o `targetServiceName` .

i) Siga as instruções em "Implementando um serviço no Axis 1.4 para usar para o transporte WebSphere para SOAP usando amqwdeployWMQService" na página 1013 para implementar o serviço no listener SOAP Java simples do WebSphere MQ . A implementação de serviço funciona somente para o cliente que está usando os proxies de WSDL ou de cliente construídos por **amqwdeployWMQService**.

ii) Em uma janela de comando, execute **amqwclientconfig** para criar o arquivo do descritor de implementação do cliente, `client-deploy.wsdd`

iii) Importe `client-deploy.wsdd` na raiz do projeto Java que você deseja testar usando o transporte do WebSphere MQ para SOAP

a) Clique com o botão direito do mouse no projeto Java **StockQuoteAxisEclipseClient** > **Import** > **Sistema de Arquivo** > **Avançar** > **Procurar ...**

b) Navegue até o diretório que contém `client-deploy.wsdd` > **Abrir** > Selecione o diretório na página do assistente de **Importação** > verifique `client-deploy.wsdd` na área de janela direita

c) Verifique se **Na pasta:** tem `StockQuoteAxisEclipseClient` inserido > **Concluir**.

iv) Confirme se o diretório ativo para executar um aplicativo Java neste projeto é o diretório `StockQuoteAxisEclipseClient` :

Clique com o botão direito do mouse no projeto Java **StockQuoteAxisEclipseClient** > **Executar como ....** > **Configurações de Execução ...** > Selecione a guia **(x) = Argumentos** > Verifique se no Diretório ativo o botão de opções **Padrão** está marcado e o caminho é `StockQuoteAxisEclipseClient` . Como alternativa, faça uma das escolhas a seguir para selecionar um local diferente ou arquivo que contém a configuração do cliente:

– Marque **Outro:** > digite um caminho de diretório de sua preferência.

– Na janela **Argumentos da VM** , digite `-Daxis.ClientConfigFile=`*`full path to client deployment descriptor file`*

v) Certifique-se de que a URL esteja configurada para apontar para o serviço implementado usando o transporte WebSphere MQ para SOAP. Execute o cliente conforme descrito na etapa ii.

Sugestão: Geralmente, você pode encontrar um desses erros:

i) `Exception: No client transport named 'jms' found!.`

ii) Um erro de conexão JMS.

iii) `Exception: The AXIS engine could not find a target service to invoke! targetService is soap.server.StockQuoteAxis.java`

iv) `Exception: java.lang.InstantiationException: soap.server.StockQuoteAxis`

Explicações:

i) `client-config.wsdd` não foi localizado ou não inclui a linha `<transport name="jms" pivot="java:com.ibm.mq.soap.transport.jms.WMQSender"/>` em `client-config.wsdd`

ii) Possivelmente um problema de caminho de construção-não incluindo os arquivos .jar em *MQ_INSTALLATION_PATH*`/java/lib` O *MQ_INSTALLATION_PATH* é o diretório no qual o WebSphere MQ está instalado.

iii) Problema de implementação de serviço, com `server-config-wsdd` ou com os parâmetros transmitidos para **SimpleSoapListener** .

iv) Incompatibilidade entre o descritor de implementação e a implementação do serviço.

Se você estiver tendo dificuldade para executar o cliente no Eclipse, tente usar uma janela de comando:

i) Alterne para o diretório `StockQuoteAxisEclipseClient\bin` na árvore de diretórios da área de trabalho.

ii) Execute **amqwsetcp** e **amqwclientconfig**

iii) Execute `java soap/client/SQAStaticClient`.

Clientes de serviço da web JAX-RPC de amostra

Os clientes de serviço da Web Java de amostra enviados com o WebSphere MQ são instalados no *MQ_INSTALLATION_PATH* `\tools\soap\samples\java\clients`. *MQ_INSTALLATION_PATH* .. é o diretório onde o WebSphere MQ está instalado.

SQAxis2Axis.java
SQAxis2Axis.java, Figura 174 na página 991, é um cliente proxy dinâmico para chamar o serviço do StockQuoteAxis. É possível substituir a URL do serviço, compilado no proxy dinâmico, fornecendo uma URL na linha de comandos.

SQAxis2DotNet.java
SQAxis2DotNet.java, Figura 175 na página 991, é um cliente proxy dinâmico para chamar o serviço StockQuoteDotNet. É possível substituir a URL do serviço, compilado no proxy dinâmico, fornecendo uma URL na linha de comandos.

WsdlClient.java
WsdlClient.java, Figura 176 na página 992, é um cliente de chamada dinâmica para chamar o serviço StockQuoteDotNet ou StockQuoteAxis .. O cliente chama o serviço StockQuoteAxis por padrão. Inclua a opção da linha de comandos -D chamar o serviço StockQuoteDotNet e -w para fornecer uma porta diferente daquela no `.\generated\StockQuoteDotNet_Wmq.wsdl`

```
package soap.clients;
import java.net.URL;
import soap.server.*;
public class SQAxis2Axis {
   public static void main(String[] args) {
      com.ibm.mq.soap.Register.extension();
      try {
         StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
         StockQuoteAxis service = null;
         if (args.length == 0)
            service = locator.getSoapServerStockQuoteAxis_Wmq();
         else
            service = locator.getSoapServerStockQuoteAxis_Wmq(
                new java.net.URL(args[0]));
         System.out.println("Response: " + service.getQuote("XXX"));
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING ProxyClient DEMO <<<\n");
                           e.printStackTrace();
         System.exit(2);
      }
   }
```

Figura 174. SQAxis2Axis.java

```
public class SQAxis2DotNet {
public static void main(String[] args) {
   com.ibm.mq.soap.Register.extension();
      try {
         StockQuoteDotNet locator = new StockQuoteDotNetLocator();
         StockQuoteDotNetSoap_PortType service = null;
         if (args.length == 0)
            service = locator.getStockQuoteDotNetSoap();
         else
            service = locator.getStockQuoteDotNetSoap(new java.net.URL(
                  args[0]));
         System.out.println("Response: " + service.getQuoteDOC("XXX"));
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING ProxyClient DEMO <<<\n");
                           e.printStackTrace();
         System.exit(2);
      }
   }
}
```

Figura 175. SQAxis2DotNet.java

```
package soap.clients;
import com.ibm.mq.soap.*;
import org.apache.axis.utils.Options;
import java.net.URL;
import javax.xml.rpc.Call;
import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceFactory;
import javax.xml.namespace.QName;
public class WsdlClient {
public static void main(String[] args) {
      String wsdlService, wsdlPort, namespace, wsdlSource, wsdlTargetURI, s;
      try {
         Register.extension();
         Options opts = new Options(args);
         if (opts.isFlagSet('D') != 0) {
            wsdlService = "StockQuoteDotNet";
            wsdlPort = "StockQuoteDotNetSoap";
            namespace = "http://stock.samples";
            wsdlSource = "file:generated/StockQuoteDotNet_Wmq.wsdl";
         } else {
            wsdlService = "StockQuoteAxisService";
            wsdlPort = "soap.server.StockQuoteAxis_Wmq";
            namespace = "soap.server.StockQuoteAxis_Wmq";
            wsdlSource = "file:generated/soap.server.StockQuoteAxis_Wmq.wsdl";
         }
         if (null != (s = (opts.isValueSet('w'))))
            wsdlPort = s;
         System.out.println("start WsdlClient demo, wsdl port " + wsdlPort
               + " resolving uri to ...");
         QName servQN = new QName(namespace, wsdlService);
         QName portQN = new QName(namespace, wsdlPort);
         Service service = ServiceFactory.newInstance().createService(
               new URL(wsdlSource), servQN);
         Call call = (Call) service.createCall(portQN, "getQuote");
         wsdlTargetURI = call.getTargetEndpointAddress().toString();
         System.out.println("  '" + wsdlTargetURI + "'");
         Object ret = call.invoke(new Object[] { "XXX" });
         System.out.println("Response: " + ret);
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING WsdlClient DEMO <<<\n");
         e.printStackTrace();
         System.exit(2);
      }
   }
}
```

Figura 176. WsdlClient.java

Os clientes de exemplo usados nesta tarefa:

```
package soap.client;
import soap.server.StockQuoteAxisProxy;
public class SQAStaticClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisProxy sqa = new StockQuoteAxisProxy();
      System.out.println("Static client synchronous result is:"
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Figura 177. Cliente estático usando proxy gerado pelo Eclipse

```
package soap.client;
import soap.server.StockQuoteAxis;
import soap.server.StockQuoteAxisService;
import soap.server.StockQuoteAxisServiceLocator;
public class SQAStaticClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
      StockQuoteAxis sqa = locator.getSoapServerStockQuoteAxis_Wmq();
      System.out.println("Static client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Figura 178. Cliente estático usando o proxy gerado por amqwdeployWMQService

```
package soap.client;
import soap.server.StockQuoteAxisProxy;
public class SQADynamicClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisProxy sqa = new StockQuoteAxisProxy(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      System.out.println("Dynamic client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Figura 179. Cliente dinâmico usando proxy gerado pelo Eclipse

```
package soap.client;

import java.net.URL;
import soap.server.StockQuoteAxis;
import soap.server.StockQuoteAxisService;
import soap.server.StockQuoteAxisServiceLocator;
public class SQADynamicClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      URL sqaURL = new URL(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
      StockQuoteAxis sqa = locator.getSoapServerStockQuoteAxis_Wmq(sqaURL);
      System.out.println("Dynamic client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Figura 180. Cliente dinâmico usando o proxy gerado por amqwdeployWMQService

```
package soap.client;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.rpc.Call;
import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceFactory;
public class SQADIIClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      URL wsdl = new URL(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl");
      Service SQAService = (ServiceFactory.newInstance()).createService(wsdl,
          new QName("http://server.soap", "StockQuoteAxisService"));
      Call SQACall = SQAService.createCall(new QName("http://server.soap",
          "StockQuoteAxis"), "getQuote");
      System.out.println("DII client synchronous result is "
          + SQACall.invoke(new Object[] { "ibm" }));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Figura 181. Cliente DII (nenhum proxy)

Tarefas relacionadas

Desenvolvendo um Cliente JAX-WS para WebSphere Transport for SOAP Usando Eclipse
Desenvolva um cliente de serviço da Web Axis2 para executar usando o WebSphere MQ Transport for SOAP. Os clientes Axis2 de amostra fornecidos com o WebSphere MQ Transport for SOAP são listados e o comando **wsimport** é usado para gerar proxies.

Desenvolvendo um cliente .NET 1 ou 2 para o transporte do WebSphere para SOAP usando o Microsoft Visual Studio 2008
Desenvolva um cliente de serviço da Web .NET 1 ou 2 para executar usando o transporte WebSphere MQ para SOAP.

Desenvolvendo um Cliente JAX-WS para WebSphere Transport for SOAP Usando Eclipse

Desenvolva um cliente de serviço da Web Axis2 para executar usando o WebSphere MQ Transport for SOAP. Os clientes Axis2 de amostra fornecidos com o WebSphere MQ Transport for SOAP são listados e o comando **wsimport** é usado para gerar proxies.

Antes de começar

Obtenha as bibliotecas Axis2 e configure um ambiente de desenvolvimento e teste para executar o cliente.

Nota: A nomenclatura de versões e liberações usada pelo Axis causa confusão. Normalmente o Axis 1.4 refere-se à implementação JAX-RPC, e o Axis2 à implementação JAX-WS.

Axis 1.4 é um nível de versão. Se você procurar por Axis 1.4 na Internet, será levado para http://ws.apache.org/axis/. A página contém uma lista de versões precedentes do Axis (1.2, 1.3) e a liberação final de 22 de abril de 2006 do Axis 1.4. Há liberações mais recentes do Axis 1.4 que corrigem erros, mas todas elas são conhecidas como Axis 1.4. É uma dessas liberações de correção de erro que é fornecida com o WebSphere MQ. Para Axis 1.4, use a versão de `axis.jar` que é fornecida com o WebSphere MQ no lugar da que pode ser obtida em http://ws.apache.org/axis/.

O Web site do Axis também encaminha para o Axis 1.1, para encaminhar para todas as versões daquele que, normalmente, é chamado de Axis 1.4. O Axis 1.2 é usado para encaminhar para aquele que normalmente é chamado de Axis2.

Axis 1.5 não é uma liberação mais recente do Axis 1.4, é uma liberação do Axis2. Se procurar o Axis 1.5, você será direcionado para http://ws.apache.org/axis2/.https://ws.apache.org/axis2/download.cgi contém uma lista de versões de liberação de Axis2, rotuladas como 0,9 1.5.1 (e incluindo, de maneira confusa, a version 1.4). A versão de liberação do Axis2 para usar com o WebSphere MQ Transport

for SOAP é 1.4.1. Faça o download do Axis2 1.4.1 em http://ws.apache.org/axis2/download/1_4_1/download.cgi.

É possível optar por gerar proxies para os clientes de serviço da para WebSphere MQ Transport for SOAP usando **wsimport** ou o conjunto de ferramentas fornecido com um IDE. Eclipse IDE para Java EE Developer 3.5 SR1 usa **wsdl2java**.**wsimport** é fornecido com Java 6. É possível usar o Java 5 para executar proxies do cliente gerados com **wsimport** ou **wsdl2java**

Os clientes Axis2 de serviço da Web de amostra fornecidos com o WebSphere MQ Transport for SOAP foram desenvolvidos usando **wsimport**; consulte "Clientes Axis2 de amostra" na página 1000.

A tarefa a seguir demonstra como gerar e usar os proxies produzidos pelo assistente de serviços da web que é empacotado com o Eclipse IDE para Java EE Developers Os clientes de amostra mostram como usar os proxies produzidos pelo **wsimport**.

Para usar o assistente de serviços da Web, deve-se incluir um servidor de aplicativos que suporte o Axis2 no ambiente de trabalho. As etapas mostram como configurar o WASCE para suportar Axis2 usando o ambiente de trabalho.

1. Configure o servidor de aplicativos usado no Eclipse IDE para Java EE Developers para suportar Axis2. Neste exemplo, configure o servidor de aplicativos WASCE 2.1.4, que faz parte da área de trabalho criada em "Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse" na página 973.
   a. Abra as preferências da área de trabalho para configurar o servidor: abra **Janela** > **Preferências**.
   b. Verifique se o JRE instalado é o `Java50`: Clique em **JREs instalados**.
   c. Inclua WASCE como o servidor: Clique em **Servidor** > **Ambientes de tempo de execução** > **Incluir ...** > **IBM** > **WASCE v2.1** > Avançar. O JRE deve ser Java50 > Procurar para o diretório de instalação do WASCE> **OK** > **Finish**. Você deve ter instalado o plug-in WASCE para o IDE Eclipse Java EE para Desenvolvedores da Web.
   d. Inclua Axis2: clique em **Serviços da web** > **Preferências de Axis2**. Na guia **Axis2 Tempo de execução** > **Procurar ...** Abra o diretório que contém muitos arquivos jar `Axis2` > **Aplicar**.
   e. Associe WASCE ao Axis2: Clique em **Serviços da Web** > **Servidor e Tempo de Execução**. Em **Server** selecione **IBM WASCE v2.1 Server**e em **Web service runtime**, selecione **Apache Axis2** >> **Aplicar** > **OK**
   f. Inicie o servidor: Abra a perspectiva da Web e abra a visualização Servidores. Clique com o botão direito na visualização Servidores > **Novo** > **Servidor**. **IBM WASCE v2.1 Server** é selecionado e configurado> **Finish**. Inicialize o servidor.
2. Verifique se você implementou o serviço StockQuoteAxis no WASCE para executar o assistente de serviços da Web.
3. Para testar o serviço com o serviço WebSphere MQ Transport for SOAP, implemente o serviço em um listener do WebSphere MQ Transport for SOAP para Axis 1.4; consulte "Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse" na página 973.

Sobre esta tarefa

O Eclipse IDE para Desenvolvedores Java EE usa Java50 e o assistente de serviços da Web para gerar as classes de proxy para o serviço As classes de proxy são diferentes das classes criadas pela ferramenta **wsimport** fornecida com Java 6. Uma abordagem alternativa é gerar as classes de proxy usando **wsimport** e importar os pacotes que ele cria para seu Eclipse Java EE IDE for Web Developers.

O assistente de serviços da web no Eclipse IDE para Desenvolvedores Java EE constrói um cliente de serviço da Web em um projeto da web É possível executar o cliente como um aplicativo Java simples; ele não requer um servidor de aplicativos.. Também é possível transferir o código para um projeto Java e configurar o caminho de construção para incluir os arquivos JAR Axis2 .

Procedimento

1. Crie um projeto da Web em novo projeto Corporativo:

a) Com nada selecionado no Explorador de Projetos > Clique com o botão direito no espaço em branco > **Novo** > **Projeto do aplicativo corporativo** > Dê a ele o nome `StockQuoteAxis2EAR` > **Concluir**. Responda No para a janela fornecendo a opção de abrir a perspectiva Java EE .

Os padrões são configurados para usar WASCE.

b) Clique com o botão direito em `StockQuoteAxis2EAR` > **Novo** > **Projeto dinâmico da web**. Dê ao projeto o nome `StockQuoteAxis2WebClient` > Marque a caixa de associação EAR para incluir o projeto em **StockQuoteAxis2EAR**. O WASCE 2.1 é selecionado como tempo de execução de destino.

c) Na seção Configuração da página **Novo Projeto Dinâmico da Web** > **Modificar ...** > Verifique o aspecto do projeto de serviço da Web Axis2 . **Módulo da Web Dinâmico 2.5**, **Java 6.0**e **Implementação do WASCE 1.2** já estão verificados > **OK** > **Concluir** Responda No para a janela fornecendo a opção de abrir a perspectiva Java EE .

2. Importe o WSDL para o serviço na área de trabalho e gere o proxy de cliente:

Neste exemplo, o documento WSDL contém a ligação de serviços HTTP e se torna o destino para o proxy de cliente da Web estático. É possível modificar a URL na ligação de serviços da Web para apontar para a URL do WebSphere MQ Transport for SOAP antes da geração do proxy de cliente. Então, o proxy de cliente da Web estático será o serviço que é implementado no WebSphere MQ Transport for SOAP.

a) Ative o Web Services Explorer: use o ícone na barra de ação ou **Executar** > **Ativar o Web Services Explorer**.

b) Selecione o explorador WSDL clicando no ícone WSDL na janela **Web Services Explorer** > Clique em **WSDL principal** na janela Navegador > Digite a URL do arquivo WSDL StockQuoteAxis > **Ir**. Neste exemplo, obtenha o WSDL diretamente do serviço HTTP: `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl`

c) No Navegador, clique na linha com a URL do serviço da Web. Na janela **Ações**, clique em **Importar WSDL para o ambiente de trabalho** > Selecione um **StockQuoteAxis2WebClient** como o **Projeto de ambiente de trabalho** > Digite o **Nome do arquivo WSDL**, `StockQuoteAxisHTTP.wsdl` > **Ir**.

d) Clique com o botão direito em **StockQuoteAxisHTTP.wsdl** > **Serviços da web** > **Gerar cliente**. Verifique se as informações de configuração sobre a página de serviços da Web do assistente são as seguintes: Servidor: IBM WASCE v2.1 Server, Tempo de execução do serviço da Web: Apache Axis2, Projeto do cliente: StockQuoteAxis2WebClient, Projeto de EAR do cliente: StockQuoteAxisEAR. Para corrigir a configuração, clique nas linhas que estão erradas.

e) Clique em **Avançar** > verifique as configurações de geração de código > **Concluir**.

Observe que um novo pacote, `soap.server`, é criado e ele contém os proxies requeridos.

3. Configure o projeto para executar o WebSphere MQ Transport for SOAP como transporte JMS.

O WebSphere MQ Transport for SOAP fornece um `transportSender`, mas não um `transportReceiver`. Em outras palavras, o WebSphere MQ Transport for SOAP suporta clientes Axis2. Atualmente, ele não suporta serviços do Axis2.

a) No projeto **StockQuoteAxis2WebClient** , clique com o botão direito em `WebContent\WEB-INF\conf\axis2.xml` > **Abrir com ...** > **Editor XML**.

b) Procure o último `transportSender` (no final do arquivo) e localize o JMS comentado `transportSender` > Clique com o botão direito na linha> **Incluir antes ...** > **transportSender**.

c) Clique com o botão direito em **transportSender** > **Incluir atributo** > **Nome** > Clique com o botão direito em **transportSender** > **Incluir atributo** > **Classe**.

d) Clique com o botão direito em **Nome** > **Editar atributo** > Digite o **Valor:** `jms`

e) Clique com o botão direito em **Classe** > **Editar Atributo** > Digite o **Valor:** `com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender`.> Salvar.

f) Inclua `com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender` no caminho de construção: Clique com o botão direito em **StockQuoteAxis2WebClient** > **Caminho de Construção** > **Configurar Caminho de Construção ...** > Clique na guia **Bibliotecas** > **Incluir JARs Externos ...**. Selecione todos os JARs em *MQ_INSTALLATION_PATH*`\java\lib` > **OK**.

O *MQ_INSTALLATION_PATH* é o diretório no qual o WebSphere MQ está instalado.

4. Crie um cliente estático síncrono, teste-o usando HTTP e, depois, converta o proxy para executar o cliente estático usando o WebSphere MQ Transport for SOAP.
   a) Clique com o botão direito em **Recursos Java: src** > **Novo** > **Pacote** > Nomear o pacote `soap.client` > Concluir
   b) Clique com o botão direito em **soap.client** > **Nova** > **Classe** > Dê à classe o nome `SQA2StaticClient` > **Concluir**.
   c) Substitua a classe pelo código a seguir, em seguida, clique em **Salvar**.

   Figura 182. `SQA2DynamicClient.java`

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
public class SQA2StaticClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub();
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("Response is: "
          + (stub.getQuote(request)).getGetQuoteReturn());
    } catch (Exception e) {
      System.out.println("Exception: " + e.getMessage());
      e.printStackTrace();
    }
  }
}
```

5. Teste o cliente com o serviço StockQuoteAxis implementado no WASCE e com o WebSphere MQ Transport for SOAP.
   a) No Explorador de Projetos, clique com o botão direito em **SQA2StaticClient** > **Executar como ...** > **Aplicativo Java**.

      O resultado, `Response is 55.25`, aparece na visualização de Console Também é possível selecionar a janela do console WASCE na visualização Console e ver a saída no servidor WASCE, `StockQuoteAxis called with parameter: ibm`.
   b) O proxy foi construído com o endereço de serviço, `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis`, e, portanto, o cliente estático chama o serviço em execução no HTTP. É possível alterar o cliente estático para chamar o serviço usando WebSphere MQ Transport for SOAP. As instruções a seguir mudam o endereço de serviço em `StockQuoteAxisServiceStub.java` sem reconstruir o proxy e configuram os parâmetros de tempo de execução `SQA2StaticClient` para carregar `axis2.xml`. Você configura `axis2.xml` para configurar o Axis2 para usar o WebSphere MQ Transport for SOAP.
   c) Abra `StockQuoteAxisServiceStub.java` >
      Substitua as duas ocorrências de `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis` por,

```
jms:/queue?destination=REQUESTAXIS@QM1
&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
&targetService=StockQuoteAxis
&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
```

   d) Se você executar `SQA2StaticClient` agora, ele emitirá uma exceção porque não localizou um `transportSender` configurado para JMS
      A exceção é:

```
Exception: null java.lang.NullPointerException at
soap.server.StockQuoteAxisServiceStub.getQuote(StockQuoteAxisServiceStub.java:547)
at soap.client.SQA2StaticClient.main(SQA2StaticClient.java:11)
```

e) No Explorador de Projetos, clique com o botão direito em **SQA2StaticClient** > **Executar como ...** > **Configurações de Execução ...**. Alterne para a guia **(x) = Argumentos** e, na área de entrada **Argumentos da VM** , digite o caminho para o arquivo `axis2.conf` > **Aplicar** > **Executar**. O argumento da VM é: `-Daxis2.xml=${workspace_loc:StockQuoteAxis2WebClient/WebContent/WEB-INF/conf}/axis2.xml`. Ou é possível fornecer um caminho padrão para o arquivo de configuração do Axis2.

f) Execute `SQA2StaticClient` novamente. Nessa execução, você está usando o WebSphere MQ Transport for SOAP. Confirme isso verificando se não há nenhuma nova saída no console do WASCE. Abra a janela de console ou de comando associada ao Listener SimpleJavae a saída será `StockQuoteAxis called with parameter: ibm`.

6. Crie um cliente dinâmico para HTTP e WebSphere MQ Transport for SOAP teste-o.

a) Clique com o botão direito em **soap.client** > **Nova** > **Classe** > Dê à classe o nome `SQA2DynamicClient` > **Concluir**.

b) Substitua a classe pelo código a seguir, em seguida, clique em **Salvar**.

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
public class SQA2DynamicClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("HTTP Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      stub = new StockQuoteAxisServiceStub(
          "jms:/queue?destination=REQUESTAXIS@QM1"
          + "&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&targetService=StockQuoteAxis&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE");
       System.out.println("JMS sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
     } catch (Exception e) {
       System.out.println("Exception: " + e.getMessage());
       e.printStackTrace();
     }
   }
 }
```

c) Crie uma configuração Executar para `SQA2DynamicClient.java` e inclua o caminho em `axis2.xml`: `-Daxis2.xml=${workspace_loc:StockQuoteAxis2WebClient/WebContent/WEB-INF/conf}/axis2.xml`

d) Execute `SQA2DynamicClient`. Verifique a saída de console para `SQA2DynamicClient`, WASCE e **`SimpleJavaListener`**.

7. Crie um cliente assíncrono e acesse o resultado em um manipulador de retorno de chamada e no encadeamento do programa principal.

Os proxies de cliente assíncronos criados pelo assistente de serviço da web do Eclipse Java EE IDE for Web Developers diferem dos proxies criados pelo **`wsimport`**. Os tipos genéricos **`wsimport`** proxies usam `Future`, `Responsee` `AsyncHandler` .

O assistente de serviço da web para o IDE Eclipse Java EE para Desenvolvedores da web cria uma classe abstrata `StockQuoteAxisServiceCallbackHandler` Deve-se estender `StockQuoteAxisServiceCallbackHandler` e criar um manipulador de retorno de chamada.

a) Clique com o botão direito em **soap.client** > **Nova** > **Classe** > Dê à classe o nome `SQA2CallbackHandler` > **Concluir**.

b) Substitua a classe pelo seguinte código.

```
package soap.client;
import soap.server.StockQuoteAxisServiceCallbackHandler;
import soap.server.StockQuoteAxisServiceStub.GetQuoteResponse;
public class SQA2CallbackHandler
    extends StockQuoteAxisServiceCallbackHandler {
```

```
  private boolean complete = false;
  SQA2CallbackHandler() {
    super();
    System.out.println("Callback constructor");
  }
  public void receiveResultgetQuote(GetQuoteResponse response) {
    System.out.println("Result in Callback " + response.getGetQuoteReturn());
    super.clientData = response;
    complete = true;
  }
  public boolean isComplete() {
    return complete;
  }
}
```

c) Clique com o botão direito em **soap.client** > **Nova** > **Classe** > Dê à classe o nome SQA2AsyncClient > **Concluir**.

d) Substitua a classe pelo seguinte código.

Figura 183. SQA2AsyncClient.java

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
import soap.server.StockQuoteAxisServiceStub.GetQuoteResponse;
import soap.server.StockQuoteAxisServiceCallbackHandler;
@SuppressWarnings("unused")
public class SQA2AsyncClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("HTTP Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      SQA2CallbackHandler callback = new SQA2CallbackHandler();
      stub.startgetQuote(request, callback);
      do {
        System.out.println("Waiting for HTTP callback");
        Thread.sleep(2000);
      } while (!callback.isComplete());
      System.out.println("HTTP poll: "
          + ((GetQuoteResponse) (callback.getClientData()))
                .getGetQuoteReturn());
      stub = new StockQuoteAxisServiceStub(
          "jms:/queue?destination=REQUESTAXIS@QM1"
          + "&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&targetService=StockQuoteAxis&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE");
      System.out.println("JMS Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      callback = new SQA2CallbackHandler();
      stub.startgetQuote(request, callback);
      while (!callback.isComplete()) {
        System.out.println("Waiting for JMS callback");
        Thread.sleep(2000);
      }
      System.out.println("JMS poll: "
        + ((GetQuoteResponse) (callback.getClientData())).getGetQuoteReturn());
    } catch (Exception e) {
        System.out.println("Exception: " + e.getMessage());
        e.printStackTrace();
     }
   }
 }
```

A saída do console é a seguinte:

```
HTTP Sync: 55.25
Callback constructor
Waiting for HTTP callback
Result in Callback 55.25
HTTP poll: 55.25
JMS Sync: 55.25
Callback constructor
Waiting for JMS callback
```

```
Result in Callback 55.25
JMS poll: 55.25
```

Clientes Axis2 de amostra

Os proxies de amostra são gerados usando a ferramenta **wsimport** que é compactada com Java 6. Seis amostras são fornecidas:

1. DynamicProxyClientSync.java
2. DynamicProxyClientAsyncPolling.java
3. DynamicProxyClientAsyncCallback.java
4. DispatchClientSync.java
5. DispatchClientAsyncPolling.java
6. DispatchClientAsyncCallback.java

As amostras do cliente são geradas para o servidor StockQuoteAxis de amostra. Gere o WSDL com o comando **amqwdepoyWMQServer**, especificando o comutador -w para selecionar o estilo `rpcLiteral`. Use o comando a seguir para gerar os proxies para as amostras:

```
wsimport soap.server.StockQuoteAxis_Wmq.wsdl -d generated -keep -p com.ibm.mq.axis2.samples
```

Figura 184. DynamicProxyClientSync.java

```
package com.ibm.mq.axis2.samples;

import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientSync {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientSync");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();

      System.out.println("Invoking getQuoteOneWay OneWay operation synchronously...");
      service.getQuoteOneWay("48");
      System.out.println(" > getQuoteOneWay has returned");

      System.out.println("Invoking getQuote Request Reply operation synchronously...");
      float result = service.getQuote("48");
      System.out.println(" > getQuote has returned result of " + result);

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Figura 185. DynamicProxyClientAsyncPolling.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.CancellationException;

import javax.xml.ws.Response;

import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientAsyncPolling {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientAsyncPolling");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();

      System.out
          .println("Invoking getQuoteAsync Request Reply operation asynchronously by
polling...");
      Response<Float> response = service.getQuoteAsync("49");

      /** Sleep main thread until response arrives **/
      System.out.println("Waiting for response to arrive...");
      while (!response.isDone()) {
        Thread.sleep(100);
      }
      System.out.println(" > Response received");

      /** Retrieve the result **/
      try {
        Float result = response.get();
        System.out.println(" > getQuoteAsync call has returned result of " + result);
      }
      catch (CancellationException ce) {
        // processing was cancelled via response.cancel()
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Figura 186. DynamicProxyClientAsyncCallback.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.Future;

import javax.xml.ws.AsyncHandler;
import javax.xml.ws.Response;
```

```
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientAsyncCallback implements AsyncHandler<Float> {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientAsyncCallback");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();

      DynamicProxyClientAsyncCallback handler = new DynamicProxyClientAsyncCallback();

      System.out
          .println("Invoking getQuoteAsync Request Reply operation asynchronously using a
callback...");
      Future<?> monitor = service.getQuoteAsync("50", handler);
      System.out.println(" > Invoke call has returned");

      /** Sleep main thread until handler has been notified **/
      System.out.println("Waiting for handler to be called...");
      while (!monitor.isDone()) {
        Thread.sleep(100);
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

  public void handleResponse(Response<Float> response) {
    try {
      Float result = response.get();
      System.out.println(" > Async Handler has received a result of " + result);
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println("Exception in handleResponce");
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

}
```

Figura 187. DispatchClientSync.java

```
package com.ibm.mq.axis2.samples;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Service;
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientSync {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientSync");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service **/
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /************************************
       * Create OneWay SOAPMessage request.
       ************************************/
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("\nCreating a OneWay SOAP Message");
      SOAPMessage request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();

      /** Construct the message payload **/
      SOAPElement operation = body.addChildElement("getQuoteOneWay", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuoteOneWay OneWay operation synchronously...");
      dispatch.invokeOneWay(request);
      System.out.println(" > getQuoteOneWay call has returned");

      /****************************************
       * Create Request Reply SOAPMessage request.
       ****************************************/
      mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("\nCreating a Request Reply SOAP Message");
      request = mf.createMessage();
```

```
      /** Obtain the SOAPEnvelope and header and body elements **/
      part = request.getSOAPPart();
      env = part.getEnvelope();
      header = env.getHeader();
      body = env.getBody();

      /** Construct the message payload **/
      operation = body.addChildElement("getQuote", "ns1", "soap.server.StockQuoteAxis_Wmq");
      value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuote Request Reply operation synchronously...");
      SOAPMessage ans = dispatch.invoke(request);
      System.out.println(" > getQuote call has returned");

      /** Retrieve the result **/
      part = ans.getSOAPPart();
      env = part.getEnvelope();
      body = env.getBody();

      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      System.out.println("Parsing SOAP response...");
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String message = responseElement.getValue();
      System.out.println(" > Response contains result of " + message);

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Figura 188. DispatchClientAsyncPolling.java

```
package com.ibm.mq.axis2.samples;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Response;
import javax.xml.ws.Service;
```

```
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientAsyncPolling {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientAsyncPolling");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service.* */
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /** Create SOAPMessage request. * */
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("Creating a Request Reply SOAP Message");
      SOAPMessage request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();

      /** Construct the message payload **/
      SOAPElement operation = body.addChildElement("getQuote", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuote Request Reply operation asynchronously by
polling...");
      Response<SOAPMessage> response = dispatch.invokeAsync(request);
      System.out.println(" > getQuote call has returned");

      /** Sleep main thread until response arrives **/
      System.out.println("Waiting for response to arrive...");
      while (!response.isDone()) {
        Thread.sleep(100);
      }
      System.out.println(" > Response received");

      /** retrieve the result **/
      SOAPMessage ans = response.get();
      part = ans.getSOAPPart();
      env = part.getEnvelope();
      body = env.getBody();

      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String message = responseElement.getValue();
      System.out.println(" > Response contains result of " + message);

      System.out.println("End of sample");

    }
    catch (Exception fault) {
```

```
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Figura 189. DispatchClientAsyncCallback.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.Future;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.AsyncHandler;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Response;
import javax.xml.ws.Service;
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientAsyncCallback implements AsyncHandler<SOAPMessage> {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientAsyncCallback");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service.* */
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /** Create SOAPMessage request. * */
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("Creating a Request Reply SOAP Message");
      SOAPMessage request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();
```

```
      /** Construct the message payload. **/
      SOAPElement operation = body.addChildElement("getQuote", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint. **/
      DispatchClientAsyncCallback handler = new DispatchClientAsyncCallback();

      System.out
          .println("Invoking getQuote Request Reply operation asynchronously using a
callback...");
      Future<?> monitor = dispatch.invokeAsync(request, handler);
      System.out.println(" > getQuote call has returned");

      /** Sleep main thread until handler has been notified **/
      System.out.println("Waiting for handler to be called...");
      while (!monitor.isDone()) {
        Thread.sleep(100);
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

  public void handleResponse(Response<SOAPMessage> response) {
    try {
      // retrieve the result
      SOAPMessage ans = response.get();
      SOAPPart part = ans.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPBody body = env.getBody();

      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String result = responseElement.getValue();

      System.out.println(" > Async Handler has received a result of " + result);
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println("Exception in handleResponce");
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
```

```
          break;
        }
      } // end of for loop
    } // end of catch block
  }

}
```

Tarefas relacionadas

Desenvolvendo um cliente JAX-RPC para o transporte WebSphere para SOAP usando Eclipse
Desenvolva um cliente de serviço da web Axis 1.4 para executar usando o transporte WebSphere MQ para SOAP.

Desenvolvendo um cliente .NET 1 ou 2 para o transporte do WebSphere para SOAP usando o Microsoft Visual Studio 2008
Desenvolva um cliente de serviço da Web .NET 1 ou 2 para executar usando o transporte WebSphere MQ para SOAP.

Desenvolvendo um cliente .NET 1 ou 2 para o transporte do WebSphere para SOAP usando o Microsoft Visual Studio 2008

Desenvolva um cliente de serviço da Web .NET 1 ou 2 para executar usando o transporte WebSphere MQ para SOAP.

Antes de começar

É possível iniciar o desenvolvimento de um cliente .NET 1 ou 2 de várias maneiras diferentes:

1. Use **amqwdeployWMQService** para gerar stubs de cliente a partir de um serviço da web e importe-os no Visual Studio.
2. Use **java2wsdl** para gerar WSDL a partir de uma implementação Java de um serviço da Web e, em seguida, use `wsdl.exe`, que é enviado com .NET, para gerar stubs do cliente
3. Gerar WSDL a partir de uma implementação .NET `.asmx` do serviço usando **amqswsdl**e, em seguida, usar `wsdl.exe`.
4. Se você tiver desenvolvido e implementado o serviço para HTTP, use o **Incluir referência da web ...** assistente no Visual Studio para configurar o cliente para acessar o serviço HTTP Altere a URL para o serviço implementado no transporte do WebSphere MQ para SOAP.

A tarefa usa o serviço desenvolvido em"Desenvolvendo um serviço .NET 1 ou 2 para o transporte WebSphere MQ para SOAP usando o Microsoft Visual Studio 2008" na página 977.

Sobre esta tarefa

Siga estas etapas para criar um Cliente .NET 1 ou 2 para HTTP e transporte WebSphere MQ para SOAP.

Procedimento

1. Crie o aplicativo do console do cliente e modifique-o para chamar o serviço da web StockQuote HTTP.
   a) Clique com o botão direito em **Solução 'StockQuoteDotNet'** no **Solution Explorer** > Incluir ...> Novo Projeto. Selecione o **C#** Tipo de Projeto, **NET Framework 2.0**e **Aplicativo do Console**. Dê ao projeto o nome `StockQuoteClientDotNet` > **OK**
   b) Clique com o botão direito em **Solução 'StockQuoteDotNet'** no **Solution Explorer** > Incluir ...> Novo Projeto. Selecione o **C#** Tipo de Projeto, **NET Framework 2.0**e **Aplicativo do Console**. Dê ao projeto o nome `StockQuoteClientDotNet` > **OK**
   c) Clique com o botão direito em **StockQuoteClientDotNet** > **Configurar como projeto de inicialização**.
   d) Clique com o botão direito em **StockQuoteClientDotNet** > **Incluir referência da web ...** > Procurar serviços da web nesta solução> Selecionar **StockQuote** > **Incluir referência**. Observe que você incluiu uma referência da web para host local e um novo arquivo de configuração `app.config`.

e) No Solution Explorer, mude o nome do aplicativo de console de `Program.cs` para `StockQuoteClientDotNet.cs` > Clique em **OK** para mudar todos os usos de `Program.cs` para `StockQuoteClientDotNet.cs`.

f) Substitua o conteúdo de `StockQuoteClientDotNet.cs` com o código em Figura 190 na página 1009

```
using System;
using StockQuoteClientDotNet.localhost;
namespace StockQuoteClientDotNet {
  class StockQuoteClientDotNet {
    static void Main(string[] args) {
      try {
        StockQuote stockobj = new StockQuote();
        Console.WriteLine("http reply is: "
          + stockobj.getNonInlineQuote("http request");
      }
      catch (System.Exception e) {
        Console.WriteLine("Exception thrown: " + e.ToString());
      }
      Console.ReadLine();
    }
  }
}
```

Figura 190. Programa HTTP StockQuoteClientDotNet

g) Ative StockQuoteClientDotNet para testar no serviço `StockQuote.asmx`:

i) Pressione **F5**, clique na seta verde na barra de ação ou **Depurar** > **Iniciar depuração (F5)**.

Se o projeto StockQuoteDOtNet estiver na mesma solução, ele será iniciado automaticamente. Caso contrário, será necessário iniciar o serviço primeiro.

A janela de comandos com os resultados é aberta atrás da área de trabalho. A instrução `Console.ReadLine();` impede que feche até **Enter** ser pressionado.

Sugestão: Certifique-se de que `StockQuote.asmx` seja a página Iniciar no projeto `StockQuoteDotNet`.

2. Modifique StockQuoteClientDotNet para chamar o serviço `StockQuote.asmx` usando o transporte WebSphere MQ para SOAP.

a) Inclua as linhas mostradas em negrito no cliente.

```
using System;
using StockQuoteClientDotNet.localhost;
namespace StockQuoteClientDotNet {
  class StockQuoteClientDotNet {
    static void Main(string[] args) {
      try {
        IBM.WMQSOAP.Register.Extension();
        StockQuote stockobj = new StockQuote();
        Console.WriteLine("http reply is: "
          + stockobj.getNonInlineQuote("http request");
        stockobj.Url = "jms:/queue?"
          + "initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&connectionFactory=()&destination=REQUESTDOTNET@QM1"
          + "&targetService=StockQuote.asmx";
        Console.WriteLine("jms reply is: "
          + stockobj.getNonInlineQuote("jms request"));
      }
      catch (System.Exception e) {
        Console.WriteLine("Exception thrown: " + e.ToString());
      }
      Console.ReadLine();
    }
  }
}
```

Figura 191. Programa StockQuoteClientDotNet modificado

Como alternativa, modifique a URL padrão. Abra **StockQuoteClientDotNet** > **Propriedades** > **Settings.settings** e altere o valor da propriedade `StockQuoteClientDotNet_localhost_StockQuote` para o transporte WebSphere MQ para URL SOAP.

b) Inclua uma referência em `amqsoap.dll`

i) No projeto **StockQuoteClientDotNet** no **Explorador de Soluções**, clique com o botão direito em **Referências** > **Incluir Referência ...** > Clique na guia **Navegar** > navegue até *MQ_INSTALLATION_PATH*\bin > Selecionar **amqsoap.dll** > **OK**. *MQ_INSTALLATION_PATH* é o diretório onde o WebSphere MQ está instalado.

3. Teste o cliente com o serviço `StockQuote.asmx` usando o transporte WebSphere MQ para SOAP.

a) Abra uma janela de comando no diretório do projeto StockQuoteDotNet: `.\StockQuoteDotNet\StockQuoteDotNet` > Verifique se o `.bin\StockQuoteDotNet.dll` existe. Caso contrário, reconstrua a solução.

b) Digite o comando **amqwRegisterdotNet**.

Você precisa apenas executar **amqwRegisterdotNet** uma vez por instalação.

c) Se você tiver executado **amqwdeployWMQServer** com o `genAsmxWMQBits`, execute o Listener SOAP .NET: `generated\server\startWMQNListener`

d) Como alternativa, execute o listener diretamente:

```
amqwSOAPNETListener -u "jms:/queue?
destination=REQUESTDOTNET@QM1
&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi
&targetService=StockQuote.asmx&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE"
-w C:\IBM\ID\StockQuoteDotNet\StockQuoteDotNet -n 10
```

4. No Visual Studio 2008, pressione **F5** para executar StockQuoteClientDotNet.

Clientes de serviço da Web do .NET Framework 1 e do .NET Framework 2

Os clientes .NET de amostra fornecidos com o transporte WebSphere MQ para SOAP usam stubs gerados para chamar os serviços Axis e .NET de amostra..

Para clientes .NET Framework 1 e .NET Framework 2, o WebSphere MQ fornece acesso a serviços da Web usando clientes .NET. O comando **amqwdeployWMQService** tem uma opção, `genProxiestoDotNet`, que gera stubs do cliente .NET Framework 1 ou .NET Framework 2 para um serviço da Web. Também é possível usar stubs de cliente gerados pelo .NET **wsdl** ou pelo Microsoft Visual Studio 2005 ou 2008.

Os clientes de serviço da web .NET Framework 1 e .NET de amostra são instalados em *MQ_INSTALLATION_PATH*\tools\soap\samples\dotnet. *MQ_INSTALLATION_PATH* é o diretório onde o WebSphere MQ está instalado.

SQVB2Axis.vb
SQVB2Axis.vb, Figura 192 na página 1011, é o cliente Visual Basic para chamar o serviço **StockQuoteAxisService** .

SQVB2DotNet.vb
QVB2DotNet.vb, Figura 193 na página 1011, é o cliente Visual Basic para chamar o serviço **StockQuoteDotNet** .

SQCS2Axis.cs
SQCS2Axis.cs, Figura 194 na página 1011, é o cliente C# para chamar o serviço **StockQuoteAxisService** .. É possível substituir a URL do serviço fornecendo uma URL na linha de comandos.

SQCS2DotNet.cs
SQCS2DotNet.cs, Figura 195 na página 1012, é o cliente C# para chamar o serviço **StockQuoteDotNet** .. É possível substituir a URL do serviço fornecendo uma URL na linha de comandos.

```
Module SQVB2Axis
   Function Main(ByVal CmdArgs() As String) As Integer
      IBM.WMQSOAP.Register.Extension()
      Dim obj As New StockQuoteAxisService()
      Dim res As Single = obj.getQuote("fromcs")
      System.Console.WriteLine("SQVB2Axis: reply is: '{0}'", res)
   End Function
End Module
```

Figura 192. SQVB2Axis

```
Module SQVB2DotNet
   Function Main(ByVal CmdArgs() As String) As Integer
      IBM.WMQSOAP.Register.Extension()
      Dim obj as new StockQuoteDotNet()
      Dim res as Single = obj.getQuote("fromcs")
      System.Console.WriteLine("SQVB2DotNet: reply is: '{0}'", res)
   End Function
End Module
```

Figura 193. SQVB2DotNet

```
using System;
class SQCS2Axis {
   [STAThread]
   static void Main(string[] args) {
      try {
         IBM.WMQSOAP.Register.Extension();
         StockQuoteAxisService stockobj = new StockQuoteAxisService();
         if (args.GetLength(0) >= 1)
            stockobj.Url = args[0];
         System.Single res = stockobj.getQuote("XXX");
         Console.WriteLine("SQCS2Axis RPC reply is: " + res);
      }
      catch (System.Exception e) {
         Console.WriteLine("\n>>> EXCEPTION WHILE RUNNING SQCS2Axis DEMO <<<\n"
            + e.ToString());
      }
   }
}
```

Figura 194. SQCS2Axis

```
using System;
class SQCS2DotNet {
   [STAThread]
   static void Main(string[] args) {
      try {
         IBM.WMQSOAP.Register.Extension();
         StockQuoteDotNet stockobj = new StockQuoteDotNet();
         if (args.GetLength(0) >= 1)
            stockobj.Url = args[0];
         System.Single res = stockobj.getQuote("XXX");
         Console.WriteLine("RPC reply is: " + res);
         if (args.GetLength(0) == 0) {
            res = stockobj.getQuoteDOC("XXX");
            Console.WriteLine("DOC reply is: " + res);
         }
      }
      catch (System.Exception e) {
         Console.WriteLine("\n>>> EXCEPTION WHILE RUNNING SQCS2DotNet DEMO <<<\n"
            + e.ToString());
      }
   }
}
```

Figura 195. SQCS2DotNet

Tarefas relacionadas

Desenvolvendo um cliente JAX-RPC para o transporte WebSphere para SOAP usando Eclipse
Desenvolva um cliente de serviço da web Axis 1.4 para executar usando o transporte WebSphere MQ para SOAP.

Desenvolvendo um Cliente JAX-WS para WebSphere Transport for SOAP Usando Eclipse
Desenvolva um cliente de serviço da Web Axis2 para executar usando o WebSphere MQ Transport for SOAP. Os clientes Axis2 de amostra fornecidos com o WebSphere MQ Transport for SOAP são listados e o comando **wsimport** é usado para gerar proxies.

Implementando Serviços da Web Usando o Transporte WebSphere MQ para SOAP

Implemente um serviço da web em um de vários ambientes de servidor diferentes e conecte-se a ele usando o transporte do WebSphere MQ para SOAP

Antes de começar

Desenvolva um serviço da web e teste-o usando o SOAP sobre HTTP no ambiente de destino.

Sobre esta tarefa

É possível implementar um serviço da web para executar com o transporte WebSphere MQ para SOAP em vários ambientes de tempo de execução SOAP diferentes É possível implementar um serviço no Axis 1.4 usando apenas o software instalado com WebSphere MQ. Para os ambientes de tempo de execução diferentes, deve-se instalar o software adicional.

Você não está restrito a executar o transporte WebSphere MQ para SOAP para os servidores para os quais há instruções de implementação. Use as instruções para implementar um serviço em um dos ambientes listados.

Nota: Alguns ambientes integrados oferecem SOAP sobre JMS utilizando a ligação SOAP JMS recomendada pelo W3C , bem como o transporte WebSphere MQ para ligação SOAP As liberações do WebSphere MQ, até e incluindo 7.0.1.2, suportam apenas o transporte do WebSphere MQ para ligação SOAP. De 7.0.1.3 em diante, é possível implementar clientes Axis2 usando um URI que esteja em conformidade com a recomendação do candidato W3C para SOAP sobre JMS. Consulte o tutorial, Desenvolva um aplicativo de serviços da web SOAP/JMS com o WebSphere Application Server V7 e o Rational Application Developer V7.5.

Implementando um serviço no Axis 1.4 para usar para o transporte WebSphere para SOAP usando amqwdeployWMQService

Implemente um serviço Axis 1.4 no transporte WebSphere MQ para SOAP criando um diretório de implementação, executando o comando **amqwdeployWMQService** e iniciando o listener do Axis 1.4 .

Antes de começar

1. Siga as instruções para instalar o transporte do WebSphere MQ para SOAP
2. Verifique a instalação e seu ambiente usando o comando **runivt**.
3. Para reimplementar um serviço:
   a. Exclua o subdiretório `./generated` e todos os seus subdiretórios.
   b. Remova solicitações da fila de destino e as exclua.
   c. Continue com as instruções da etapa “2” na página 1013.

Sobre esta tarefa

Essas instruções são para implementar um serviço do Axis 1.4 pela primeira vez. Para reiniciar um serviço do Axis 1.4, execute novamente o listener de SOAP do Axis 1.4: etapa “11” na página 1014.

Use as instruções a seguir para implementar um novo serviço Axis 1.4 no transporte WebSphere MQ para SOAP:

Procedimento

1. Crie um diretório *deployDir* para manter os arquivos de implementação.
   O utilitário de implementação requer que cada serviço seja implementado a partir de um diretório separado.
2. Abra uma janela de comandos no Windows ou um shell de comando usando o X Window System em sistemas UNIX and Linux em *deployDir* para executar **amqwdeployWMQService**.
3. Execute **amqwsetcp** para configurar o caminho de classe.
   JRE e JDK devem estar no caminho de classe, na versão 5.0 ou posterior, e no mesmo nível de versão.
4. Copie a origem da classe, *className*`.java`, no *deployDir*
5. Copie todos os arquivos de origem Java no mesmo pacote que *className* em *deployDir*/*packageName*, em que *packageName* é uma árvore de diretório correspondente ao nome do pacote.
6. Executar o **javac** *packageName*`.`*className*`.`
   Pode ser necessário incluir um caminho para o diretório atual "`.`"ou para o diretório *packageName* para **javac** localizar as outras classes.
7. Crie o WSDL do Axis para o serviço:

```
amqwdeployWMQService -f packageName.className.java -c genAxisWsdl
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

8. Crie os recursos do WebSphere MQ para o serviço:

```
amqwdeployWMQService -f packageName.className.java -c genAxisWMQBits
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

Sugestão:

Se desejar configurar um novo gerenciador de filas e os recursos de que necessita, para realizar desenvolvimento e teste, execute **setupWMQSOAP**.

Se desejar configurar o novo gerenciador de filas como o padrão, tome uma cópia de **setupWMQSOAP** do diretório *WMQ install directory*\tools\soap\samples e inclua o parâmetro -q na linha

```
call :try -q crtmqm %QMGR%
```

9. Crie o listener do Axis e implemente o serviço:

```
amqwdeployWMQService -f packageName.className.java -c AxisDeploy
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

10. Se for necessário gerar o WSDL para o serviço, gere stubs do cliente ou proxies de cliente, execute **amqwdeployWMQService** com um dos parâmetros a seguir:

    - genAsmxWsdl
    - genAxisWsdl
    - genProxiesToDotNet
    - genProxiestoAxis

    Nota: Deve-se gerar o WSDL antes de gerar os proxies. A opção AllAxis falhará se o CLASSPATH não estiver configurado para localizar todas as classes que são importadas para compilar *className*.java. Se houver diversos arquivos Java no pacote que contém *className*.java, deve-se compilá-los primeiro usando **javac amqwdeployWMQService** -f *packageName*.*className*.java -c CompileJava compila apenas *className*.java.

11. Inicie o listener do Axis gerado.

```
.\generated\server\startWMQJListener.cmd
```

Tarefas relacionadas

Implementando um serviço no serviço .NET Framework 1 ou 2 para usar o transporte do WebSphere MQ para SOAP
Implemente um serviço .NET Framework 1 ou 2 no transporte do WebSphere MQ para SOAP. Crie um diretório de implementação, execute o comando **amqwdeployWMQService** e inicie o listener .NET.

Implementando um serviço no CICS Transaction Server para usar o WebSphere Transport for SOAP
O transporte do WebSphere MQ para SOAP é integrado ao suporte de serviço da web do CICS Transaction Server 4.1

Implementando um serviço no WebSphere Application Server para usar o WebSphere Transporte para SOAP
O transporte do WebSphere MQ para SOAP é integrado ao barramento de integração de serviços no WebSphere Application Server.

Configurando o WebSphere Application Server para usar o W3C SOAP sobre JMS
Um serviço da Web ligado à recomendação do candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos Java EE . Esta tarefa é a etapa 1 de conectar um cliente de serviço da web Axis2 e um serviço da web implementado no servidor de aplicativos WebSphere usando o protocolo SOAP sobre JMS W3C . Configure os recursos do WebSphere MQ e do WebSphere Application Server para desenvolver e implementar o serviço da Web ligado ao W3C SOAP sobre JMS como um transporte

Implementando um serviço no terminal em serviço do WebSphere ESB e do Process Server para usar o WebSphere Transport for SOAP
WebSphere MQ transporte para SOAP não é diretamente suportado pelo WebSphere ESB e Process Server. Deve-se configurar um Export customizado.

Implementando um serviço no serviço .NET Framework 1 ou 2 para usar o transporte do WebSphere MQ para SOAP

Implemente um serviço .NET Framework 1 ou 2 no transporte do WebSphere MQ para SOAP. Crie um diretório de implementação, execute o comando **amqwdeployWMQService** e inicie o listener .NET.

Antes de começar

1. Siga as instruções para instalar o transporte do WebSphere MQ para SOAP
2. Verifique a instalação e seu ambiente usando o comando **runivt**.
3. O caminho para os arquivos de estrutura .NET `wsdl.exe` e `csc.exe` deve ser configurado As cópias de `wsdl.exe` e `csc.exe` identificadas pela variável PATH devem estar no mesmo nível da estrutura .NET. Se você tiver várias estruturas .NET instaladas ou estiver usando o Visual Studio, verifique a variável PATH cuidadosamente.
4. Para reimplementar um serviço:
   a. Exclua o subdiretório `./generated` e todos os seus subdiretórios
   b. Remova solicitações da fila de destino e as exclua.
   c. Continue com as instruções da etapa “2” na página 1015.

Sobre esta tarefa

Estas instruções são para implementar um serviço .NET pela primeira vez. Para reiniciar um serviço .NET, execute novamente o listener SOAP do .NET, etapa “9” na página 1016

Use as instruções a seguir para implementar um novo serviço .NET Framework 1 ou .NET Framework 2 no transporte WebSphere MQ para SOAP:

Procedimento

1. Crie um diretório *deployDir* para manter os arquivos de implementação.

   O utilitário de implementação requer que cada serviço seja implementado a partir de um diretório separado.
2. Abra uma janela de comando no *deployDir* para executar o **amqwdeployWMQService**.

```
C:\IBM\ID\QuoteClient>
```

3. Execute **amqwsetcp** para configurar o caminho de classe.

   Um caminho de classe é necessário apenas para clientes do Axis.
4. Copie o serviço .NET, *className*`.asmx`, em *deployDir*
5. Construir a implementação de serviço em uma biblioteca (`.dll`).

   A implementação do serviço sequencial está em *className*`.asmx`. A implementação do serviço de code-behind pode ser *className*`.asmx.cs`.

   Figura 196 na página 1015 mostra um exemplo de um comando para construir um serviço .NET Framework V2 como uma biblioteca.

```
c:\WINDOWS\Microsoft.NET\Framework\v3.5\Csc.exe /noconfig /nowarn:1701,1702
/errorreport:prompt /warn:4 /define:TRACE
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.configuration.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Data.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Drawing.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Web.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Web.Services.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Xml.dll
/debug:pdbonly /filealign:512 /optimize+
/out:obj\Quote.dll /target:library Properties\AssemblyInfo.cs Quote.asmx.cs
```

   Figura 196. Comando de construção para o serviço .NET Framework V2

6. Copie o *className*`.dll` em *deployDir*`\bin`
7. Configure os recursos do WebSphere MQ e crie o listener necessário para o serviço:

```
amqwdeployWMQService -f className.asmx -c genAsmxWMQBits
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))
&targetService=className.asmx"
```

8. Se for necessário gerar o WSDL para o serviço, gere stubs do cliente ou proxies de cliente, execute **amqwdeployWMQService** com um dos parâmetros a seguir:
   - genAsmxWsdl
   - genAxisWsdl
   - genProxiesToDotNet
   - genProxiestoAxis

   Nota: Deve-se gerar o WSDL antes de gerar os proxies.
9. Inicie o listener do .NET gerado

```
.\generated\server\startWMQNListener.cmd
```

Tarefas relacionadas

Implementando um serviço no Axis 1.4 para usar para o transporte WebSphere para SOAP usando amqwdeployWMQService
Implemente um serviço Axis 1.4 no transporte WebSphere MQ para SOAP criando um diretório de implementação, executando o comando **amqwdeployWMQService** e iniciando o listener do Axis 1.4 .

Implementando um serviço no CICS Transaction Server para usar o WebSphere Transport for SOAP
O transporte do WebSphere MQ para SOAP é integrado ao suporte de serviço da web do CICS Transaction Server 4.1

Implementando um serviço no WebSphere Application Server para usar o WebSphere Transporte para SOAP
O transporte do WebSphere MQ para SOAP é integrado ao barramento de integração de serviços no WebSphere Application Server.

Configurando o WebSphere Application Server para usar o W3C SOAP sobre JMS
Um serviço da Web ligado à recomendação do candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos Java EE . Esta tarefa é a etapa 1 de conectar um cliente de serviço da web Axis2 e um serviço da web implementado no servidor de aplicativos WebSphere usando o protocolo SOAP sobre JMS W3C . Configure os recursos do WebSphere MQ e do WebSphere Application Server para desenvolver e implementar o serviço da Web ligado ao W3C SOAP sobre JMS como um transporte

Implementando um serviço no terminal em serviço do WebSphere ESB e do Process Server para usar o WebSphere Transport for SOAP
WebSphere MQ transporte para SOAP não é diretamente suportado pelo WebSphere ESB e Process Server. Deve-se configurar um Export customizado.

Implementando um serviço no CICS Transaction Server para usar o WebSphere Transport for SOAP

O transporte do WebSphere MQ para SOAP é integrado ao suporte de serviço da web do CICS Transaction Server 4.1

Antes de começar

Use as mesmas ferramentas para desenvolver para um cliente ou serviço para o WebSphere MQ, como você desenvolveria para HTTP. O CICS possui ferramentas correspondentes a **Java2wsdl** e **wsdl2Java**:

- **DFHWS2LS** tem uma descrição de serviço da web como um ponto de início. Ele usa as descrições das mensagens e os tipos de dados usados nessas mensagens para construir estruturas de dados de linguagem de alto nível. É possível usar nas estruturas em programas de aplicativos gravados em linguagens diferentes.

- **DFHLS2WS** tem uma estrutura de dados de linguagem de alto nível como um ponto de início. Ele usa a estrutura para construir uma descrição de serviços da web que contém descrições de mensagens. Ele também cria os esquemas para as mensagens da estrutura de dados de linguagem.

Siga as instruções Criando um serviço da web na documentação do produto CICS para criar um serviço da web.

Sobre esta tarefa

Siga as instruções, Configurando CICS para usar o transporte WebSphere MQ na documentação do produto CICS . Usando as instruções, é possível implementar o serviço da web no transporte WebSphere MQ para SOAP.

Tarefas relacionadas

Implementando um serviço no Axis 1.4 para usar para o transporte WebSphere para SOAP usando amqwdeployWMQService
Implemente um serviço Axis 1.4 no transporte WebSphere MQ para SOAP criando um diretório de implementação, executando o comando **amqwdeployWMQService** e iniciando o listener do Axis 1.4 .

Implementando um serviço no serviço .NET Framework 1 ou 2 para usar o transporte do WebSphere MQ para SOAP
Implemente um serviço .NET Framework 1 ou 2 no transporte do WebSphere MQ para SOAP. Crie um diretório de implementação, execute o comando **amqwdeployWMQService** e inicie o listener .NET.

Implementando um serviço no WebSphere Application Server para usar o WebSphere Transporte para SOAP
O transporte do WebSphere MQ para SOAP é integrado ao barramento de integração de serviços no WebSphere Application Server.

Configurando o WebSphere Application Server para usar o W3C SOAP sobre JMS
Um serviço da Web ligado à recomendação do candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos Java EE . Esta tarefa é a etapa 1 de conectar um cliente de serviço da web Axis2 e um serviço da web implementado no servidor de aplicativos WebSphere usando o protocolo SOAP sobre JMS W3C . Configure os recursos do WebSphere MQ e do WebSphere Application Server para desenvolver e implementar o serviço da Web ligado ao W3C SOAP sobre JMS como um transporte

Implementando um serviço no terminal em serviço do WebSphere ESB e do Process Server para usar o WebSphere Transport for SOAP
WebSphere MQ transporte para SOAP não é diretamente suportado pelo WebSphere ESB e Process Server. Deve-se configurar um Export customizado.

Implementando um serviço no WebSphere Application Server para usar o WebSphere Transporte para SOAP

O transporte do WebSphere MQ para SOAP é integrado ao barramento de integração de serviços no WebSphere Application Server.

Antes de começar

Use o Rational Application Developer, o WebSphere Integration Developer ou um kit de ferramentas de serviços da web para desenvolver o serviço da web

Sobre esta tarefa

Use as instruções a seguir para implementar um serviço usando o transporte do WebSphere MQ para SOAP como um transporte SOAP no WebSphere Application Server

Procedimento

1. Configure o WebSphere MQ como o provedor de sistemas de mensagens JMS para o barramento de integração de serviços no WebSphere Application Server.
2. Configure os recursos do WebSphere MQ requeridos pelo serviço

3. Siga as instruções, Configurando recursos JMS para o listener terminal SOAP sobre JMS síncrono, na documentação do produto WebSphere Application Server Network Deployment.

   Há instruções correspondentes para outras plataformas do WebSphere Application Server.
4. Modifique o URI de serviço para estar em conformidade com o transporte WebSphere MQ para o URI SOAP.
5. Implemente o serviço no WebSphere Application Server.

Como proceder a seguir

Implemente o serviço com HTTP como um transporte para que os clientes possam consultar o serviço e receber o WSDL na resposta.

Tarefas relacionadas

Implementando um serviço no Axis 1.4 para usar para o transporte WebSphere para SOAP usando amqwdeployWMQService

Implemente um serviço Axis 1.4 no transporte WebSphere MQ para SOAP criando um diretório de implementação, executando o comando **amqwdeployWMQService** e iniciando o listener do Axis 1.4 .

Implementando um serviço no serviço .NET Framework 1 ou 2 para usar o transporte do WebSphere MQ para SOAP

Implemente um serviço .NET Framework 1 ou 2 no transporte do WebSphere MQ para SOAP. Crie um diretório de implementação, execute o comando **amqwdeployWMQService** e inicie o listener .NET.

Implementando um serviço no CICS Transaction Server para usar o WebSphere Transport for SOAP

O transporte do WebSphere MQ para SOAP é integrado ao suporte de serviço da web do CICS Transaction Server 4.1

Configurando o WebSphere Application Server para usar o W3C SOAP sobre JMS

Um serviço da Web ligado à recomendação do candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos Java EE . Esta tarefa é a etapa 1 de conectar um cliente de serviço da web Axis2 e um serviço da web implementado no servidor de aplicativos WebSphere usando o protocolo SOAP sobre JMS W3C . Configure os recursos do WebSphere MQ e do WebSphere Application Server para desenvolver e implementar o serviço da Web ligado ao W3C SOAP sobre JMS como um transporte

Implementando um serviço no terminal em serviço do WebSphere ESB e do Process Server para usar o WebSphere Transport for SOAP

WebSphere MQ transporte para SOAP não é diretamente suportado pelo WebSphere ESB e Process Server. Deve-se configurar um Export customizado.

Configurando o WebSphere Application Server para usar o W3C SOAP sobre JMS

Um serviço da Web ligado à recomendação do candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos Java EE . Esta tarefa é a etapa 1 de conectar um cliente de serviço da web Axis2 e um serviço da web implementado no servidor de aplicativos WebSphere usando o protocolo SOAP sobre JMS W3C . Configure os recursos do WebSphere MQ e do WebSphere Application Server para desenvolver e implementar o serviço da Web ligado ao W3C SOAP sobre JMS como um transporte

Antes de começar

A tarefa requer WebSphere Application Server v7.0.0.9 e WebSphere MQ v7.0.1.3.

Sobre esta tarefa

A tarefa tem duas etapas:

Procedimento

1. “Configure recursos do WebSphere MQ” na página 1019
2. “Configure recursos do WebSphere Application Server” na página 1020

Como proceder a seguir

“Configure recursos do WebSphere MQ” na página 1019

Tarefas relacionadas

Implementando um serviço no Axis 1.4 para usar para o transporte WebSphere para SOAP usando amqwdeployWMQService
Implemente um serviço Axis 1.4 no transporte WebSphere MQ para SOAP criando um diretório de implementação, executando o comando **amqwdeployWMQService** e iniciando o listener do Axis 1.4 .

Implementando um serviço no serviço .NET Framework 1 ou 2 para usar o transporte do WebSphere MQ para SOAP
Implemente um serviço .NET Framework 1 ou 2 no transporte do WebSphere MQ para SOAP. Crie um diretório de implementação, execute o comando **amqwdeployWMQService** e inicie o listener .NET.

Implementando um serviço no CICS Transaction Server para usar o WebSphere Transport for SOAP
O transporte do WebSphere MQ para SOAP é integrado ao suporte de serviço da web do CICS Transaction Server 4.1

Implementando um serviço no WebSphere Application Server para usar o WebSphere Transporte para SOAP
O transporte do WebSphere MQ para SOAP é integrado ao barramento de integração de serviços no WebSphere Application Server.

Implementando um serviço no terminal em serviço do WebSphere ESB e do Process Server para usar o WebSphere Transport for SOAP
WebSphere MQ transporte para SOAP não é diretamente suportado pelo WebSphere ESB e Process Server. Deve-se configurar um Export customizado.

Configure recursos do WebSphere MQ

Antes de começar

Para o suporte Axis2 , você precisou WebSphere MQ 7.0.1.3 ou posterior.

Sobre esta tarefa

Para simplificar, a tarefa assume que o WebSphere MQ está instalado na mesma estação de trabalho que o outro software e usa conexões de ligações.. O WebSphere Application Server e as configurações do cliente Axis2 trabalham com conexões do cliente. Para executar com a tarefa usando conexões do cliente, verifique se é possível colocar e obter mensagens para e a partir das filas de solicitação e resposta do cliente Axis2 e dos computadores do WebSphere Application Server.

Novamente, para simplificar, nenhuma configuração de segurança é usada. O ID do usuário possui total autoridade `mqm`.

Procedimento

1. Crie um gerenciador de filas padrão, `QM1`.

   Use WebSphere MQ Explorer para criar `QM1` como um gerenciador de filas padrão. Configure-o para iniciar automaticamente e selecione a opção para criar um listener. Como alternativa, use os seguintes comandos:

   ```
   crtmqm -q -sa QM1
   strmqm
   echo define listener (LISTENER.TCP) trptype(tcp) ipaddr(localhost) port(1414)
              control(qmgr) replace | runmqsc
   echo start listener(LISTENER.TCP) | runmqsc
   ```

2. Defina uma fila de pedidos, `REQUESTAXIS`, e uma fila de resposta, `REPLYAXIS`.

   Use o Explorer ou os seguintes comandos:

   ```
   echo define ql(REQUESTAXIS) replace | runmqsc
   echo define ql(REPLYAXIS) replace | runmqsc
   ```

Como proceder a seguir

"Configure recursos do WebSphere Application Server" na página 1020

Configure recursos do WebSphere Application Server

Antes de começar

Para o suporte SOAP sobre JMS W3C , é necessário WebSphere Application Server v7. Essa configuração foi executada no WebSphere Application Server Versão 7.0 Test Environment v7.0.0.9 Atualização 1. WebSphere Application Server foi fornecido com o Rational Software Architect para WebSphere Software 7.5.4. Rational Software Architect foi atualizado para v7.5.5.1, aplicando as atualizações mais recentes que estavam disponíveis.

Como parte do processo de instalação, crie um perfil para o WebSphere Application Server. Na tarefa, a segurança administrativa não está ativada. O nome do perfil padrão é `was70profile1` e o servidor é `server1`.

Sobre esta tarefa

Configure o WebSphere Application Server. É possível iniciar o servidor a partir do Rational Application Developer e iniciar o console administrativo a partir da visualização Servidores ou é possível iniciar o servidor usando um arquivo de comando e administrar o servidor usando um navegador. A tarefa usa o segundo método.

Os arquivos de comando do servidor estão na pasta, *Rational Installation Root*`\SDP\runtimes\base_v7\profiles\was70profile1\bin` Os arquivos de log que você pode desejar inspecionar estão em *Rational Installation Root*`\SDP\runtimes\base_v7\profiles\was70profile1\logs\server1`

Como uma Convenção, todos os nomes de objetos do WebSphere MQ são maiúsculos e todos os nomes JNDI que fazem referência aos objetos do WebSphere MQ são minúsculos.

Procedimento

1. Inicialize o servidor.

   ```
   startServer server1
   ```

2. Inicie um navegador, abra o console de administração e efetue login.

   ```
   http://localhost:9061/ibm/console/unsecureLogon.jsp
   ```

   Digite qualquer sequência no campo ID do usuário.

3. Crie uma connection factory, `qm1`
   a) No navegador, abra **Recursos** > **JMS** > **Connection factories**.
   b) Na janela Connection factories, selecione o escopo **Node =*nodename***, clique em **Novo**.
   c) Selecione **WebSphere MQ provedor de sistemas de mensagens** > **OK**.
   d) Forneça as informações de conexão do gerenciador de filas de Tabela 143 na página 1020 > **Avançar**.

   Tabela 143. Informações de conexão do gerenciador de filas

Nome de Campo	**Value**
Nome	`qm1`
Nome JNDI	`qm1`

   e) Selecione **Inserir todas as informações necessárias neste assistente** como o método de conexão> **Avançar**

f) Digite QM1 como detalhes da conexão da fila> **Avançar**.

g) Insira os detalhes da conexão de Tabela 144 na página 1021 > **Avançar**.

Tabela 144. Detalhes de Conexão	
Nome de Campo	**Value**
Transporte	`Bindings, then client`
Nome do Host	`localhost`
Port	`1414`
Canal de conexão do servidor	`SYSTEM.DEF.SVRCONN`

h) **Conexão de teste** > **Avançar** > **Concluir** > **Salvar**

4. Crie a fila de solicitações do JMS, `requestaxis`

a) No Navigator, abra **Recursos** > **JMS** > **Filas**.

b) Na janela Connection factories, selecione o escopo **Node =*nodename***, clique em **Novo**.

c) Selecione **WebSphere MQ provedor de sistemas de mensagens** > **OK**.

d) Insira os detalhes da fila em Tabela 145 na página 1021 > **OK** > **Salvar**.

Tabela 145. Detalhes da fila	
Nome de Campo	**Value**
Nome	`requestaxis`
Nome JNDI	`requestaxis`
Nome da fila	`REQUESTAXIS`
Nome do gerenciador de filas	`QM1`

5. Repita a etapa "4" na página 1021 para criar a fila de resposta do JMS, `replyaxis`

6. Crie uma especificação de ativação, `qm1as`.

A especificação de ativação aciona o roteador do serviço da web MDB (Message Driven Bean) quando uma mensagem chega na fila de pedidos. O MDB é definido no descritor de implementação do serviço da Web criado pelo assistente de serviço da web do Rational Application Developer.

a) No Navigator, abra **Recursos** > **JMS** > **Especificações de Ativação**

b) Na janela Connection factories, selecione o escopo **Node =*nodename***, clique em **Novo**.

c) Selecione **WebSphere MQ provedor de sistemas de mensagens** > **OK**.

d) Insira os atributos básicos da especificação de ativação em Tabela 146 na página 1021 > **Avançar**.

Tabela 146. Nome da especificação de ativação	
Nome de Campo	**Value**
Nome	`qm1as`
Nome JNDI	`qm1as`

e) Especifique suas informações de MBD em Tabela 147 na página 1021 > **Avançar**.

Tabela 147. Informações do MDB	
Nome de Campo	**Value**
Nome da JNDI de destino	`requestaxis`
Seletor de mensagem	*Espaço em branco à esquerda*
Tipo de destino	`Queue`

f) Selecione **Inserir todas as informações necessárias neste assistente** como o método de conexão> **Avançar**

g) Digite `QM1` como detalhes da conexão da fila> **Avançar**.

h) Insira os detalhes da conexão de Tabela 144 na página 1021 > **Avançar**.

Tabela 148. Detalhes de Conexão	
Nome de Campo	**Value**
Transporte	`Bindings, then client`
Nome do Host	`localhost`
Port	`1414`
Canal de conexão do servidor	`SYSTEM.DEF.SVRCONN`

i) **Conexão de teste** > **Avançar** > **Concluir** > **Salvar**

7. Crie um connection factory da fila, `jms/WebServicesReplyQCF`, para a fila de resposta.

   O roteador de serviços da web usa um connection factory de fila para acessar uma fila de resposta. No descritor de implementação do serviço da web do connection factory de fila, é fornecido o nome JNDI padrão de `jms/WebServicesReplyQCF`. É possível mudar o nome no descritor de implementação. Nesta tarefa, inclua o nome padrão para as definições de recurso JMS

   a) No Navigator, abra **Recursos** > **JMS** > **Connection Factories de fila**

   b) Na janela Connection factories, selecione o escopo **Node =*nodename***, clique em **Novo**.

   c) Selecione **WebSphere MQ provedor de sistemas de mensagens** > **OK**.

   d) Insira os atributos básicos do factory de conexão da fila em Tabela 149 na página 1022 > **Avançar**.

Tabela 149. Nome da connection factory de filas	
Nome de Campo	**Value**
Nome	`WebServicesReplyQCF`
Nome JNDI	`jms/WebServicesReplyQCF`

   e) Selecione **Inserir todas as informações necessárias neste assistente** como o método de conexão> **Avançar**

   f) Digite `QM1` como detalhes da conexão da fila> **Avançar**.

   g) Insira os detalhes da conexão de Tabela 144 na página 1021 > **Avançar**.

Tabela 150. Detalhes de Conexão	
Nome de Campo	**Value**
Transporte	`Bindings, then client`
Nome do Host	`localhost`
Port	`1414`
Canal de conexão do servidor	`SYSTEM.DEF.SVRCONN`

   h) **Conexão de teste** > **Avançar** > **Concluir** > **Salvar**

Como proceder a seguir

"Desenvolvendo um Serviço da Web EJB JAX-WS para SOAP W3C sobre JMS" na página 980

Implementando um serviço no terminal em serviço do WebSphere ESB e do Process Server para usar o WebSphere Transport for SOAP

WebSphere MQ transporte para SOAP não é diretamente suportado pelo WebSphere ESB e Process Server. Deve-se configurar um Export customizado.

Sobre esta tarefa

WebSphere Integration Developer fornece uma transformação de dados SOAP que pode ser ligada à Exportação JMS do WebSphere MQ para criar uma Exportação SOAP JMS customizada do WebSphere MQ .

Siga as instruções para criar uma Exportação customizada para receber solicitações SOAP sobre o transporte WebSphere MQ para SOAP.

Procedimento

1. Leia Visão geral de importações e exportações e Como conectar-se ao WebSphere MQ na documentação do produto WebSphere Process Server for Multiplatforms V6.2
2. Siga a tarefa Gerando uma MQ ligação de exportação JMS na documentação do produto IBM Business Process Manager, Versão 8.6 .

   Use a ligação de dados SOAP descrita em Transformações de formato de dados JMS pré-empacotados para formatar a mensagem SOAP

Tarefas relacionadas

Implementando um serviço no Axis 1.4 para usar para o transporte WebSphere para SOAP usando amqwdeployWMQService
Implemente um serviço Axis 1.4 no transporte WebSphere MQ para SOAP criando um diretório de implementação, executando o comando **amqwdeployWMQService** e iniciando o listener do Axis 1.4 .

Implementando um serviço no serviço .NET Framework 1 ou 2 para usar o transporte do WebSphere MQ para SOAP
Implemente um serviço .NET Framework 1 ou 2 no transporte do WebSphere MQ para SOAP. Crie um diretório de implementação, execute o comando **amqwdeployWMQService** e inicie o listener .NET.

Implementando um serviço no CICS Transaction Server para usar o WebSphere Transport for SOAP
O transporte do WebSphere MQ para SOAP é integrado ao suporte de serviço da web do CICS Transaction Server 4.1

Implementando um serviço no WebSphere Application Server para usar o WebSphere Transporte para SOAP
O transporte do WebSphere MQ para SOAP é integrado ao barramento de integração de serviços no WebSphere Application Server.

Configurando o WebSphere Application Server para usar o W3C SOAP sobre JMS
Um serviço da Web ligado à recomendação do candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos Java EE . Esta tarefa é a etapa 1 de conectar um cliente de serviço da web Axis2 e um serviço da web implementado no servidor de aplicativos WebSphere usando o protocolo SOAP sobre JMS W3C . Configure os recursos do WebSphere MQ e do WebSphere Application Server para desenvolver e implementar o serviço da Web ligado ao W3C SOAP sobre JMS como um transporte

Implementando clientes de serviço da Web para usar o transporte WebSphere MQ para SOAP

Implemente um cliente de serviço da web em um de vários ambientes de cliente diferentes e conecte-se a um serviço usando o transporte do WebSphere MQ para SOAP

Antes de começar

Desenvolva o serviço da Web e implemente-o para usar o transporte do WebSphere MQ para SOAP

Sobre esta tarefa

É possível implementar um cliente de serviço da web a ser executado com o transporte WebSphere MQ para SOAP em vários ambientes de cliente diferentes É possível implementar um cliente Java no Axis 1.4 usando apenas o software instalado com WebSphere MQ. Para os outros ambientes de cliente, deve-se instalar o software adicional.

Você não está restrito a executar o transporte do WebSphere para SOAP nos ambientes do cliente para os quais há instruções de implementação Use as instruções para implementar um cliente a um dos ambientes suportados.

Nota: Alguns ambientes integrados oferecem SOAP sobre JMS utilizando a ligação SOAP JMS recomendada pelo W3C , bem como o transporte WebSphere MQ para ligação SOAP As liberações do WebSphere MQ, até e incluindo 7.0.1.2, suportam apenas o transporte do WebSphere MQ para ligação SOAP. De 7.0.1.3 em diante, é possível implementar clientes Axis2 usando um URI que esteja em conformidade com a recomendação do candidato W3C para SOAP sobre JMS. Consulte o tutorial, Desenvolva um aplicativo de serviços da web SOAP/JMS com o WebSphere Application Server V7 e o Rational Application Developer V7.5.

Implementando um cliente de serviço da web para o Axis 1.4 para usar o transporte do IBM WebSphere MQ para SOAP

Prepare um diretório de implementação e um descritor de implementação para o cliente. Forneça os proxies do cliente e a classe do cliente e configure CLASSPATH. Configure as filas e os canais do IBM WebSphere MQ, inicie o serviço e teste o cliente.

Antes de começar

Sugestão: Implemente o serviço para HTTP, desenvolva e teste o cliente para HTTP e, em seguida, modifique o cliente para o transporte do IBM WebSphere MQ para SOAP:

1. Inclua a chamada Register.extension() no cliente.
2. Mude o endereço de serviço da web estático na classe do localizador de proxy de cliente para usar o URI para o transporte do IBM WebSphere MQ para SOAP.

Sobre esta tarefa

Implementando um cliente Axis 1.4 para usar o transporte do IBM WebSphere MQ para SOAP requer uma etapa de implementação adicional comparada com um cliente HTTP. Deve-se criar um descritor de implementação do cliente, `client-config.wsdd`, para mapear o transporte `jms:` para o emissor de classe `com.ibm.mq.soap.transport.jms.WMQSender`.

Se você usar o comando **amqwdeployWMQService** para gerar proxies de cliente, será possível implementar o cliente usando os diretórios que o comando gera.

Procedimento

1. Crie um diretório *deployDir* para manter os arquivos de implementação do cliente.
2. Abra uma janela de comandos em sistemas Windows ou um shell de comando usando X Window System em sistemas UNIX and Linux, em *deployDir*.
3. Execute o comando **amqwsetcp.cmd** para configurar o CLASSPATH
4. Execute o comando **amqwclientconfig.cmd** para criar um descritor de implementação do cliente Axis 1.4, `client-config.wsdd` em *deployDir*.
5. Certifique-se de que as classes no pacote do cliente, as classes de proxy cliente e as bibliotecas que o cliente usa estejam no CLASSPATH.

   O **amqwdeployWMQService** coloca os proxies do cliente .NET em `./generated/server/soap/client/remote/dotnetService` e os proxies do Axis 1.4 em `./generated/server/soap/client/remote/`*`client package`*

exemplo

O exemplo mostra a configuração e a saída, Figura 199 na página 1025, a partir de um cliente Java Axis 1.4. O cliente, Figura 198 na página 1025, chama um serviço da web que ecoa seu parâmetro de entrada. A definição de serviço, Figura 197 na página 1025, mostra o URI obtido do serviço WSDL.

```
<wsdl:service name="QuoteSOAPImplService">
    wsdl:port binding="intf:org.example.www.QuoteSOAPImplBindingSoap"
                      name="org.example.www.QuoteSOAPImpl_Wmq">
       <wsdlsoap:address location="jms:/queue?destination=REQUESTAXIS
           &connectionFactory=(connectQueueManager(QM1)binding(server))
           &initialContextFactory=com.ibm.mq.jms.Nojndi
           &targetService=org.example.www.QuoteSOAPImpl.java
           &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE" />
    </wsdl:port>
</wsdl:service>
```

Figura 197. Definição de serviço

```
package org.example.www;
import com.ibm.mq.soap.Register;
public class QuoteClient {
   public static void main(String[] args) {
      try {
         Register.extension();
         QuoteSOAPImplServiceLocator locator = new QuoteSOAPImplServiceLocator();
         System.out.println("Response = "
                  + locator.getOrgExampleWwwQuoteSOAPImpl_Wmq().getQuote("IBM"));
      } catch (Exception e) {
         System.out.println("Exception = " + e.getMessage());
      }
   }
}
```

Figura 198. Cliente do Axis 1.4 Java

```
C:\IBM\ID\Test>dir /s /b
C:\IBM\ID\Test\client-config.wsdd
C:\IBM\ID\Test\org
C:\IBM\ID\Test\org\example
C:\IBM\ID\Test\org\example\www
C:\IBM\ID\Test\org\example\www\GetQuoteFaultMsg.class
C:\IBM\ID\Test\org\example\www\OrgExampleWwwQuoteSOAPImplBindingSoapStub.class
C:\IBM\ID\Test\org\example\www\QuoteClient.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImpl.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImplService.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImplServiceLocator.class

C:\IBM\ID\Test>amqwsetcp
C:\IBM\ID\Test>java org.example.www.QuoteClient.class
Response = IBM
```

Figura 199. Configuração do cliente e de saída

Como proceder a seguir

1. Se você estiver implementando o cliente como um cliente IBM WebSphere MQ, configure o canal de conexão do cliente e do servidor.
2. Se você estiver implementando o cliente em um gerenciador de filas diferente no serviço, deve-se disponibilizar a fila de destino para o cliente. Configure a fila de destino no gerenciador de filas de serviço como uma fila de cluster ou no gerenciador de filas do cliente como uma definição de fila remota.

Tarefas relacionadas

Implementando um cliente de serviço da web no Axis2 para usar o WebSphere MQ de transporte para SOAP
Prepare um diretório de implementação e o arquivo de configuração do Axis2 para o cliente. Forneça os proxies do cliente e a classe do cliente e configure CLASSPATH. Configure filas e canais do WebSphere MQ , inicie o serviço e teste o cliente.

Implementando em um cliente Axis2 usando W3C SOAP sobre JMS
Um serviço da Web ligado à recomendação do candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos Java EE . Esta tarefa é a etapa 4 de conectar um cliente de serviço da Web Axis2 e um serviço da Web implementado no WebSphere Application Server usando o protocolo SOAP sobre JMS do W3C . Modifique a URL no cliente Axis2 desenvolvido para o transporte WebSphere MQ para SOAP para usar a recomendação do candidato W3C para SOAP sobre JMS.

Implementando um cliente de serviço da web no .NET Framework 1 e 2 para usar o transporte do WebSphere MQ para SOAP
Prepare um diretório de implementação e um descritor de implementação para o cliente. Forneça o proxy de cliente e a classe do cliente. Configure filas e canais do WebSphere MQ , inicie o serviço e teste o cliente.

Implementando um cliente de serviço da web no Axis2 para usar o WebSphere MQ de transporte para SOAP

Prepare um diretório de implementação e o arquivo de configuração do Axis2 para o cliente. Forneça os proxies do cliente e a classe do cliente e configure CLASSPATH. Configure filas e canais do WebSphere MQ , inicie o serviço e teste o cliente.

Antes de começar

Sugestão: Implemente o serviço para HTTP. Desenvolva e teste o cliente para HTTP e, em seguida, modifique a URL para fazer referência ao serviço usando o transporte WebSphere MQ para SOAP.

A tarefa mostra como implementar um cliente Axis2 não gerenciado no Java Standard Edition. Pode ser que deseje implementar um cliente do Axis2 em um contêiner da web. No “Desenvolvendo um Cliente JAX-WS para WebSphere Transport for SOAP Usando Eclipse” na página 994, você desenvolveu um cliente em um contêiner de Web e implementou-o no WebSphere Application Server Community Edition Como parte da configuração do servidor, você ativou o aspecto do Axis2 e incluiu o aspecto na configuração do contêiner da web. Para configurar contêineres da web em outros servidores de aplicativos, consulte a documentação do Axis2, http://ws.apache.org/axis2/1_4_1/installationguide.html#servlet_container ou a documentação fornecida com o servidor da web.

Nota: O Axis2 usa o termo, contêiner do Servlet. Um contêiner do Servlet é o mesmo que um contêiner da web.

Sobre esta tarefa

Implementar um cliente Axis2 para usar o transporte WebSphere MQ para SOAP é como implementar um cliente Axis2 para usar HTTP. Etapas adicionais são necessárias para fornecer um caminho de classe para os arquivos JAR do WebSphere MQ e para modificar o arquivo de configuração Axis2 . O arquivo de configuração Axis2 requer uma entrada adicional para JMS. A entrada refere-se ao transporte WebSphere MQ para o arquivo JAR SOAP que implementa o JMS `transportSender`.

O Axis2 fornece um script, `axis2.bat` ou `axis2.sh`, que simplifica a implementação do cliente; consulte os exemplos em Figura 203 na página 1029 e Figura 204 na página 1029.

Nota:

1. `axis2.bat` tem um erro que deve ser corrigido. A sequência `-Djava.ext.dirs="%AXIS2_HOME%\lib\"` deve ser mudada para `-Djava.ext.dirs="%AXIS2_HOME%\lib\\"`
2. No `axis2.bat` e `axis2.sh`, `-Djava.ext.dirs` é usado como uma maneira rápida para fazer referência a todos os arquivos JAR Axis2, em vez de incluí-los separadamente no caminho de classe.

Infelizmente esta abordagem é imperfeita e funciona apenas com alguns JREs. Ele não funciona com os JREs do IBM

O parâmetro JVM, `-Djava.ext.dirs="%AXIS2_HOME%\lib\\"`, torna os arquivos JAR Axis disponíveis para a JVM. A JVM tenta instanciar alguns dos arquivos JAR Axis e leva a um erro, os detalhes dos quais dependem da JVM. Geralmente, você poderá ver uma das linhas a seguir no rastreio de pilha:

```
org.apache.axiom.om.util.UUIDGenerator.getInitialUUID(UUIDGenerator.java:76)
```

ou `org.apache.axis2.deployment.DeploymentException: java.security.NoSuchAlgorithmException: MD5 MessageDigest not available`

A maneira correta para executar um cliente não gerenciado Axis2 é incluir os arquivos JAR do Axis2 no caminho de classe. O caminho de classe está disponível apenas para o aplicativo cliente e não para a JVM.

O procedimento descreve as etapas gerais para executar um cliente Axis2 não gerenciado sem usar o script axis2. Os exemplos em Figura 201 na página 1028 e Figura 202 na página 1028 são scripts para Windows e Linux..

Procedimento

1. Faça download do Axis2 1.4.1 em http://ws.apache.org/axis2/download/1_4_1/download.cgi e descompacte em uma pasta, `Axis2-1.4.1`.
2. Atualize `axis2.xml` em `Axis2-1.4.1\conf`.
   a) Atualize `axis2.xml` em `Axis2-1.4.1\conf`. Inclua o transporte WebSphere MQ para SOAP como um `transportSender`:

   ```
   <transportSender name="jms"
   class="com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender"/>
   ```

   b) Se necessário, altere o tamanho do conjunto de conexões do padrão de 10.

   ```
   <transportSender name="jms"
   class="com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender">
   <parameter name="ResourcePoolCapacity">20</parameter>
   </transportSender>
   ```

   `ResourcePoolCapacity` define quantas entradas de terminal de serviço são mantidas no cache. O valor deve ser pelo menos 1. Se o número de entradas do terminal em serviço exceder o tamanho do cache, as entradas serão excluídas para abrir espaço para novas entradas. O tamanho de uma entrada do terminal varia. Configure um número que seja grande o suficiente para evitar a sobrecarga de cache.

   Consulte a etapa 3 em "Desenvolvendo um Cliente JAX-WS para WebSphere Transport for SOAP Usando Eclipse" na página 994.
3. Crie um diretório *deployDir*. Nesse diretório, copie a estrutura de pasta contendo o cliente e os proxies de cliente. *deployDir* é equivalente à pasta *project*`\bin` em um projeto Java Eclipse .
4. Abra uma janela de comando no Windows, ou um shell de comando usando X Window System em sistemas UNIX and Linux , em *deployDir*.
5. Atualize o caminho de classe para incluir o diretório atual, arquivos JAR do Axis2, `com.ibm.mqjms.jar` e `com.ibm.mq.axis2.jar`.

   `com.ibm.mqjms.jar` referencia todos os outros arquivos JAR do WebSphere MQ que são necessários.
6. Use o comando **Java** para iniciar o programa cliente.

Examples

Quatro exemplos da execução de um cliente do Axis2 são listados em Figura 202 na página 1028 para Figura 204 na página 1029. Figura 200 na página 1028 mostra a saída da execução do cliente assíncrono listado em Figura 183 na página 999.

```
cd C:\IBM\ID\Workspaces\Axis2docs\StockQuoteAxis2PojoClient\bin>
runpojo soap/client/SQA2AsyncClient

HTTP Sync: 55.25
Callback constructor
Waiting for HTTP callback
Result in Callback 55.25
HTTP poll: 55.25
JMS: Sync: 55.25
Callback constructor
Waiting for JMS callback
Result in Callback 55.25
JMS poll: 55.25
Press any key to continue . . .
```

Figura 200. Saída da execução SQA2AsyncClient

```
@echo off
set AXIS2_HOME=C:\OpenSource\axis2-1.4.1
set JAVA_HOME=C:\IBM\Java50
set WMQ_HOME=C:\IBM\MQ\java\lib

setlocal EnableDelayedExpansion
set CLASSPATH=
set AXIS2_CLASS_PATH=
FOR %%c in ("%AXIS2_HOME%\lib\*.jar") DO set AXIS2_CLASS_PATH=!AXIS2_CLASS_PATH!;%%c

"%JAVA_HOME%\bin\java" -Daxis2.repo="%AXIS2_HOME%\repository"
-Daxis2.xml="%AXIS2_HOME%\conf\axis2.xml" -cp
".;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar;%AXIS2_CLASS_PATH%" %1

pause
```

Figura 201. runpojo.bat: Windows, usando um caminho de classe

```
export AXIS2_HOME=/home/OpenSource/axis2-1.4.1
export JAVA_HOME=/usr/lib/j2sdk1.5-ibm
# update classpath
AXIS2_CLASSPATH=""
for f in "$AXIS2_HOME"/lib/*.jar
do
  AXIS2_CLASSPATH="$AXIS2_CLASSPATH":$f
done
AXIS2_CLASSPATH="$AXIS2_HOME":"$JAVA_HOME/lib/tools.jar":"$AXIS2_CLASSPATH":"$CLASSPATH"
java -cp /home/alex/dev/sandbox/Soap/axis2:/opt/mqm/java/lib/com.ibm.mqjms.jar:
/opt/mqm/java/lib/com.ibm.mq.axis2.jar:$AXIS2_CLASSPATH
-Daxis2.xml=/home/alex/dev/sandbox/axis2-1.4.1/conf/axis2.xml %1
```

Figura 202. runpojo.sh: Linux, usando um caminho de classe.

```
@echo off
set AXIS2_HOME=C:\OpenSource\axis2-1.4.1
set JAVA_HOME=C:\IBM\Java50
set WMQ_HOME=C:\IBM\MQ\java\lib

%AXIS2_HOME%\bin\axis2 -cp .;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar; %1
pause
```

Figura 203. runaxis2.bat: Windows, usando axis2.bat

Nota

```
export AXIS2_HOME=/home/OpenSource/axis2-1.4.1
export JAVA_HOME=/usr/lib/j2sdk1.5-ibm

%AXIS2_HOME%\bin\axis2 -cp .;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar; %1
```

Figura 204. runaxis2.sh: Linux, usando axis2.sh

Nota

Tarefas relacionadas

Implementando um cliente de serviço da web para o Axis 1.4 para usar o transporte do IBM WebSphere MQ para SOAP

Prepare um diretório de implementação e um descritor de implementação para o cliente. Forneça os proxies do cliente e a classe do cliente e configure CLASSPATH. Configure as filas e os canais do IBM WebSphere MQ, inicie o serviço e teste o cliente.

Implementando em um cliente Axis2 usando W3C SOAP sobre JMS

Um serviço da Web ligado à recomendação do candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos Java EE . Esta tarefa é a etapa 4 de conectar um cliente de serviço da Web Axis2 e um serviço da Web implementado no WebSphere Application Server usando o protocolo SOAP sobre JMS do W3C . Modifique a URL no cliente Axis2 desenvolvido para o transporte WebSphere MQ para SOAP para usar a recomendação do candidato W3C para SOAP sobre JMS.

Implementando um cliente de serviço da web no .NET Framework 1 e 2 para usar o transporte do WebSphere MQ para SOAP

Prepare um diretório de implementação e um descritor de implementação para o cliente. Forneça o proxy de cliente e a classe do cliente. Configure filas e canais do WebSphere MQ , inicie o serviço e teste o cliente.

Implementando em um cliente Axis2 usando W3C SOAP sobre JMS

Um serviço da Web ligado à recomendação do candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos Java EE . Esta tarefa é a etapa 4 de conectar um cliente de serviço da Web Axis2 e um serviço da Web implementado no WebSphere Application Server usando o protocolo SOAP sobre JMS do W3C . Modifique a URL no cliente Axis2 desenvolvido para o transporte WebSphere MQ para SOAP para usar a recomendação do candidato W3C para SOAP sobre JMS.

Antes de começar

Você deve primeiro concluir a tarefa, “Desenvolvendo um Cliente JAX-WS para WebSphere Transport for SOAP Usando Eclipse” na página 994 para chamar **SimpleJavaListener** usando um cliente Axis2 e o transporte WebSphere MQ para o protocolo SOAP.

Você também deve ter criado o serviço da Web e configurado o WebSphere MQ e o WebSphere Application Server nas tarefas anteriores:

1. “Configure recursos do WebSphere MQ” na página 1019.
2. “Configure recursos do WebSphere Application Server” na página 1020.
3. “Desenvolvendo um Serviço da Web EJB JAX-WS para SOAP W3C sobre JMS” na página 980.

Na tarefa, o cliente é executado no Eclipse Galileo. Você pode executar o cliente a partir da linha de comandos modificando o arquivo `Axis2.bat` fornecido com o Axis2.

Sobre esta tarefa

A única mudança que você deve fazer no cliente estático Axis2 `StockQuoteAxis` existente para chamar o serviço StockQuoteAxis hospedado pelo WebSphere Application Server é alterar a URL transmitida para o cliente. Como o WSDL não foi mudado, será possível usar as mesmas classes de proxy no pacote `soap.server`.

Você tem duas abordagens para definir a URL para transmitir para o cliente. Você pode usar a mesma URL como gerada em `StockQuoteAxis.wsdl`. Deve-se incluir os parâmetros `jndiInitialContextFactory` e `jndiURL` para acessar o diretório JNDI do WebSphere Application Server. Outra abordagem é mudar a URL e fornecer o acesso direto do cliente para as filas `REQUESTAXIS` e `REPLYAXIS` em `QM1`, sem usar uma consulta JNDI.

Os parâmetros de conexão definidos na URL transmitida para o cliente Axis2 são usados para se conectar ao gerenciador de filas do WebSphere MQ e filas necessárias para enviar e receber mensagens SOAP. Os parâmetros de conexão transmitidos ao cliente Axis2 não são necessariamente usados pelo serviço. É possível usar os recursos de enfileiramento distribuído do WebSphere MQ para separar o cliente e o serviço do uso do mesmo gerenciador de filas ou do mesmo servidor de nomes.

Procedimento

1. Salve a URL do `StockQuoteAxis.wsdl` gerado e feche o Rational Application Developer para economizar na memória.

   Se você não alterou a configuração do servidor, fechar o Rational Application Developer para o servidor de aplicativos. Nesse caso, inicie o servidor com o comando:

   ```
   startserver server1
   ```

2. Abra o Eclipse Galileo na área de trabalho com o projeto do cliente do Axis2.
3. Abra `SQA2StaticClient.java`.

   Consulte SQA2StaticClient.java.
4. Chame o serviço usando a variante queue do URI.
   a) Modifique a URL.

      O novo URI é:

      ```
      jms:queue:REQUESTAXIS
                ?replyToName=REPLYAXIS
                &connectionFactory=connectQueueManager(QM1)Bind(Server)
                &targetService=StockQuoteAxis;
      ```

      Compare a esta URL do `StockQuoteAxis.wsdl`:

      ```
      jms:jndi:requestaxis
               ?jndiConnectionFactoryName=qm1
               &targetService=StockQuoteAxis
      ```

      Figura 205. URL a partir do `StockQuoteAxis.wsdl`

      - `REQUESTAXIS` está agora em maiúsculas já que ele é um nome de fila e não um nome de JNDI.
      - A conexão com `QM1` é direta.
      - O URI não contém o nome da resposta para o destino. O cliente deve definir a fila a qual ele espera as respostas.

   b) Execute `SQA2StaticClient.java` usando o mesmo **Executar como ...** como você fez na tarefa, “Desenvolvendo um Cliente JAX-WS para WebSphere Transport for SOAP Usando Eclipse” na página 994.

5. Chame o serviço usando a variante `jndi` do URI, usando o WebSphere Application Server como o servidor de nomenclatura
   a) Use a URL de `StockQuoteAxis.wsdl`, Figura 205 na página 1030, fornecendo os parâmetros ausentes para usar o serviço de nomenclatura no WebSphere Application Server.

      Os parâmetros e valores ausentes que deve-se fornecer são:

Tabela 151. Parâmetros adicionais de JNDI

Parâmetro	Valor usado neste exemplo	Descrição
`&jndiURL`	`iiop://localhost:2810` ou `corbaname:iiop:localhost:2810`	URI do provedor de nomenclatura. Para o WebSphere Application Server, o valor é padronizado para 2809. Também é conhecido como o número da porta do conector RMI e a porta de autoinicialização. O valor é listado no `SystemOut.log` `00000000 NameServerImp A NMSV0018I: Name server available on bootstrap port 2810`
`&jndiInitialContextFactory`	`com.ibm.websphere.naming.WsnInitialContextFactory`	O nome do factory de contexto inicial usado pelo WebSphere Application Server.
`&replyToName`	`replyaxis`	Nome JNDI da fila REPLYAXIS.

```
jms:jndi:requestaxis?
         &jndiURL=iiop//localhost:2810
         &jndiConnectionFactoryName=qm1
         &jndiInitialContextFactory=com.ibm.websphere.naming.WsnInitialContextFactory
         &targetService=StockQuoteAxis
         &replyToName=replyaxis;
```

   b) Inclua os arquivos JAR requeridos pela consulta JNDI.

      Nessa configuração, os seguintes arquivos JAR foram incluídos no caminho de compilação para executar a tarefa usando a variante `jndi` da URL JMS:

      - `com.ibm.jaxws.thinclient_7.0.0.jar` a partir de *`Rational install directory`*`\SDP\runtimes\base_v7\runtimes`.
      - `com.ibm.ws.runtime.jar` de *`Rational install directory`*`\SDP\runtimes\base_v7\plugins`

      Para um provedor JNDI diferente, requeira arquivos JAR diferentes.

      Os outros arquivos JAR no caminho de construção são:

      i) Todos os arquivos JAR em *`WebSphere MQ Install directory`*`\java\lib`

      ii) Todos os arquivos JAR em *`Axis2-1.5.1`*`\lib`

      iii) Java 6.0 JRE..

   c) Execute `SQA2StaticClient.java` usando o mesmo **Executar como ...** como você fez na tarefa, "Desenvolvendo um Cliente JAX-WS para WebSphere Transport for SOAP Usando Eclipse" na página 994.

Resultados

Em ambos os casos a resposta do serviço é exibida na visualização de console do cliente.

Tarefas relacionadas

Implementando um cliente de serviço da web para o Axis 1.4 para usar o transporte do IBM WebSphere MQ para SOAP
Prepare um diretório de implementação e um descritor de implementação para o cliente. Forneça os proxies do cliente e a classe do cliente e configure CLASSPATH. Configure as filas e os canais do IBM WebSphere MQ, inicie o serviço e teste o cliente.

Implementando um cliente de serviço da web no Axis2 para usar o WebSphere MQ de transporte para SOAP
Prepare um diretório de implementação e o arquivo de configuração do Axis2 para o cliente. Forneça os proxies do cliente e a classe do cliente e configure CLASSPATH. Configure filas e canais do WebSphere MQ , inicie o serviço e teste o cliente.

Implementando um cliente de serviço da web no .NET Framework 1 e 2 para usar o transporte do WebSphere MQ para SOAP
Prepare um diretório de implementação e um descritor de implementação para o cliente. Forneça o proxy de cliente e a classe do cliente. Configure filas e canais do WebSphere MQ , inicie o serviço e teste o cliente.

Implementando um cliente de serviço da web no .NET Framework 1 e 2 para usar o transporte do WebSphere MQ para SOAP

Prepare um diretório de implementação e um descritor de implementação para o cliente. Forneça o proxy de cliente e a classe do cliente. Configure filas e canais do WebSphere MQ , inicie o serviço e teste o cliente.

Antes de começar

Sugestão: Desenvolva e teste o serviço e o cliente usando o Visual Studio. Em seguida, modifique o cliente para o transporte do WebSphere MQ para SOAP

1. Se você estiver implementando um serviço usando .NET Framework 1 ou 2, construa o serviço como uma biblioteca (`.dll`). Implemente usando o transporte WebSphere MQ para SOAP.
2. Inclua a chamada Register.Extension() no cliente.
3. Inclua uma referência ao amqsoap.dll, que está localizado em *MQ_Install*`\bin`
4. Altere a propriedade `Url` estática no construtor de classe de proxy do cliente para o URI `jms:/` , para o transporte WebSphere MQ para SOAP

Sobre esta tarefa

Implementar um cliente de serviço da web para o .NET Framework 1 ou 2 para usar o transporte WebSphere MQ para SOAP requer uma etapa de implementação adicional. É necessário registrar o `amqsoap.dll` com o .NET Framework `amqsoap.dll` é registrado automaticamente como parte da instalação do transporte do WebSphere MQ para SOAP, mas pode ser necessário registrá-lo novamente

Se você usar o comando **amqwdeployWMQService** para gerar proxies de cliente, será possível implementar o cliente usando os diretórios que o comando gera.

Procedimento

1. Crie um diretório *deployDir* para manter os arquivos de implementação do cliente.
2. Abra uma janela de comando em *deployDir*.
3. Execute **amqwsetcp** para configurar o CLASSPATH se o serviço for ser executado no Axis 1.4.
4. Se necessário, execute **amqwRegisterDotNet** para registrar `amqsoap.dll` no .NET Framework.

exemplo

O exemplo mostra a configuração e a saída, Figura 208 na página 1033, de um cliente do .NET Framework V2 . O cliente, Figura 207 na página 1033, chama um serviço da web que ecoa seu parâmetro de entrada. A definição da Url estática, Figura 206 na página 1033, mostra o construtor para o proxy de cliente.

```
public Quote() {
    this.Url = "jms:/queue?destination=REQUESTDOTNET
               &connectionFactory=(connectQueueManager(QM1)binding(server))
               &initialContextFactory=com.ibm.mq.jms.Nojndi
               &targetService=Quote.asmx
               &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE";
}
```

Figura 206. Construtor de proxy de cliente estático

```
using System;
namespace QuoteClientProgram {
    class QuoteMain {
        static void Main(string[] args) {
            try {
                IBM.WMQSOAP.Register.Extension();
                Quote q = new Quote();
                Console.WriteLine("Response is: " + q.getQuote("ibm"));
            } catch (Exception e) {
                Console.WriteLine("Exception is: " + e);
            }
        }
    }
}
```

Figura 207. Programa cliente

```
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram>dir /s /b
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram\QuoteClientProgram.exe
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram>quoteclientprogram
Response is: IBM
```

Figura 208. Configuração e saída

Como proceder a seguir

1. Se você estiver implementando o cliente como um cliente MQI do WebSphere MQ , configure o canal de conexão do cliente e do servidor.
2. Se você estiver implementando o cliente em um gerenciador de filas diferente no serviço, deve-se disponibilizar a fila de destino para o cliente. Configure a fila de destino no gerenciador de filas de serviço como uma fila de cluster ou no gerenciador de filas do cliente como uma definição de fila remota.

Tarefas relacionadas

Implementando um cliente de serviço da web para o Axis 1.4 para usar o transporte do IBM WebSphere MQ para SOAP

Prepare um diretório de implementação e um descritor de implementação para o cliente. Forneça os proxies do cliente e a classe do cliente e configure CLASSPATH. Configure as filas e os canais do IBM WebSphere MQ, inicie o serviço e teste o cliente.

Implementando um cliente de serviço da web no Axis2 para usar o WebSphere MQ de transporte para SOAP

Prepare um diretório de implementação e o arquivo de configuração do Axis2 para o cliente. Forneça os proxies do cliente e a classe do cliente e configure CLASSPATH. Configure filas e canais do WebSphere MQ , inicie o serviço e teste o cliente.

Implementando em um cliente Axis2 usando W3C SOAP sobre JMS
Um serviço da Web ligado à recomendação do candidato W3C para SOAP sobre JMS deve ser executado no contêiner EJB de um servidor de aplicativos Java EE . Esta tarefa é a etapa 4 de conectar um cliente de serviço da Web Axis2 e um serviço da Web implementado no WebSphere Application Server usando o protocolo SOAP sobre JMS do W3C . Modifique a URL no cliente Axis2 desenvolvido para o transporte WebSphere MQ para SOAP para usar a recomendação do candidato W3C para SOAP sobre JMS.

Conecte um cliente Axis2 a um serviço JAX-WS usando W3C SOAP sobre JMS e WebSphere Application Server

Ao concluir esta tarefa, você terá chamado um serviço da Web JAX-WS em execução no WebSphere Application Server a partir de um cliente Axis2 ... O cliente Axis2 e o WebSphere Application Server usam a recomendação do candidato W3C para o protocolo SOAP sobre JMS em execução no WebSphere MQ. Use Eclipse Galileo e Rational Application Developer para construir o cliente de serviço da web e o serviço da web, respectivamente.

Antes de começar

A tarefa requer a versão 7 do Rational Software Development Environment e do WebSphere Application Server A tarefa foi criada usando o Rational Application Developer empacotado com Rational Software Architect para WebSphere Software v7.5.5.1e WebSphere Application Server Versão 7.0 Test Environment v7.0.0.9 Atualização 1. Você também precisa do WebSphere MQ v7.0.1.3.

A tarefa baseia-se em duas outras tarefas, “Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse” na página 973e “Desenvolvendo um Cliente JAX-WS para WebSphere Transport for SOAP Usando Eclipse” na página 994 Para concluir essas tarefas, seu ambiente de desenvolvimento já possui Eclipse Galileo, WASCE, o plug-in Eclipse para WASCE e Axis2 1.4.1 instalado. Você não requer WASCE para esta tarefa.

Algumas das etapas são complexas. As etapas assumem alguma familiaridade com o desenvolvimento de aplicativos de serviços da web para o WebSphere Application Server usando o Rational Application Developer. As demandas do processador e da memória da tarefa são grandes. A tarefa foi executada em uma máquina virtual VMWare Windows XP SP3 alocada 1.8GB de memória..

Instale todo o software antes de iniciar a tarefa. O software leva cerca de um dia para fazer download e um dia para instalar, dependendo da largura de banda. A tarefa leva pelo menos meio dia.

Sobre esta tarefa

O cenário para essa tarefa é que você desenvolveu um serviço da web de cotação de ações, `StockQuoteAxis`, usando uma ferramenta de software livre, Eclipse Galileo. `StockQuoteAxis` é implementado usando SOAP sobre HTTP em execução em um servidor de software livre, WASCE.

Você deseja ligar os serviços da Web implementados a um transporte de sistema de mensagens baseado em padrões, como SOAP sobre JMS, ou ao sistema de mensagens confiável de serviços da web, bem como SOAP sobre HTTP. Você deseja que o cliente e o serviço usem interfaces baseadas em padrões. Por essa razão, embora sua futura equipe de desenvolvimento de projetos tenha implementado uma solução usando o transporte WebSphere MQ para SOAP, você não entrou em produção.

O cliente Axis2 removeu o problema de que o cliente SOAP para o transporte WebSphere MQ para SOAP precisava de uma mudança do cliente HTTP. O problema ainda permanecia que o serviço conectado pelo transporte IBM WebSphere MQ para SOAP é hospedado por um listener especial fornecido pelo WebSphere MQ: `SimpleJavaListener`.

Com o padrão SOAP sobre JMS W3C no status de recomendação candidato, alguns fornecedores estão fornecendo suporte para SOAP sobre JMS W3C . O suporte permite implementar um serviço da web em um servidor de aplicativos e se conectar ao mesmo serviço usando uma variedade de protocolos de

conectividade. O suporte fornecido pelo WebSphere Application Server v7 remove o problema de ter que hospedar o serviço da Web separadamente para usar um transporte SOAP baseado em mensagens O uso de uma interface de transporte de mensagens baseada em padrões, JMS, significa que é possível desenvolver soluções usando ferramentas de diferentes fornecedores... Você espera que as ferramentas de serviços da Web no Eclipse incluam as ligações SOAP sobre JMS no futuro

A maioria das etapas é executada usando o Eclipseou as ferramentas de gerenciamento fornecidas com os produtos WebSphere As etapas são descritas para um ambiente Windows . Com pequenas modificações para alguns comandos, é possível executar as etapas em outras plataformas.

A criação do serviço da web HTTP pelas etapas preliminares e a conexão a ele usando Axis2 são listadas. O cliente e WSDL, a partir destas etapas, são usados para criar a solução

Procedimento

1. Conecte-se ao serviço da web StockQuoteAxis usando um cliente do Axis2 e o IBM WebSphere MQ Transport for SOAP
   a) “Desenvolvendo um serviço JAX-RPC para o transporte do WebSphere MQ para SOAP usando Eclipse” na página 973
   b) “Desenvolvendo um Cliente JAX-WS para WebSphere Transport for SOAP Usando Eclipse” na página 994
   c) “Implementando um cliente de serviço da web no Axis2 para usar o WebSphere MQ de transporte para SOAP” na página 1026
2. Conecte-se ao serviço da web StockQuoteAxis usando um cliente Axis2 e a recomendação do candidato W3C para SOAP sobre JMS.
   a) “Configure recursos do WebSphere MQ” na página 1019
   b) “Configure recursos do WebSphere Application Server” na página 1020
   c) “Desenvolvendo um Serviço da Web EJB JAX-WS para SOAP W3C sobre JMS” na página 980
   d) “Implementando em um cliente Axis2 usando W3C SOAP sobre JMS” na página 1029

WebSphere MQ ponte para HTTP

Com a ponte WebSphere MQ para HTTP, os aplicativos clientes podem trocar mensagens com o WebSphere MQ sem a necessidade de instalar um cliente MQI do WebSphere MQ . É possível chamar o WebSphere MQ de qualquer plataforma ou idioma com recursos HTTP.

Introdução à ponte do WebSphere MQ para HTTP (Protocolo de Transporte de Hipertexto)

A ponte do WebSphere MQ para HTTP é um aplicativo da Web Java, Enterprise Environment (JEE). Os clientes HTTP podem enviar solicitações **POST**, **GET**e **DELETE** para ele colocar, navegar e excluir mensagens de filas do WebSphere MQ . A ponte do WebSphere MQ para HTTP não é adequada para uso com mensagens, se a entrega assegurada for necessária

Benefícios

Com a ponte do WebSphere MQ para HTTP, é possível enviar e receber mensagens do WebSphere MQ usando HTTP de uma ampla variedade de ambientes:

- Ambientes que suportam HTTP, mas não WebSphere MQ.
- Ambientes que têm espaço de armazenamento insuficiente para instalar um cliente MQI do WebSphere MQ .
- Ambientes que são muitos para instalar o cliente MQI do WebSphere MQ em cada sistema que requer acesso ao WebSphere MQ.
- Aplicativos baseados na web dos quais você deseja enviar ou receber mensagens sem codificar sua própria ponte para o WebSphere MQ.

- Aplicativos baseados na web que você deseja aprimorar, usando técnicas assíncronas, como AJAX. O WebSphere MQ Bridge for HTTP disponibiliza filas e tópicos do WebSphere MQ usando Representation State Transfer (REST) sobre HTTP.

O suporte a HTTP pode ser usado com topologias de sistema de mensagens ponto a ponto e de publicar/assinar.

Como o suporte a HTTP funciona?

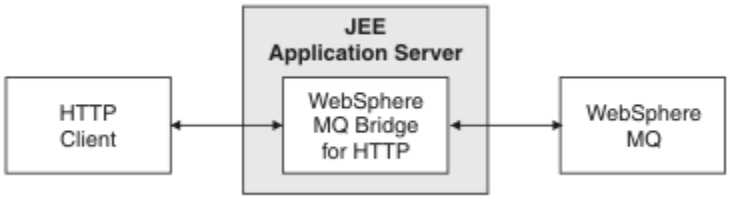


Figura 209. WebSphere MQ ponte para HTTP

A ponte do WebSphere MQ para o aplicativo da web HTTP recebe solicitações de HTTP de um ou mais clientes Ele interage com WebSphere MQ em seu nome e retorna respostas HTTP para eles.

A ponte do WebSphere MQ para HTTP é um servlet JEE conectado ao WebSphere MQ usando um adaptador de recursos. O servlet HTTP manipula três tipos diferentes de solicitações de HTTP: **POST**, **GET** e **DELETE**.

Tabela 152. Ponte do WebSphere MQ para verbos HTTP

Solicitação de HTTP	**Resultado**
POST	Coloca uma mensagem em uma fila ou tópico.
RECEBER	Procura a primeira mensagem em uma fila. Alinhado ao protocolo HTTP, **GET** não exclui a mensagem da fila. Não use **GET** com mensagens de publicar/assinar.
EXCLUIR	Obtém e exclui uma mensagem de uma fila ou tópico.

Exemplo de HTTP POST

HTTP **POST** coloca uma mensagem em uma fila, ou uma publicação para um tópico. A amostra do Java **HTTPPOST** é um exemplo de uma solicitação de HTTP **POST** de uma mensagem para uma fila. Em vez de usar Java, seria possível criar uma solicitação de HTTP **POST** usando um formulário do navegador ou um kit de ferramentas AJAX em vez disso.

Figura 210 na página 1036 mostra uma solicitação HTTP para colocar uma mensagem em uma fila chamada `myQueue`. Essa solicitação contém o cabeçalho de HTTP `x-msg-correlId` para configurar o ID de correlação da mensagem WebSphere MQ .

```
POST /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
Content-Type: text/plain
x-msg-correlID: 1234567890
Content-Length: 50

Here is my message body that is posted on the queue.
```

*Figura 210. Exemplo de uma solicitação de HTTP **POST** para uma fila*

Figura 211 na página 1037 mostra a resposta enviada de volta ao cliente. Não há conteúdo de resposta.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 0
```

Figura 211. Exemplo de uma resposta de HTTP POST

Exemplo de HTTP DELETE

HTTP **DELETE** recebe uma mensagem de uma fila e exclui a mensagem, ou recupera e exclui uma publicação. A amostra do Java **HTTPDELETE** é um exemplo de uma solicitação de HTTP **DELETE** lendo uma mensagem de uma fila. Em vez de usar Java, seria possível criar uma solicitação de HTTP **DELETE** usando um formulário do navegador ou um kit de ferramentas AJAX em vez disso.

Figura 212 na página 1037 é uma solicitação HTTP para excluir a próxima mensagem na fila chamada `myQueue`. Em resposta, o corpo da mensagem é retornado ao cliente. Nos termos do WebSphere MQ , o HTTP **DELETE** é um get destrutivo

A solicitação contém o cabeçalho de solicitação de HTTP `x-msg-wait`, que instrui a ponte do WebSphere MQ para HTTP quanto tempo esperar até que uma mensagem chegue na fila. A solicitação também contém o cabeçalho da solicitação `x-msg-require-headers`, que especifica que o cliente deve receber o ID de correlação da mensagem na resposta.

```
DELETE /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Figura 212. Exemplo de uma solicitação de HTTP **DELETE***

Figura 213 na página 1037, é a resposta retornada para o cliente O ID de correlação é retornado ao cliente, conforme solicitado no `x-msg-require-headers` da solicitação.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that is retrieved from the queue.
```

*Figura 213. Exemplo de uma resposta HTTP **DELETE***

Exemplo de HTTP GET

HTTP **GET** recebe uma mensagem de uma fila. A mensagem permanece na fila. Em termos do WebSphere MQ , HTTP **GET** é uma solicitação de procura Seria possível criar uma solicitação de HTTP **GET** usando um cliente Java, um formulário do navegador ou um kit de ferramentas AJAX.

Figura 214 na página 1038 é uma solicitação de HTTP para procurar a próxima mensagem na fila chamada `myQueue`

A solicitação contém o cabeçalho de solicitação de HTTP `x-msg-wait`, que instrui a ponte do WebSphere MQ para HTTP quanto tempo esperar até que uma mensagem chegue na fila. A solicitação também contém o cabeçalho da solicitação `x-msg-require-headers`, que especifica que o cliente deve receber o ID de correlação da mensagem na resposta.

```
GET /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Figura 214. Exemplo de uma solicitação de HTTP **GET***

Figura 215 na página 1038 é a resposta retornada ao cliente. O ID de correlação é retornado ao cliente, conforme solicitado no `x-msg-require-headers` da solicitação.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that appears on the queue.
```

*Figura 215. Exemplo de uma resposta HTTP **GET***

Instalando, configurando e verificando a ponte do WebSphere MQ para HTTP

Obtenha a ponte do WebSphere MQ para HTTP instalando o "Java Messaging and Web Services" a partir dos materiais de instalação do cliente ou do servidor MQI do WebSphere MQ . Implemente a ponte do WebSphere MQ para HTTP em um servidor de aplicativos adequado

Antes de começar

Verifique os produtos com pré-requisitos em Requisitos do sistema para IBM WebSphere MQ. O processo de instalação não verifica a presença e disponibilidade do software obrigatório para executar a ponte WebSphere MQ para HTTP. Deve-se verificar se os pré-requisitos estão instalados.

A ponte do WebSphere MQ para HTTP é executada em qualquer servidor de aplicativos compatível com o Java EE 1.4 , instalando o adaptador de recursos WebSphere MQ . Também é possível executar a ponte do WebSphere MQ para HTTP em uma liberação do WebSphere Application Server anterior à versão 6.0.2.1 Use o WebSphere Application Server Message Listener Port (MLP) para integrar o WebSphere MQ como o provedor JMS.

O suporte para a ponte do WebSphere MQ para HTTP é fornecido apenas para os seguintes servidores de aplicativos:

- WebSphere Application Server 6.0.2.1 e posterior.
- WebSphere Application Server Community Edition Versão 1.1 e mais recente

Sobre esta tarefa

WebSphere MQ ponte para HTTP é fornecido como um arquivo `.war` , `WMQHTTP.war`.

- Em plataformas UNIX e Linux,
  - `WMQHTTP.war` é incluído como parte da opção de instalação do "Java Messaging and Web Services" A opção está disponível nos materiais de instalação de servidor e clientes.
  - `WMQHTTP.war` é instalado no *<mqmtop>*`/java/http/WMQHTTP.war`. < *mqmtop*> é o diretório no qual o WebSphere MQ está instalado
  - `WMQHTTP.samples` é instalado no *<mqmtop>*`/java/http/samples`. < *mqmtop*> é o diretório no qual o WebSphere MQ está instalado

Execute as etapas de instalação a seguir para instalar a ponte do WebSphere MQ para HTTP, implementar e configurá-la e verificar a configuração. Os detalhes das etapas de configuração variam em diferentes servidores de aplicativos. Use o "Implementando e verificando a ponte do WebSphere MQ para HTTP no WebSphere Application Server V6.1.0.9" na página 1039 como um modelo para as etapas a seguir em seu servidor de aplicativos.

Procedimento

1. Obtenha `WMQHTTP.war` instalando o cliente ou servidor MQI do WebSphere MQ .
2. Copie `WMQHTTP.war` para um servidor a partir do qual ele pode ser implementado em um servidor de aplicativos.
3. Implemente `WMQHTTP.war` em um servidor de aplicativos.
4. Se necessário, instale o WebSphere MQ como um adaptador de recursos em seu servidor de aplicativos.

   Descubra se o WebSphere MQ já está configurado como um provedor de sistemas de mensagens no servidor de aplicativos. Use a ferramenta de administração ou de gerenciamento fornecida com seu servidor de aplicativos para procurar o WebSphere MQ WebSphere MQ pode ser localizado no caminho a seguir, **Recursos> JMS> Provedores de mensagens**.
5. Configure um connection factory no servidor de aplicativos para se conectar a um gerenciador de filas que usa o transporte do cliente MQI do WebSphere MQ[12].
6. Configure o aplicativo da web `WMQHTTP.war` no servidor de aplicativos para usar a connection factory
7. Verifique a configuração.
   a) Configure o gerenciador de filas denominado na connection factory e uma fila local.
   b) Coloque uma mensagem na fila local.
   c) Crie o canal de conexão do servidor denominado na connection factory, com a autoridade para ler e gravar a fila local.
   d) Inicie o gerenciador de filas e o listener.
   e) Inicie o servidor de aplicativos e `WMQHTTP.war`, se eles ainda não estiverem em execução.
   f) Abra um navegador e digite `http://`*`hostname`*`:`*`web port`*`/`*`Context root`*`/msg/queue/`*`local queue`*

Resultados

A janela do navegador exibe a mensagem colocada na fila local.

Como proceder a seguir

1. Tente o exemplo, "Implementando e verificando a ponte do WebSphere MQ para HTTP no WebSphere Application Server V6.1.0.9" na página 1039.
2. Execute os aplicativos Java de HTTP de amostra

Implementando e verificando a ponte do WebSphere MQ para HTTP no WebSphere Application Server V6.1.0.9

Use o exemplo a seguir para preparar uma implementação da ponte WebSphere MQ para HTTP para executar os programas HTTP Java de amostra. A implementação está no WebSphere Application Server V6.1.0.9..

[12] Inicialmente, pelo menos, configure o transporte do cliente Alguns servidores de aplicativos podem se conectar ao WebSphere MQ usando conexões diretas ou de modo de ligação.

Antes de começar

1. Siga as instruções em "Instalando, configurando e verificando a ponte do WebSphere MQ para HTTP" na página 1038, para copiar `WMQHTTP.war` em um servidor acessível para sua instalação do WebSphere Application Server
2. Configure um gerenciador de filas, e uma fila, para usar para testar a configuração:
   - No exemplo, o gerenciador de filas é configurado como usando os valores em Tabela 153 na página 1040:

Tabela 153. Configuração do Gerenciador de Filas	
Object	**Value**
Nome do host	`itso-01`
Gerenciador de Filas	`QM1`
Fila local	`HTTPTESTQ`
Canal de conexão do servidor	`MYSVRCON`. Configure um ID do usuário do MCA com autoridade suficiente para ler e gravar em `HTTPTESTQ`.
Porta listener	1414

3. Inicie o gerenciador de filas e o listener
4. Coloque uma mensagem de teste em `HTTPTESTQ`. Por exemplo:
   a. Inicie o WebSphere MQ Explorer.
   b. Na lista de filas locais para QM1, clique com o botão direito em **HTTPTESTQ > Colocar mensagem de teste > digite `First Message` > Colocar mensagem > Fechar**
5. Inicie o servidor de aplicativos e efetue sign on no Integrated Solutions Console.

Sobre esta tarefa

O exemplo mostra as etapas a serem executadas se estiver executando o WebSphere Application Server V6.1.0.9 como seu servidor de aplicativos. Se estiver executando uma versão diferente do WebSphere Application Server ou executando um servidor de aplicativos diferente, as etapas serão diferentes. O WebSphere Application Server V6.1.0.9 é pré-configurado com o WebSphere MQ instalado como um provedor de mensagens, usando as bibliotecas do cliente MQI do WebSphere MQ . Se o WebSphere MQ não for pré-configurado como um provedor de sistemas de mensagens ou se você desejar usar as ligações do servidor WebSphere MQ , será necessário instalar e configurar o adaptador de recursos do WebSphere MQ para JEE no servidor de aplicativos.

Siga as instruções para implementar a ponte do WebSphere MQ para HTTP no WebSphere Application Server V6.1.0.9e verifique a implementação usando um navegador:

Procedimento

1. Na área de janela de navegação, clique em **Recursos> Provedores JMS> Provedor de sistema de mensagens WebSphere MQ**

   É possível configurar no nível de Nó, Célula ou Servidor, dependendo da implementação do WebSphere Application Server. O exemplo usa a implementação no nível do Servidor.
2. Em **Propriedades adicionais**, clique em **Connection factories > Novo**.
3. No formulário de provedores JMS, forneça as informações no Tabela 154 na página 1041ou alternativas de sua escolha, clique em **Aplicar> Salvar**.

Tabela 154. Configure ou modifique os campos a seguir	
Campo	**Value**
Nome	`WMQHTTPBridge`
Nome JNDI	`jms/WMQHTTPJCAConnectionFactory`
Gerenciador de Filas	`QM1`
Host	`itso-01`
Port	`1414`
Canal	`MYSVRCON`
Tipo de transporte	`CLIENTE`

4. Na área de janela de navegação, clique em **Aplicativos > Instalar novo aplicativo**.
5. Insira o caminho para `WMQHTTP.war` no formulário e forneça uma raiz de contexto, clique em **Avançar**.
   a) A raiz de contexto é opcional. `mq` é a raiz de contexto padrão para os aplicativos HTTP de amostra.
   b) A raiz de contexto faz parte do URI que identifica a ponte WebSphere MQ para HTTP. É possível omitir a raiz de contexto ou mudá-la posteriormente.
6. Na página **Selecionar opções de instalação** do assistente de instalação, não é necessário mudar nenhum dos padrões, clique em **Avançar**.
7. Na página **Mapear módulos para servidores** , selecione um Cluster ou Servidor, marque a caixa Selecionar, clique em **Aplicar> Avançar**.
8. Na página **Mapear Referências de Recursos para Recursos** , no formulário **javax.jms.ConnectionFactory** , clique em **Procurar ...** na ponte IBM WebSphere MQ para a linha HTTP.
9. Na página **Aplicativos corporativos > Recursos disponíveis**, selecione **WMQHTTPBridge**, clique em **Aplicar**.
10. De volta ao formato **javax.jms.ConnectionFactory**, selecione o método de autenticação.
    a) Para o exemplo, escolha **Nenhum**, clique em **Aplicar**. As outras opções requerem configuração adicional.
11. Marque a caixa de seleção **Selecionar** para IBM WebSphere MQ ponte para HTTP, clique em **Avançar> Avançar> Concluir> Salvar**
12. Na área de janela de navegação, clique em **Aplicativos > Aplicativos corporativos**.
13. Marque a caixa de seleção para `WMQHTTP.war`, clique em **Iniciar**.
14. Abra a janela do navegador. Digite `http://itso-01:9080/mq/msg/queue/HTTPTESTQ`, usando o nome do host e a porta apropriados.

Resultados

A janela do navegador exibirá `First Message`, se a configuração for bem-sucedida.

Como proceder a seguir

Execute os aplicativos Java de HTTP de amostra

Publicar / assinar usando a ponte do WebSphere MQ para HTTP

A ponte do WebSphere MQ para HTTP usa as classes WebSphere MQ para a interface de publicação / assinatura JMS. HTTP **POST** cria uma publicação. HTTP **DELETE** cria uma assinatura gerenciada não durável. Deve-se configurar a publicação / assinatura para JMS antes de usar o URI do tópico

A publicação / assinatura é totalmente integrada ao WebSphere MQ na versão 7 Antes da versão 7, um broker de publicação / assinatura separado manipulava publicações e assinaturas.. Ele é chamado de publicação / assinatura "enfileirada" , para distingui-lo da publicação / assinatura totalmente integrada na versão 7 A Versão 7 emula a publicação / assinatura enfileirada usando a publicação / assinatura integrada. A emulação permite que aplicativos de publicação/assinatura enfileirada existentes coexistam com aplicativos integrados em execução no mesmo gerenciador de filas. Os aplicativos de publicação/ assinatura enfileirada também podem interoperar com aplicativos integrados, compartilhando os mesmos tópicos. Na versão 6, o broker foi fornecido com o WebSphere MQ; antes da versão 6, ele estava disponível como um SupportPack

Configuração

A ponte WebSphere MQ para HTTP usa a interface JMS para publicar e assinar. Na versão 7, é possível controlar se as classes WebSphere MQ para JMS usam publicação / assinatura enfileirada ou integrada, usando a propriedade JMS `PROVIDERVERSION` .

Uma consideração adicional é que é possível usar bibliotecas do cliente MQI do WebSphere MQ com ponte WebSphere MQ para HTTP ou bibliotecas do servidor As bibliotecas do cliente Versão 6 suportam apenas publicação / assinatura enfileirada, enquanto as bibliotecas Versão 7 suportam publicação / assinatura enfileirada e integrada. A maioria dos servidores da Web ou de aplicativos que usam o WebSphere MQ como um provedor de sistemas de mensagens faz isso usando bibliotecas do cliente.. Para usar a publicação / assinatura integrada, as bibliotecas do cliente MQI e do servidor do WebSphere MQ devem estar pelo menos na versão 7 Se uma delas estiver executando uma versão anterior do WebSphere além de 7, você deverá configurar a publicação / assinatura enfileirada; consulte Tabela 155 na página 1042. Verifique quais bibliotecas estão instaladas ou configuradas com o servidor da web ou servidor de aplicativos que você está usando.

Tabela 155. Modos de configuração de publicação/assinatura

	Cliente V6 ou anterior	**Cliente V7 ou posterior**
Servidor V6 ou anterior	1. Execute o script `\java\bin\MQJMS_PSQ.mqsc`	Não Suportado
Servidor V7 ou posterior	1. Execute o script `\java\bin\MQJMS_PSQ.mqsc` 2. Configure o gerenciador de filas como `PSMODE=ENABLED`	1. Se PROVIDERVERSION = 7 a. Configure o gerenciador de filas como `PSMODE=ENABLED` ou `PSMODE=COMPAT` 2. If PROVIDERVERSION = 6 a. Configure o gerenciador de filas como `PSMODE=ENABLED`

Publicar

Envie uma solicitação de HTTP **POST** com o URI:

```
http://hostname:port/context_root/msg/topic/topicString
```

O conteúdo da mensagem é publicado usando a sequência de tópicos *topicString*.

Assinar

Envie uma solicitação de HTTP **DELETE** com o URI:

```
http://hostname:port/context_root/msg/topic/topicString
```

A ponte do WebSphere MQ para HTTP cria uma assinatura não durável gerenciada para a sequência de tópicos *topicString* A assinatura é excluída assim que uma publicação é retornada ou até que o intervalo de espera configurado pelo cabeçalho de entidade customizado, `x-msg-wait`, expire.

Executando a ponte do WebSphere MQ para amostras HTTP

A ponte do WebSphere MQ para amostras de HTTP estão disponíveis para uso apenas no sistema operacional Windows As amostras mostram como enviar comandos HTTP **POST** e HTTP **DELETE** para a ponte WebSphere MQ para HTTP de programas Java.

Antes de começar

Verifique sua ponte do WebSphere MQ para instalação HTTP executando a etapa "7" na página 1039 em "Instalando, configurando e verificando a ponte do WebSphere MQ para HTTP" na página 1038.

As amostras de HTTP são instaladas nos diretórios mostrados em Tabela 156 na página 1043. Em cada caso, o código-fonte é instalado no subdiretório `/src`.

Tabela 156. Local de amostras HTTP

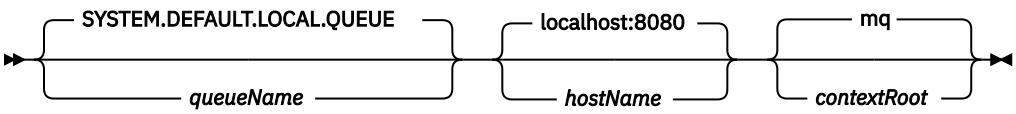
Plataforma	**Local**
Windows	*MQ_INSTALLATION_PATH*/tools/http/samples
Todas as outras plataformas	*MQ_INSTALLATION_PATH*/samp/http
MQ_INSTALLATION_PATH representa o diretório onde o WebSphere MQ está instalado.	

Sobre esta tarefa

As amostras simulam os aplicativos de amostra AMQSPUT e AMQSGET do WebSphere MQ . Eles ilustram as funções a seguir em um ambiente de sistema de mensagens ponto a ponto:

- **HTTPPOST** -Envia solicitações de HTTP **POST** em um aplicativo Java para colocar mensagens em uma fila do WebSphere MQ , usando a ponte do WebSphere MQ para HTTP e manipulando as respostas
- **HTTPDELETE** -Envia solicitações de HTTP **DELETE** em um aplicativo Java para obter mensagens de uma fila do WebSphere MQ , usando a ponte do WebSphere MQ para HTTP e manipula as respostas que contêm a mensagem do WebSphere MQ

Parâmetros para HTTPPOST e HTTPDELETE



Para executar a amostra **HTTPPOST**, conclua as etapas a seguir:

Procedimento

1. Em uma janela de comandos, navegue até o diretório de amostras HTTP.
2. Execute a amostra **HTTPPOST**.

   ```
   java -classpath . HTTPPOST [parameters]
   ```

   Quando a amostra **HTTPPOST** iniciar, a saída a seguir será exibida:

```
HTTP POST Sample start
Target server is 'hostName'
Target queue is 'queueName'
Target context-root is 'contextRoot'
```

3. No prompt de comandos, digite o texto que deseja para formar o corpo de sua mensagem.
4. Pressione Enter para postar a mensagem na fila WebSphere MQ .
   a) Se desejar enviar outra mensagem, insira mais algum texto.
      Os textos formam o corpo de uma segunda mensagem WebSphere MQ .
   b) Pressione Enter para postar a mensagem na fila WebSphere MQ .
5. Pressione Enter duas vezes para terminar **HTTPPOST**.
   A seguinte saída é exibida:

```
HTTP POST Sample end
```

Como proceder a seguir

A amostra **HTTPDELETE** executa um get destrutivo de todas as mensagens colocadas na fila do WebSphere MQ .

Execute a amostra **HTTPDELETE** concluindo as etapas a seguir:

1. Em uma janela de comandos, navegue até *MQ_INSTALLATION_PATH*/tools/samples.*MQ_INSTALLATION_PATH* representa o diretório onde o WebSphere MQ está instalado.
2. Execute a amostra **HTTPDELETE**.

```
java -classpath . HTTPPOST [parameters]
```

   Quando a amostra **HTTPDELETE** iniciar, a saída a seguir será exibida:

```
HTTP DELETE Sample start
Target server is 'host:port'
Target queue is 'your queue name'
Target context-root is 'your context-root'
message
message
...
```

Considerações de segurança para a ponte do WebSphere para HTTP

As contraprestações de segurança da web padrão se aplicam a autenticação de um cliente do navegador da web. A autorização para os recursos do WebSphere MQ está no nível do usuário que está executando o WebSphere Bridge para servlet HTTP, e não o cliente do navegador da Web individual A consideração de segurança padrão do WebSphere MQ aplica-se ao WebSphere MQ

Os dados que fluem de um navegador da web para um aplicativo WebSphere MQ usando a ponte WebSphere para HTTP e, de volta, executa três etapas:

Conexão do cliente
Do navegador para o WebSphere Bridge for HTTP sobre uma conexão TCP/IP usando HTTP.

Conexão do adaptador de recursos para o WebSphere MQ
A conexão é do WebSphere Bridge for HTTP para um gerenciador de filas WebSphere MQ . A conexão é uma conexão do cliente, sobre TCP/IP, ou uma conexão de ligações locais do WebSphere MQ local. Quando a conexão for estabelecida, a solicitação de HTTP será colocada em uma fila local padrão ou uma fila de transmissão.

Da fila local do WebSphere MQ sobre um ou mais canais, para a fila de destino.
Aplique as técnicas padrão para proteger filas, tópicos, gerenciadores de filas e canais.

A resposta usa as etapas em reverso.

Conexão do cliente

Conexões seguras entre clientes HTTP e o servidor de aplicativos usando o contêiner da web. Use as técnicas do servidor HTTP padrão, como ao usar HTTPS. Consulte a documentação de seu servidor de aplicativos para obter informações.

Conexão do adaptador de recursos para o WebSphere MQ

A conexão entre o adaptador de recursos e o gerenciador de filas é autorizada usando apenas um único ID do usuário. Designe um único ID do usuário para identificar solicitações do WebSphere Bridge for HTTP. O ID do usuário deve ter autorizações restritas do WebSphere MQ somente para os recursos que usuários externos devem ter acesso. Deve-se autenticar o cliente real separadamente e estabelecer a confiança para interações sucessivas com o cliente, usando técnicas padrão para segurança da web.

Proteja a conexão entre o adaptador de recursos e o gerenciador de filas usando o ID de usuário único. Ao restringir as autoridades, o ID do usuário não terá mais do que necessário para ler e gravar mensagens em filas e tópicos. O WebSphere Bridge for HTTP é um ponto de ataque entre a Internet e a sua intranet

Como você protege a conexão entre seu adaptador de recursos e o WebSphere MQ é dependente de seu adaptador de recursos específico Consulte a documentação para o adaptador de recursos.

Usando o Component Object Model Interface (WebSphere MQ Classes de Automação para ActiveX)

O WebSphere MQ Automation Classes para ActiveX (MQAX) são componentes ActiveX que fornecem classes que podem ser usadas em seu aplicativo para acessar o WebSphere MQ

O MQAX requer um ambiente WebSphere MQ e um aplicativo WebSphere MQ correspondente com o qual se comunicar.

Ele fornece ao aplicativo ActiveX a capacidade de executar transações e acessar dados em qualquer um dos sistemas corporativos que você pode acessar por meio do WebSphere MQ.

WebSphere MQ Classes de Automação para ActiveX:

- Forneça acesso às funções e aos recursos da API do WebSphere MQ , permitindo a interconectividade total com outras plataformas do WebSphere MQ
- Conformidade com as convenções normais esperadas de um componente ActiveX.
- Em conformidade com o modelo de objeto do WebSphere MQ , também disponível para .NET, C + +, Java e LotusScript

São fornecidas amostras iniciais do MQAX. É possível usar essas amostras inicialmente para verificar se sua instalação do MQAX foi bem-sucedida e se você tem o ambiente básico do WebSphere MQ em vigor. As amostras também demonstram como o MQAX pode ser usado.

Script de ActiveX e COM

O Component Object Model (COM) é um modelo de programação baseado em objeto definido pela Microsoft. Ele especifica como os componentes de software podem ser fornecidos de maneira que os permita localizar e se comunicam uns com os outros independentemente da linguagem do computador na qual estão gravados ou seus locais.

ActiveX é um conjunto de tecnologias, com base em COM, que integra desenvolvimento de aplicativos, componentes reutilizáveis e tecnologias da Internet nas plataformas Microsoft Windows . Os componentes ActiveX fornecem interfaces que podem ser acessadas dinamicamente por aplicativos. Um cliente de script ActiveX é um aplicativo, por exemplo um compilador, que pode construir ou executar um problema ou script que usa as interfaces fornecidas pelos componentes ActiveX (ou COM).

WebSphere MQ

WebSphere MQ Classes de Automação para ActiveX pode ser usado apenas com **32-bit** ActiveX clientes de script..

O componente COM pode ser usado apenas para aplicativos de **32 bits**. Se você deseja gravar o aplicativo COM de 64 bits, é possível usar a interface .NET.

Para executar o MQAX em um ambiente do servidor WebSphere MQ , deve-se ter o Windows 2000 ou posterior instalado em seu sistema.

Para executar o MQAX em um ambiente do cliente MQI do WebSphere MQ , você precisa do cliente MQI WebSphere MQ no Windows 2000 ou posterior instalado em seu sistema:

O cliente MQI do WebSphere MQ requer acesso a pelo menos um servidor do WebSphere MQ Quando o cliente MQI do WebSphere MQ e o servidor WebSphere MQ são instalados em seu sistema, os aplicativos MQAX sempre são executados no servidor. A interface ActiveX para o MQAI está disponível apenas em ambientes do servidor WebSphere MQ .

Projetando e Programando Usando Classes de Automação do WebSphere MQ para ActiveX

Projetando aplicativos MQAX que acessam aplicativos não ActiveX

As Classes de Automação WebSphere MQ fornecem acesso às funções da API WebSphere MQ . Portanto, é possível se beneficiar de todas as vantagens que o uso do WebSphere MQ pode trazer para o seu aplicativo Windows

O design geral de seu aplicativo é o mesmo de qualquer aplicativo WebSphere MQ , portanto, considere todos os aspectos de design descritos na seção “Desenvolvendo Aplicativos” na página 7 .

Para usar as Classes de Automação do WebSphere MQ , você codifique os programas Windows em seu aplicativo usando uma linguagem que suporte a criação e o uso de objetos COM Por exemplo, Visual Basic, Java e outros clientes de script ActiveX . As classes podem, então, ser facilmente integradas em seu aplicativo porque os objetos do WebSphere MQ necessários podem ser codificados usando a sintaxe nativa da linguagem de implementação.

Usando WebSphere MQ Classes de Automação para ActiveX

Ao projetar um aplicativo ActiveX que usa WebSphere MQ Classes de Automação para ActiveX, o item de informações mais importante é a mensagem enviada ou recebida do sistema WebSphere MQ remoto. Portanto, deve-se conhecer o formato dos itens inseridos na mensagem. Para um script MQAX para um trabalho, ele e o aplicativo WebSphere MQ que seleciona ou envia a mensagem devem conhecer a estrutura da mensagem.

Se você estiver enviando uma mensagem com um aplicativo MQAX e deseja executar a conversão de dados no final MQAX, deve-se também saber:

- A página de códigos usada pelo sistema remoto
- A codificação usada pelo sistema remoto

Para manter seu código portátil, é uma boa prática configurar a página de códigos e codificação, mesmo se eles estiverem atualmente o mesmo nos sistemas emissor e receptor.

Ao considerar como estruturar a implementação do sistema que você projeta, lembre-se de que seus scripts MQAX são executados na mesma máquina em que você tem o gerenciador de filas do WebSphere MQ ou o cliente do WebSphere MQ instalado.

Dicas e sugestões de programação

As dicas e sugestões a seguir não estão em nenhuma ordem significativa. Elas são assuntos que, se relevantes para o trabalho que você está realizando, podem economizar tempo.

Propriedades do Descritor de Mensagens

Se você manipular propriedades do descritor de mensagens em um programa, pode ser melhor usar equivalentes hexadecimais dos campos.

As informações nesta seção se referem às propriedades a seguir:

- AccountingToken
- CorrelationId
- GroupId
- MessageId

Onde um aplicativo WebSphere MQ é o originador de uma mensagem e o WebSphere MQ gera essas propriedades, é melhor usar as propriedades AccountingTokenHex, CorrelationIdHex, GroupIdHex e MessageIdHex se você desejar consultar seus valores ou manipulá-los, incluindo passá-los de volta em uma mensagem para o WebSphere MQ A razão para isso é que os valores gerados do WebSphere MQ são sequências de bytes que têm qualquer valor de 0 a 255 inclusivo, eles não são sequências de caracteres para impressão.

Onde seu script MQAX é o originador de uma mensagem, é possível usar as propriedades AccountingToken, CorrelationId, GroupId e MessageId ou seus equivalentes hexadecimais.

Constantes WebSphere MQ

WebSphere MQ constantes são fornecidas como membros do enum WebSphere MQ na biblioteca MQAX200.

Constantes de sequência do WebSphere MQ

WebSphere MQ constantes de sequência e suas sequências de caracteres correspondentes

WebSphere MQ constantes de sequência não estão disponíveis ao usar WebSphere MQ Classes de Automação para ActiveX. Deve-se usar a sequência de caracteres explícita para aqueles mostrados na lista a seguir e quaisquer outros que possam ser necessários. Os comandos devem ser preenchidos com oito caracteres usando espaços:

MQFMT_NONE	" "
MQFMT_ADMIN	"MQADMIN "
MQFMT_CHANNEL_COMPLETED	"MQCHCOM "
MQFMT_CICS	"MQCICS "
MQFMT_COMMAND_1	"MQCMD1 "
MQFMT_COMMAND_2	"MQCMD2 "
MQFMT_DEAD_LETTER_HEADER	"MQDEAD "
MQFMT_DIST_HEADER	"MQHDIST "
MQFMT_EVENT	"MQEVENT "
MQFMT_IMS	"MQIMS "
MQFMT_IMS_VAR_STRIsNG	"MQIMSVS "
MQFMT_MD_EXTENSION	"MQHMDE "
MQFMT_PCF	"MQPCF "
MQFMT_REF_MSG_HEADER	"MQHREF "
MQFMT_RF_HEADER	"MQHRF "
MQFMT_STRING	"MQSTR "

MQFMT_TRIGGER	"MQTRIG "
MQFMT_WORK_INFO_HEADER	"MQHWIH "
MQFMT_XMIT_Q_HEADER	"MQXMIT "

Constantes de sequência nula

As constantes WebSphere MQ , usadas para a inicialização de quatro propriedades MQMessage, MQMI_NONE (24 CARACTERES NULL), MQCI_NONE (24 CARACTERES NULL), MQGI_NONE (24 CARACTERES NULL) e MQACT_NONE (32 CARACTERES NULL), não são suportadas por WebSphere MQ Classes de Automação para ActiveX. Configurá-las para sequências vazias tem o mesmo efeito.

Por exemplo, para configurar os vários IDs de uma MQMessage com estes valores: *mymessage*.**MessageId** = "" *mymessage*.**CorrelationId** = "" *mymessage*.**AccountingToken** = ""

Recebendo uma mensagem do WebSphere MQ

Há várias maneiras de receber uma mensagem do WebSphere MQ:

- Pesquisando emitindo um GET seguido por uma Espera, usando a função do Visual Basic TIMER.
- Emitindo um GET com a opção de Espera; você especifica a duração de espera, configurando a propriedade WaitInterval. Considere isto quando, mesmo se você configurar seu sistema para executar no ambiente multiencadeado, o software em execução no momento pode executar apenas um único encadeamento. Isso evita que o sistema trave indefinidamente.

  Outros encadeamentos operam não afetados. No entanto, se seus outros encadeamentos precisarem de acesso ao WebSphere MQ, eles precisarão de uma segunda conexão com o WebSphere MQ usando o gerenciador de filas MQAX adicional e objetos de fila.

Emitir um GET com a opção de Espera e configurar o WaitInterval para MQWI_UNLIMITED faz com que o sistema trave até a conclusão da chamada GET, se o processo for um único encadeamento.

Usando a conversão de dados

Duas formas de conversão de dados são suportadas pelo WebSphere MQ Automation Classes para ActiveX -codificação numérica e conversão do conjunto de caracteres.

Codificação numérica

Se você configurar a propriedade MQMessage Encoding, os seguintes métodos serão convertidos entre diferentes sistemas de codificação numérica:

- Método ReadDecimal2
- Método ReadDecimal4
- Método ReadDouble
- Método ReadDouble4
- Método ReadFloat
- Método ReadInt2
- Método ReadInt4
- Método ReadLong
- Método ReadShort
- Método ReadUInt2
- Método WriteDecimal2
- Método WriteDecimal4
- Método WriteDouble

- Método WriteDouble4
- Método WriteFloat
- Método WriteInt2
- Método WriteInt4
- Método WriteLong
- Método WriteShort
- Método WriteUInt2

A propriedade Codificação pode ser configurada e interpretada usando as constantes WebSphere MQ fornecidas Figura 216 na página 1049 mostra um exemplo destes:

```
/* Encodings for Binary Integers */
        MQENC_INTEGER_UNDEFINED
        MQENC_INTEGER_NORMAL
        MQENC_INTEGER_REVERSED

/* Encodings for Decimals */
        MQENC_DECIMAL_UNDEFINED
        MQENC_DECIMAL_NORMAL
        MQENC_DECIMAL_REVERSED

/* Encodings for Floating-Point Numbers */
        MQENC_FLOAT_UNDEFINED
        MQENC_FLOAT_IEEE_NORMAL
        MQENC_FLOAT_IEEE_REVERSED
        MQENC_FLOAT_S390
```

Figura 216. Fornecidas constantes do WebSphere MQ para codificação

Por exemplo, para enviar um número inteiro de um sistema Intel para um sistema operacional System/390 na codificação System/390 :

```
Dim msg As New MQMessage   'Define a WebSphere MQ message for our use..
Print msg.Encoding    'Currently 546 (or X'222')
                                          'Set the encoding property
                                           to 785 (or X'311')
msg.Encoding = MQENC_INTEGER_NORMAL OR MQENC_DECIMAL_NORMAL
                        OR MQENC_FLOAT_S390
Print msg.Encoding    'Print it to see the change
Dim local_num As long 'Define a long integer
local_num = 1234                          'Set it
msg.WriteLong(local_num)         'Write the number into the message
```

Conversão do conjunto de caracteres

A conversão do conjunto de caracteres será necessária ao enviar uma mensagem de um sistema para outro em que as páginas de códigos são diferentes. A conversão da página de códigos é usada por:

- Método ReadString
- Método ReadNullTerminatedString
- Método WriteString
- Método WriteNullTerminatedString
- Propriedade MessageData

Deve-se configurar a propriedade MQMessage CharacterSet para um valor de conjunto de caracteres suportados (CCSID).

WebSphere MQ Classes de Automação para ActiveX usa tabelas de conversão para executar a conversão do conjunto de caracteres.

Por exemplo, para converter sequências automaticamente para a página de códigos 437:

```
Dim msg As New MQMessage                'Define a WebSphere MQ message
msg.CharacterSet = 437                 'Set code page required
msg.WriteString "A character string"'Put character string in message
```

O método WriteString recebe os dados de sequência ("Uma sequência de caracteres" no exemplo) como uma sequência Unicode.. Ele então converte esses dados de Unicode para a página de códigos 437 usando a tabela de conversão 34B001B5.TBL.

Os caracteres na sequência Unicode que não são suportados pela página de códigos 437 serão fornecidos o caractere de substituição padrão da página de códigos 437.

Da mesma forma, quando você usa o método ReadString , a mensagem recebida possui um conjunto de caracteres estabelecido pelo valor do WebSphere MQ Message Descriptor (MQMD) e há uma conversão dessa página de códigos em Unicode antes de ser transmitida de volta para sua linguagem de script.

Encadeamentos

WebSphere MQ Classes de Automação para ActiveX implementam um modelo de encadeamento livre no qual objetos podem ser usados entre encadeamentos.

Enquanto o MQAX permite o uso de objetos MQQueue e MQQueueManager , o WebSphere MQ não permite atualmente o compartilhamento de identificações entre diferentes encadeamentos..

As tentativas de usá-las em outro encadeamento resultam em um erro e WebSphere MQ retorna um código de retorno de MQRC_HCONN_ERROR.

Nota: Há apenas um objeto MQSession por processo. O uso do MQSession CompletionCode e ReasonCode não é recomendado em ambientes multiencadeados. Os valores de erro MQSession podem ser sobrescritos por um segundo encadeamento entre um erro sendo levantado e verificado no primeiro encadeamento. Os encadeamentos são serializados para a duração de cada chamada de método ou acesso de propriedade. Portanto, emitindo um Get com a opção Espera faz com que outros encadeamentos acessem objetos MQAX para serem suspensos até que a operação seja concluída.

Manipulação de Erros

Estas informações descrevem propriedades do objeto MQAX, como funciona a manipulação de erros, regras que descrevem como exceções levantadas são manipuladas e obtenção de uma propriedade.

Cada objeto MQAX inclui propriedades para manter informações de erro e um método para reconfigurar ou limpar as mesmas. As propriedades são:

- CompletionCode
- ReasonCode
- ReasonName

O método é:

- ClearErrorCodes

Como a manipulação de erros funciona

O script ou aplicativo MQAX chama o método de um objeto MQAX ou acessa ou atualiza uma propriedade do objeto MQAX:

1. O ReasonCode e o CompletionCode no objeto em questão são atualizados.
2. O ReasonCode e CompletionCode no objeto MQSession também são atualizados com as mesmas informações.

   Nota: Consulte “Encadeamentos” na página 1050 para obter as restrições sobre o uso de códigos de erro MQSession em aplicativos encadeados.

Se o CompletionCode for maior ou igual à propriedade ExceptionThreshold de MQSession, MQAX lançará uma exceção (número 32000). Use isso dentro de seu script usando a instrução On Error (ou equivalente) para processá-la.

3. Use a função Error para recuperar a sequência de erros associada, que tem o formato:

```
MQAX: CompletionCode=xxx, ReasonCode=xxx, ReasonName=xxx
```

Para obter mais informações sobre como usar as instruções On Error, consulte a documentação para a linguagem de script ActiveX.

Usar CompletionCode e ReasonCode no objeto MQSession é conveniente para manipuladores de erros simples.

A propriedade ReasonName retorna o nome simbólico do WebSphere MQ para o valor atual do ReasonCode.

Levantando exceções

As regras a seguir descrevem como exceções levantadas são manipuladas:

- Sempre que uma propriedade ou um método configurar o código de conclusão para um valor maior ou igual ao limite de exceção (geralmente configurado como 2), uma exceção será levantada.
- Todas as chamadas de método e conjuntos de propriedades configuram o código de conclusão.

Obtendo uma propriedade

Este é um caso especial, pois CompletionCode e ReasonCode não são sempre atualizados:

- Se uma propriedade get for bem-sucedida, o objeto e o ReasonCode e o CompletionCode do objeto MQSession permanecem inalterados.
- Se uma propriedade get falhar com um CompletionCode de aviso, o ReasonCode e o CompletionCode permanecerão inalterados.
- Se uma propriedade get falhar com um CompletionCode de erro, o ReasonCode e o CompletionCode serão atualizados para refletirem os valores verdadeiros e o processamento de erro continua conforme descrito.

A classe MQSession possui um método *ReasonCodeName* que pode ser usado para substituir um código de razão WebSphere MQ por um nome simbólico. Isso é especialmente útil durante o desenvolvimento de programas em que podem ocorrer erros inesperados. No entanto, o nome não é ideal para apresentação aos usuários.

Cada classe também tem uma propriedade *ReasonName*, que retorna o nome simbólico do código de razão atual para essa classe.

WebSphere MQ Classes de Automação para referência ActiveX

Esta seção descreve as classes do WebSphere MQ Automation Classes para ActiveX (MQAX), desenvolvidas para ActiveX. As classes permitem que você grave aplicativos ActiveX que podem acessar outros aplicativos em execução em ambientes nãoActiveX , usando WebSphere MQ.

WebSphere MQ Classes de Automação para a interface ActiveX

WebSphere MQ Automation Classes para ActiveX fornece constantes ActiveX numéricas predefinidas (como MQMT_REQUEST) necessárias para usar as classes.

As classes de automação ActiveX consistem no seguinte:

- "Classe MQSession" na página 1053
- "Classe MQQueueManager" na página 1056

- “Classe MQQueue” na página 1067
- “Classe MQMessage” na página 1082
- “Classe MQPutMessageOptions” na página 1103
- “Classe MQGetMessageOptions” na página 1106
- “Classe MQDistributionList” na página 1108
- “Classe MQDistributionListItem” na página 1112

Além disso, o WebSphere MQ Automation Classes para ActiveX fornece constantes ActiveX numéricas predefinidas (como MQMT_REQUEST) necessárias para usar as classes. Essas são fornecidas na enumeração MQ na biblioteca MQAX200. As constantes são um subconjunto daqueles definidos nos arquivos de cabeçalho C do WebSphere MQ (cmqc * .h) com alguns códigos de Razão adicionais do WebSphere MQ Automation Classes para ActiveX .

Sobre as Classes de Automação do WebSphere MQ para classes ActiveX

Leia estas informações juntamente com os tópicos de referência em Desenvolvendo referência de aplicativos.

Consulte Recursos que podem ser usados somente com a instalação primária no Windows para obter informações importantes

A classe MQSession fornece um objeto raiz que contém o status da última ação executada em qualquer um dos objetos MQAX. Consulte o “Manipulação de Erros” na página 1050 para obter informações adicionais.

As classes MQQueueManager e MQQueue fornecem acesso aos objetos subjacentes do WebSphere MQ . Os métodos ou acessos de propriedade para essas classes em geral resultam em chamadas feitas no MQI do WebSphere MQ .

As classes MQMessage, MQPutMessageOptions e MQGetMessageOptions contêm as estruturas de dados do MQMD, MQPMO e MQGMO e são usadas para ajudá-lo a enviar mensagens para filas e recuperar mensagens delas.

A classe MQDistributionList contém uma coleta de filas - local, remoto ou alias para saída. A classe MQDistributionListItem contém as estruturas MQOR, MQRR e MQPMR e as associa a uma lista de distribuição de propriedade.

Transmissão de parâmetros

Os parâmetros em chamadas de métodos são todos transmitidos por valor, exceto se esse parâmetro for um objeto, em cujo caso é uma referência transmitida.

As definições de classe fornecidas listam o Tipo de dados para cada parâmetro ou propriedade. Para muitos clientes ActiveX, como Visual Basic, se a variável usada não for do tipo requerido, o valor será automaticamente convertido para ou a partir do tipo requerido - o fornecimento de tal conversão é possível. Isso segue regras padrão do cliente; MQAX não fornece essa conversão.

Muitos dos métodos têm parâmetros de sequência de comprimento fixo ou retorna uma sequência de caracteres de comprimento fixo. As regras de conversão são as seguintes:

- Se o usuário fornecer uma sequência de comprimento fixo de comprimento incorreto, como um parâmetro de entrada ou como um valor de retorno, o valor será truncado ou preenchido com espaços à direita, conforme necessário.
- Se o usuário fornecer uma sequência de comprimento variável de comprimento incorreto como um parâmetro de entrada, o valor será truncado ou preenchido com espaços à direita.
- Se o usuário fornecer uma sequência de comprimento variável do comprimento incorreto como um valor de retorno, a sequência será ajustada para o comprimento necessário (porque retornar um valor inutilizará o valor anterior na sequência de qualquer forma).
- As sequências fornecidas como parâmetros de entrada podem conter Nulos integrados.

Estas classes podem ser localizadas na biblioteca MQAX200.

Métodos de acesso ao objeto

Esses métodos não se relacionam diretamente a nenhuma única chamada do WebSphere MQ Cada um desses métodos cria um objeto no qual as informações de referência são, então, mantidas, seguidas pela conexão ou abertura de um objeto WebSphere MQ :

Quando uma conexão é feita com um gerenciador de filas, ele contém o atributo 'connection handle' gerado pelo WebSphere MQ..

Quando uma fila é aberta, ela mantém o atributo 'object handle' gerado pelo WebSphere MQ.

Esses atributos do WebSphere MQ não estão diretamente disponíveis para o programa MQAX.

Erros

Os erros sintáticos na transmissão de parâmetro podem ser detectados no tempo de compilação e no tempo de execução pelo cliente ActiveX. Os erros podem ser capturados usando On Error no Visual Basic.

Todas as classes WebSphere MQ ActiveX contêm duas propriedades especiais somente leitura- ReasonCode e CompletionCode. Essas propriedades podem ser lidas a qualquer momento.

Uma tentativa de acessar qualquer outra propriedade ou emitir qualquer chamada de método pode gerar um erro a partir do WebSphere MQ.

Se um conjunto de propriedades ou chamada de método for bem-sucedido, o ReasonCode do objeto de propriedade será configurado para MQRC_NONE e CompletionCode para MQCC_OK.

Se o acesso de propriedade ou chamada de método não for bem-sucedido, os códigos de razão e de conclusão serão configurados nesses campos.

Classe MQSession

Esta é a classe raiz para WebSphere MQ Classes de Automação para ActiveX.

Há sempre apenas um objeto MQSession por processo do cliente ActiveX. Uma tentativa de criar um segundo objeto cria uma segunda referência ao objeto original.

Criação

Novo cria um novo objeto MQSession.

Sintaxe

Dim *mqsess* **As New MQSession Set** *mqsess* = **New MQSession**

Propriedades

- “propriedade CompletionCode” na página 1054.
- “Propriedade ExceptionThreshold” na página 1054.
- “propriedade ReasonCode” na página 1054.
- “Propriedade ReasonName” na página 1055.

Método

- “Método AccessGetMessageOptions” na página 1055.
- “Método AccessMessage” na página 1055.
- “Método AccessPutMessageOptions” na página 1055.
- “Método AccessQueueManager” na página 1055.

- “método ClearErrorCodes” na página 1056.
- “Método ReasonCodeName” na página 1056.

propriedade CompletionCode

Somente leitura. Retorna o código de conclusão do WebSphere MQ configurado pelo método ou conjunto de propriedades mais recente emitido em qualquer objeto do WebSphere MQ .

Ele é reconfigurado para MQCC_OK quando um método ou um conjunto de propriedades é chamado com êxito em relação a qualquer objeto MQAX.

Um manipulador de eventos de erro pode inspecionar essa propriedade para diagnosticar o erro, sem ter que saber qual objeto foi envolvido.

Usando o CompletionCode e ReasonCode no objeto MQSession é muito conveniente para manipuladores de erro simples.

Nota: Consulte “Encadeamentos” na página 1050 para obter as restrições sobre o uso de códigos de erro MQSession em aplicativos encadeados.

Definido em: classe MQSession

Tipo de dados: longo

Valores:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

Sintaxe:

Para obter: *completioncode & = MQSession*.CompletionCode

Propriedade ExceptionThreshold

Leitura/gravação. Define o nível de erro do WebSphere MQ para o qual MQAX emitirá uma exceção. Usa como padrão MQCC_FAILED. Um valor maior do que MQCC_FAILED efetivamente impede o processamento de exceções, deixando o programador executar verificações no CompletionCode e ReasonCode.

Definido em: classe MQSession

Tipo de dados: longo

Valores:

- Qualquer, mas considere MQCC_WARNING, MQCC_FAILED ou superior.

Sintaxe:

Para obter: *ExceptionThreshold& = MQSession*.**ExceptionThreshold**

Para configurar: *MQSession*.**ExceptionThreshold** = *ExceptionThreshold$*

propriedade ReasonCode

Somente leitura. Retorna o código de razão configurado pelo método ou conjunto de propriedades mais recente emitido em qualquer objeto WebSphere MQ .

Um manipulador de eventos de erro pode inspecionar essa propriedade para diagnosticar o erro, sem ter que saber qual objeto foi envolvido.

Usando o CompletionCode e ReasonCode no objeto MQSession é muito conveniente para manipuladores de erro simples.

Nota: Consulte “Encadeamentos” na página 1050 para obter as restrições sobre o uso de códigos de erro MQSession em aplicativos encadeados.

Definido em: classe MQSession

Tipo de dados: longo

Valores:

- Consulte Reason (MQLONG) e os valores MQAX adicionais listados em “Códigos de Razão” na página 1119.

Sintaxe: Obter: *reasoncode & = MQSession*.**ReasonCode**

Propriedade ReasonName

Somente leitura. Retorna o nome simbólico do código de razão mais recente. Por exemplo, "MQRC_QMGR_NOT_AVAILABLE".

Nota: Consulte “Encadeamentos” na página 1050 para obter as restrições sobre o uso de códigos de erro MQSession em aplicativos encadeados.

Definido em: classe MQSession

Tipo de dados: sequência

Valores:

- Consulte Códigos de razão de API.

Sintaxe: Obter: *reasonname $= MQSession*.**ReasonName**

Método AccessGetMessageOptions

Cria um novo objeto MQGetMessageOptions.

Definido em: classe MQSession

Sintaxe: *gmo = MQSession*.**AccessGetMessageOptions()**

Método AccessMessage

Cria um novo objeto MQMessage.

Definido em: classe MQSession

Sintaxe: *msg = MQSession*.**AccessMessage()**

Método AccessPutMessageOptions

Cria um novo objeto MQPutMessageOptions.

Definido em: classe MQSession

Sintaxe: *pmo = MQSession*.**AccessPutMessageOptions()**

Método AccessQueueManager

Cria um novo objeto MQQueueManager e o conecta a um gerenciador de filas real por meio do cliente MQI do WebSphere MQ ou servidor WebSphere MQ . Assim como executar uma conexão, esse método também executa uma abertura para o objeto do gerenciador de filas.

Quando o cliente MQI do WebSphere MQ e o servidor WebSphere MQ estiverem instalados em seu sistema, os aplicativos MQAX serão executados no servidor por padrão. Para executar MQAX com relação ao cliente, a biblioteca de ligações do cliente deve ser especificada na variável de ambiente `GMQ_MQ_LIB`, por exemplo, configure `GMQ_MQ_LIB=mqic.dll`.

Para uma instalação somente do cliente, não é necessário configurar a variável de ambiente `GMQ_MQ_LIB`. Quando essa variável não for configurada o WebSphere MQ tenta carregar amqzst.dll. Se essa DLL não estiver presente (como é o caso em uma instalação somente do cliente), o WebSphere MQ tentará carregar mqic.dll.

Se bem-sucedido, configura o ConnectionStatus do MQQueueManager para TRUE.

Um gerenciador de filas pode ser conectado a no máximo um objeto MQQueueManager por instância ActiveX.

Se a conexão com o gerenciador de filas falhar, um evento de erro será levantado e o ReasonCode e o CompletionCode do objeto MQSession serão configurados.

Definido em: classe MQSession

Sintaxe: *configurado qm = MQSession*.**AccessQueueManager** (*Name$*)

Parâmetro:*Name$* Sequência. Nome do Gerenciador de filas ao qual ser conectado.

método ClearErrorCodes

Reconfigura o CompletionCode para MQCC_OK e MQRC_NONE para ReasonCode.

Definido em: classe MQSession

Sintaxe: Chamar *MQSession*.**ClearErrorCodes ()**

Método ReasonCodeName

Retorna o nome do código de razão com o valor numérico determinado. É útil fornecer indicações mais claras de condições de erro aos usuários. O nome ainda é um pouco críptico (por exemplo, ReasonCodeName (2059) é **MQRC_Q_MGR_NOT_AVAILABLE**), portanto, quando possíveis erros devem ser capturados e substituídos por texto descritivo apropriado para o aplicativo.

Definido em: classe MQSession

Sintaxe: *errname $= MQSession*.**ReasonCodeNome**(*reasonCode&*)

Parâmetro:*reasoncode &* longo. O código de razão para o qual o nome simbólico é necessário.

Classe MQQueueManager

Esta classe representa uma conexão com um gerenciador de filas. O gerenciador de filas pode estar em execução localmente (um servidor WebSphere MQ) ou remotamente com acesso fornecido pelo cliente WebSphere MQ . Um aplicativo deve criar um objeto desta classe e conectá-lo a um gerenciador de filas. Quando um objeto dessa classe é destruído, ele é automaticamente desconectado de seu gerenciador de filas.

Restrição

Os objetos da classe MQQueue são associadas a essa classe.

Novo cria um novo objeto MQQueueManager e configura todas as propriedades como os valores iniciais. Como alternativa, use o método AccessQueueManager da classe MQSession.

Criação

Novo cria um **novo** objeto MQQueueManager e configura todas as propriedades como os valores iniciais. Como alternativa, use o método AccessQueueManager da classe MQSession.

Sintaxe

Dim *mgr* **As New MQQueueManager set** *mgr* = **New MQQueueManager**

Propriedades

- "Propriedade AlternateUserId" na página 1058.
- "Propriedade AuthorityEvent" na página 1058.

- “Propriedade BeginOptions” na página 1058.
- “Propriedade ChannelAutoDefinition” na página 1059.
- “Propriedade ChannelAutoDefinitionEvent” na página 1059.
- “Propriedade ChannelAutoDefinitionExit” na página 1059.
- “Propriedade CharacterSet” na página 1059.
- “Propriedade CloseOptions” na página 1060.
- “Propriedade CommandInputQueueName” na página 1060.
- “Propriedade CommandLevel” na página 1060.
- “propriedade CompletionCode” na página 1060.
- “Propriedade ConnectionHandle” na página 1060.
- “Propriedade ConnectionStatus” na página 1060.
- “Propriedade ConnectOptions” na página 1061.
- “Propriedade DeadLetterQueueName” na página 1061.
- “Propriedade DefaultTransmissionQueueName” na página 1061.
- “Propriedade Description” na página 1061.
- “Propriedade DistributionLists” na página 1061.
- “Propriedade InhibitEvent” na página 1062.
- “Propriedade IsConnected” na página 1062.
- “Propriedade IsOpen” na página 1062.
- “Propriedade LocalEvent” na página 1062.
- “Propriedade MaximumHandles” na página 1063.
- “Propriedade MaximumMessageLength” na página 1063.
- “Propriedade MaximumPriority” na página 1063.
- “Propriedade MaximumUncommittedMessages” na página 1063.
- “Propriedade Name” na página 1063.
- “Propriedade ObjectHandle” na página 1063.
- “Propriedade PerformanceEvent” na página 1064.
- “Propriedade Platform” na página 1064.
- “propriedade ReasonCode” na página 1064.
- “Propriedade ReasonName” na página 1064.
- “Propriedade RemoteEvent” na página 1064.
- “Propriedade StartStopEvent” na página 1065.
- “Propriedade SyncPointAvailability” na página 1065.
- “Propriedade TriggerInterval” na página 1065.

Methods

- “Método AccessQueue” na página 1065.
- “Método AddDistributionList” na página 1066.
- “Método Backout” na página 1066.
- “Método Begin” na página 1066.
- “método ClearErrorCodes” na página 1066.
- “Método Commit” na página 1067.
- “Método Connect” na página 1067.

- “Método Disconnect” na página 1067.

Acesso à propriedade

As propriedades a seguir podem ser acessadas a qualquer momento.

- “Propriedade AlternateUserId” na página 1058.
- “propriedade CompletionCode” na página 1060.
- “Propriedade ConnectionStatus” na página 1060.
- “propriedade ReasonCode” na página 1064.

As propriedades restantes poderão ser acessadas apenas se o objeto estiver conectado a um gerenciador de filas e o ID do usuário estiver autorizado a consultar em relação a esse gerenciador de filas. Se um ID de usuário alternativo for configurado e o ID do usuário atual estiver autorizado a usá-lo, o ID de usuário alternativo será marcado para autorização para consulta no lugar

Se essas condições não se aplicarem, o WebSphere MQ Automation Classes para ActiveX tentará se conectar ao gerenciador de filas e abri-lo para consulta automaticamente. Se isto for malsucedido, a chamada configurará um CompletionCode de MQCC_FAILED e um dos seguintes ReasonCodes:

- MQRC_CONNECTION_BROKEN
- MQRC_NOT_AUTHORIZED
- MQRC_Q_MGR_NAME_ERROR
- MQRC_Q_MGR_NOT_AVAILABLE

Propriedade AlternateUserId

Leitura/gravação. O ID do usuário alternativo a ser utilizado para validar o acesso aos atributos do gerenciador de filas.

Esta propriedade não deve ser configurada se IsConnected for TRUE.

Esta propriedade não pode ser configurada enquanto o objeto está aberto.

`Defined in:` Classe MQQueueManager

`Data Type:` Sequência de 12 caracteres

`Syntax:` Para obter: *altuser $= MQQueueManager*.**AlternateUserId** Para configurar: *MQQueueManager*.**AlternateUserId** = *altuser $*

Propriedade AuthorityEvent

Somente leitura. O atributo de MQI AuthorityEvent.

Definido em:
Classe MQQueueManager

Tipo de dados:
Long

Valores:

- MQEVR_DISABLED
- MQEVR_ENABLED

Sintaxe: Para obter: *authevent = MQQueueManager*.**AuthorityEvent**

Propriedade BeginOptions

Leitura/gravação. Estas são as opções que se aplicam ao método Begin. Inicialmente MQBO_NONE.

Definido em:
Classe MQQueueManager

Tipo de dados:
Long

Valores:

- MQBO_NONE

Sintaxe: Para obter: *beginoptions & =MQQueueManager*.**BeginOptions**

Para configurar: *MQQueueManager*.**BeginOptions**=*beginoptions &*

Propriedade ChannelAutoDefinition

Somente leitura. Isso controla se a definição de canal automática é permitida.

Definido em:
Classe MQQueueManager

Tipo de dados:
Long

Valores:

- MQCHAD_DISABLED
- MQCHAD_ENABLED

Sintaxe: Para obter: *channelautodef & = MQQueueManager*.**ChannelAutoDefinição**

Propriedade ChannelAutoDefinitionEvent

Somente leitura. Isso controla se os eventos de definição de canal automáticos são gerados.

Definido em:
Classe MQQueueManager

Tipo de dados:
Long

Valores:

- MQEVR_DISABLED
- MQEVR_ENABLED

Sintaxe: Para obter: *channelautoderrovent & =MQQueueManager*.**ChannelAutoDefinitionEvent**

Propriedade ChannelAutoDefinitionExit

Somente leitura. O nome do usuário utilizado para saída de definição automática do canal.

Definido em:
Classe MQQueueManager

Tipo de dados:
Sequência

Sintaxe: Obter: *channelautodefexit$= MQQueueManager*.**ChannelAutoDefinitionExit**

Propriedade CharacterSet

Somente leitura. O atributo CodedCharSetId de MQI.

Definido em: classe MQQueueManager

Tipo de dados: longo

Sintaxe: Para obter: *characterset & = MQQueueManager*.**CharacterSet**

Propriedade CloseOptions

Leitura/gravação. Opções usadas para controlar o que acontece quando o gerenciador de filas é fechada. O valor inicial é MQCO_NONE.

Definido em:
Classe MQQueueManager

Tipo de dados:
Long

Valores:

- MQCO_NONE

Sintaxe: Para obter: *closeopt & = MQQueueManager*.**CloseOptions**

Para configurar: *MQQueueManager*.**CloseOptions** =*closeopt &*

Propriedade CommandInputQueueName

Somente leitura. O atributo de MQI CommandInputQName.

Definido em: classe MQQueueManager

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *comanddinputqname $= MQQueueManager*.**CommandInputQueueName**

Propriedade CommandLevel

Somente leitura. Retorna a versão e o nível da implementação do gerenciador de filas do WebSphere MQ (atributo MQI CommandLevel)

Definido em: classe MQQueueManager

Tipo de dados: longo

Sintaxe: Para obter: *level & = MQQueueManager*.**CommandLevel**

propriedade CompletionCode

Somente leitura. Retorna o código de conclusão configurados pelo último método ou acesso de propriedade emitidos com relação ao objeto.

Definido em: classe MQQueueManager

Tipo de dados: longo

Valores:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

Sintaxe: Para obter: *completioncode & = MQQueueManager*.**CompletionCode**

Propriedade ConnectionHandle

Somente leitura. A manipulação de conexões para o objeto do gerenciador de fila do WebSphere MQ

Definido em:
Classe MQQueueManager

Tipo de dados:
Long

Sintaxe: Para obter: *hconn & = MQQueueManager.***ConnectionHandle** .

Propriedade ConnectionStatus

Somente leitura. Indica se o objeto está conectado a seu gerenciador de filas ou não.

Definido em: classe MQQueueManager

Tipo de dado: booleano

Valores:

- TRUE (-1)
- FALSE (0)

Sintaxe: Para obter: *status = MQQueueManager***.ConnectionStatus**

Propriedade ConnectOptions

Leitura/Gravação. Essas opções se aplicam ao método Connect. Inicialmente MQCNO_NONE.

Definido em:
Classe MQQueueManager

Tipo de dados:
Long

Valores:

- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING
- MQCNO_NONE

Sintaxe: Para obter: *connectoptions & =MQQueueManager*.**ConnectOptions**

Para configurar: *MQQueueManager***.ConnectOptions**=*connectoptions &*

Propriedade DeadLetterQueueName

Somente leitura. O atributo de MQI DeadLetterQName.

Definido em: classe MQQueueManager

Tipo de dados: sequência de 48 caracteres

Sintaxe: para obter: *dlqname $= MQQueueManager***.DeadLetterQueueName** .

Propriedade DefaultTransmissionQueueName

Somente leitura. O atributo de MQI DefXmitQName.

Definido em: classe MQQueueManager

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *defxmitqname $= MQQueueManager***.DefaultTransmissionQueueName**

Propriedade Description

Somente leitura. O atributo QMgrDesc do MQI.

Definido em: classe MQQueueManager

Tipo de dado: Sequência de 64 caracteres

Sintaxe: Para obter: *descrição $= MQQueueManager***..Descrição**

Propriedade DistributionLists

Somente leitura. Essa é a capacidade do gerenciador de filas para suportar listas de distribuição.

Definido em:
Classe MQQueueManager

Tipo de dados:
Booleana

Valores:

- TRUE (-1)
- FALSE (0)

Sintaxe: Para obter: *distributionlists= MQQueueManager*.**DistributionLists**

Propriedade InhibitEvent

Somente leitura. O atributo de MQI InhibitEvent.

Definido em: classe MQQueueManager

Tipo de dados: longo

Valores:

- MQEVR_DISABLED
- MQEVR_ENABLED

Sintaxe: Para obter: *inibir & = MQQueueManager*.**InhibitEvent** .

Propriedade IsConnected

Um valor que indica se o gerenciador de filas está atualmente conectado.

Somente leitura.

Definido em: classe MQQueueManager

Tipo de dado: booleano

Valores:

- TRUE (-1)
- FALSE (0)

Sintaxe: Para obter: *isconnected = MQQueueManager*.**IsConnected**

Propriedade IsOpen

Um valor que indica se o gerenciador de filas está atualmente aberto para consulta.

Somente leitura.

Definido em:
Classe MQQueueManager

Tipo de dados:
Booleana

Valores:

- TRUE (-1)
- FALSE (0)

Sintaxe: Para obter: *IsOpen = MQQueueManager*.**IsOpen**

Propriedade LocalEvent

Somente leitura. O atributo de MQI LocalEvent.

Definido em: classe MQQueueManager

Tipo de dados: longo

Valores:

- MQEVR_DISABLED
- MQEVR_ENABLED

Sintaxe: Para obter: *localevent & = MQQueueManager*.**LocalEvent**

Propriedade MaximumHandles

Somente leitura. O atributo MaxHandles MQI.

Definido em: classe MQQueueManager

Tipo de dados: longo

Sintaxe: Para obter: *maxidentificadores & = MQQueueManager*.**MaximumHandles**

Propriedade MaximumMessageLength

Somente leitura. O atributo Gerenciador de Filas MaxMsgLength MQI.

Definido em: classe MQQueueManager

Tipo de dados: longo

Sintaxe: Para obter: *maxmessagelength & = MQQueueManager*.**MaximumMessageLength**

Propriedade MaximumPriority

Somente leitura. O atributo MaxPriority do MQI.

Definido em: classe MQQueueManager

Tipo de dados: longo

Sintaxe: Para obter: *maxpriority & = MQQueueManager*.**MaximumPriority**

Propriedade MaximumUncommittedMessages

Somente leitura. O atributo MQI MaxUncommittedMsgs.

Definido em: classe MQQueueManager

Tipo de dados: longo

Sintaxe: Obter: *maxuncommitted & = MQQueueManager*.**MaximumUncommittedMensagens**

Propriedade Name

Leitura/gravação. O atributo QMgrName do MQI. Esta propriedade não pode ser gravada quando o MQQueueManager está conectado.

Definido em: classe MQQueueManager

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *name $= MQQueueManager*.**name**

Para configurar: *MQQueueManager*.**name** = *name $*

Nota: O Visual Basic reserva a propriedade "Name" para uso na interface visual. Portanto, ao usar no Visual Basic, utilize letras minúsculas, ou seja, "name".

Propriedade ObjectHandle

Somente leitura. A manipulação de objetos para o objeto do gerenciador de filas do WebSphere MQ

Definido em:
Classe MQQueueManager

Tipo de Dados
Long

Sintaxe: Para obter: *hobj & = MQQueueManager.***ObjectHandle**

Propriedade PerformanceEvent

Somente leitura. O atributo de MQI PerformanceEvent.

Definido em: classe MQQueueManager

Tipo de dados: longo

Valores:

- MQEVR_DISABLED
- MQEVR_ENABLED

Sintaxe: Para obter: *perfevent & = MQQueueManager*.PerformanceEvent

Propriedade Platform

Somente leitura. O atributo de Plataforma MQI.

Definido em: classe MQQueueManager

Tipo de dados: longo

Valores:

- MQPL_WINDOWS_NT
- MQPL_WINDOWS

Sintaxe: Para obter: *platform & = MQQueueManager***Plataforma**

propriedade ReasonCode

Somente leitura. Retorna o código de razão definido pelo último método ou acesso de propriedade emitidos com relação ao objeto.

Definido em: classe MQQueueManager

Tipo de dados: longo

Valores:

- Consulte Códigos de razão de API.

Sintaxe: Para obter: *reasoncode & = MQQueueManager***.ReasonCode**

Propriedade ReasonName

Somente leitura. Retorna o nome simbólico do código de razão mais recente. Por exemplo, "MQRC_QMGR_NOT_AVAILABLE".

Definido em: classe MQQueueManager

Tipo de dados: sequência

Valores:

- Consulte Códigos de razão de API.

Sintaxe: Para obter: *reasonName $= MQQueueManager***.ReasonName**

Propriedade RemoteEvent

Somente leitura. O atributo de MQI RemoteEvent.

Definido em: classe MQQueueManager

Tipo de dados: longo

Valores:

- MQEVR_DISABLED
- MQEVR_ENABLED

Sintaxe: Para obter: *remoteevent & = MQQueueManager*.**RemoteEvent**

Propriedade StartStopEvent

Somente leitura. O atributo de MQI StartStopEvent.

Definido em: classe MQQueueManager

Tipo de dados: longo

Valores:

- MQEVR_DISABLED
- MQEVR_ENABLED

Sintaxe: Para obter: *strstpevent & = MQQueueManager*.**StartStopEvento**

Propriedade SyncPointAvailability

Somente leitura. O atributo SyncPoint MQI.

Definido em: classe MQQueueManager

Tipo de dados: longo

Valores:

- MQSP_AVAILABLE
- MQSP_NOT_AVAILABLE

Sintaxe: Para obter: *syncpointavailability & = MQQueueManager*.**SyncPointDisponibilidade**

Propriedade TriggerInterval

Somente leitura. O atributo de MQI TriggerInterval.

Definido em: classe MQQueueManager

Tipo de dados: longo

Sintaxe: Para obter: *trigint & = MQQueueManager*.**TriggerInterval**

Método AccessQueue

Cria um objeto MQQueue e associa-o a este objeto MQQueueManager configurando a propriedade de referência de conexão da fila. Ele configura as propriedades Nome, OpenOptions, DynamicQueueName e AlternateUserId do objeto MQQueue para os valores fornecidos e tentativas para abri-lo.

Se a abertura for mal sucedida, a chamada falhará. Um evento de erro é emitido em relação ao objeto. O ReasonCode e CompletionCode e o MQSession ReasonCode e CompletionCode do objeto são configurados.

Os parâmetros DynamicQueueName, QueueManagerName e AlternateUserId são opcionais e padronizados para "".

O OpenOption MQOO_INQUIRE deve ser especificado além das outras opções, se as propriedades da fila deverem ser lidas.

Não configure o QueueManagerName ou configure-o para "" se a fila a ser aberta for local. Caso contrário, configure-o para o nome do gerenciador de filas remotas que possui a fila e uma tentativa é feita para abrir uma definição local da fila remota. Para obter mais informações sobre resolução do nome da fila remota e alias do gerenciador de filas, consulte O que são aliases? .

Se a propriedade Nome estiver configurada para um nome de fila modelo, especifique o nome da fila dinâmica a ser criada no parâmetro DynamicQueueName$. Se o valor fornecido no parâmetro DynamicQueueName$ for "", o valor configurado no objeto de fila e usado na chamada aberta será "AMQ.* ". Consulte “Criando filas dinâmicas” na página 226 para obter mais informações sobre a nomenclatura de filas dinâmicas.

Definição

Definido em: classe MQQueueManager.

Sintaxe

Sintaxe: configurar fila = MQQueueManager.**AccessQueue**(Name$, OpenOptions&, QueueManagerName$, DynamicQueueName$, AlternateUserId$)

Parâmetros

Name$ String. Nome da fila do WebSphere MQ .

OpenOptions: Long. Opções a serem usadas quando a fila for aberta. Consulte OpenOptions (MQLONG).

QueueManagerName$ String. Nome do gerenciador de filas que possui a fila a ser aberta. Um valor "" indica que o gerenciador de filas é local.

*DynamicQueueName$* String. O nome designado à fila dinâmica no momento em que a fila é aberta quando o parâmetro Name$ especifica uma fila modelo.

AlternateUserId$ String. O ID do usuário alternativo usado para validar o acesso ao abrir a fila.

Método AddDistributionList

Cria um novo objeto MQDistributionList e configura sua referência de conexão para o gerenciador de filas proprietário.

Definido em:
Classe MQQueueManager

Sintaxe: *set distributionlist* = **MQQueueManager.***AddDistributionLista*

Método Backout

Faz saída quaisquer entradas de mensagens não consolidadas e obtenções que ocorreram como parte de uma unidade de trabalho desde a última sincronização.

Definido em: classe MQQueueManager

Sintaxe: Call *MQQueueManager.***Backout ()**

Método Begin

Inicia uma unidade de trabalho que é coordenada pelo gerenciador de filas. O iniciar opções afetam o comportamento deste método.

Definido em:
Classe MQQueueManager

Sintaxe: Chamada *MQQueueManager*.**Begin ()**

método ClearErrorCodes

Reconfigura o CompletionCode para MQCC_OK e MQRC_NONE para ReasonCode para a classe MQQueueManager e a classe MQSession.

Definido em: classe MQQueueManager

Sintaxe: Chamar *MQQueueManager*.**ClearErrorCódigos ()**

Método Commit

Confirma quaisquer inserções de mensagem e obtenções que ocorreram como parte de uma unidade de trabalho desde o último ponto de sincronização.

Definido em: classe MQQueueManager

Sintaxe: Call *MQQueueManager*.**Commit ()**

Método Connect

Conecta o objeto MQQueueManager a um gerenciador de fila real por meio do cliente ou servidor MQI WebSphere MQ . Além de estabelecer a conexão, este método também abre o objeto de gerenciador de filas para que ele possa ser consultado.

Configura IsConnected para TRUE.

Um máximo de um objeto MQQueueManager por instância do ActiveX é permitido para se conectar a um gerenciador de filas.

Definido em: classe MQQueueManager

Sintaxe: Call *MQQueueManager*.**Conecte ()**

Método Disconnect

Desconecta o objeto MQQueueManager do gerenciador de filas.

Configura IsConnected para FALSE.

Todos os objetos de Filas associados ao objeto MQQueueManager tornam-se inutilizáveis e não podem ser reabertos.

Qualquer mudança não confirmada (envios e recebimentos de mensagens) é confirmada.

Definido em: classe MQQueueManager

Sintaxe: Call *MQQueueManager*.**Desconectar ()**

Classe MQQueue

Essa classe representa o acesso a uma fila do WebSphere MQ . Essa conexão é fornecida por um objeto MQQueueManager associado. Quando um objeto desta classe é destruído, é fechado automaticamente.

Restrição

A classe MQQueue é contida pela classe MQQueueManager.

Criação

New cria um novo objeto MQQueue e define todas as propriedades para os valores iniciais. Como alternativa, use o método AccessQueue da classe MQQueueManager.

Sintaxe

```
Dim que As New MQQueue Set que  = New MQQueue
```

Propriedades

- “Propriedade AlternateUserId” na página 1070.
- “Propriedade BackoutRequeueName” na página 1070.

- “Propriedade BackoutThreshold” na página 1070.
- “Propriedade BaseQueueName” na página 1070.
- “Propriedade CloseOptions” na página 1071.
- “propriedade CompletionCode” na página 1071.
- “Propriedade ConnectionReference” na página 1071.
- “Propriedade CreationDateTime” na página 1071.
- “Propriedade CurrentDepth” na página 1071.
- “Propriedade DefaultInputOpenOption” na página 1072.
- “Propriedade DefaultPersistence” na página 1072.
- “Propriedade DefaultPriority” na página 1072.
- “Propriedade DefinitionType” na página 1072.
- “Propriedade DepthHighEvent” na página 1072.
- “Propriedade DepthHighLimit” na página 1073.
- “Propriedade DepthLowEvent” na página 1073.
- “Propriedade DepthLowLimit” na página 1073.
- “Propriedade DepthMaximumEvent” na página 1073.
- “Propriedade DepthHighEvent” na página 1072.
- “Propriedade DepthHighLimit” na página 1073.
- “Propriedade DepthLowEvent” na página 1073.
- “Propriedade DepthLowLimit” na página 1073.
- “Propriedade DepthMaximumEvent” na página 1073.
- “Propriedade Description” na página 1073.
- “Propriedade DynamicQueueName” na página 1073.
- “Propriedade HardenGetBackout” na página 1074.
- “Propriedade InhibitGet” na página 1074.
- “Propriedade InhibitPut” na página 1074.
- “Propriedade InitiationQueueName” na página 1074.
- “Propriedade IsOpen” na página 1075.
- “Propriedade MaximumDepth” na página 1075.
- “Propriedade MaximumMessageLength” na página 1075.
- “Propriedade MessageDeliverySequence” na página 1075.
- “Propriedade ObjectHandle” na página 1076.
- “Propriedade OpenInputCount” na página 1076.
- “Propriedade OpenOptions” na página 1076.
- “Propriedade OpenOutputCount” na página 1076.
- “Propriedade OpenStatus” na página 1076.
- “Propriedade ProcessName” na página 1076.
- “Propriedade QueueManagerName” na página 1077.
- “Propriedade QueueType” na página 1077.
- “propriedade ReasonCode” na página 1077.
- “Propriedade ReasonName” na página 1077.
- “Propriedade RemoteQueueManagerName” na página 1077.
- “Propriedade RemoteQueueName” na página 1078.

- “Propriedade ResolvedQueueManagerName” na página 1078.
- “Propriedade ResolvedQueueName” na página 1078.
- “Propriedade RetentionInterval” na página 1078.
- “Propriedade Scope” na página 1078.
- “Propriedade ServiceInterval” na página 1078.
- “Propriedade ServiceIntervalEvent” na página 1078.
- “Propriedade Shareability” na página 1079.
- “Propriedade TransmissionQueueName” na página 1079.
- “Propriedade TriggerControl” na página 1079.
- “Propriedade TriggerData” na página 1079.
- “Propriedade TriggerDepth” na página 1079.
- “Propriedade TriggerMessagePriority” na página 1080.
- “Propriedade TriggerType” na página 1080.
- “Propriedade Usage” na página 1080.

Methods

- “método ClearErrorCodes” na página 1080
- “Método Close” na página 1080
- “Método Get” na página 1081
- “Método Open” na página 1081
- “Método Put” na página 1082

Acesso à propriedade

Se o objeto da fila não estiver conectado a um gerenciador de filas, será possível ler as seguintes propriedades:

- “propriedade CompletionCode” na página 1071
- “Propriedade OpenStatus” na página 1076
- “propriedade ReasonCode” na página 1077

e será possível ler e gravar em:

- “Propriedade AlternateUserId” na página 1070
- “Propriedade CloseOptions” na página 1071
- “Propriedade ConnectionReference” na página 1071
- “Propriedade Name” na página 1075
- “Propriedade OpenOptions” na página 1076

Se o objeto de fila estiver conectado a um gerenciador de filas, será possível ler todas as propriedades.

Propriedades do atributo da fila

Propriedades não listadas na seção anterior são todos os atributos da fila subjacente do WebSphere MQ . Eles podem ser acessados apenas se o objeto for conectado a um gerenciador de filas e o ID de Usuário do usuário estiver autorizado a consultar ou configurar com relação a essa fila. Se um ID do usuário alternativo for configurado e o ID do usuário atual estiver autorizado a usá-lo, o ID do usuário alternativo será verificado para autorização.

A propriedade deve ser adequada para o QueueType fornecido. Consulte Atributos para filas para obter mais informações.

Se essas condições não se aplicarem, o acesso de propriedade irá configurar um CompletionCode de MQCC_FAILED e um dos ReasonCodes a seguir:

- MQRC_CONNECTION_BROKEN
- MQRC_NOT_AUTHORIZED
- MQRC_Q_MGR_NAME_ERROR
- MQRC_Q_MGR_NOT_CONNECTED
- MQRC_SELECTOR_NOT_FOR_TYPE (CompletionCode é MQCC_WARNING)

Abrindo uma fila

A única maneira de criar um objeto MQQueue é usando o método AccessQueue de MQQueueManager ou em Novo. Um objeto MQQueue aberto permanece aberto (OpenStatus=TRUE) até que ele seja fechado ou excluído ou até que o objeto do gerenciador de filas criador seja excluído ou a conexão é perdida ao gerenciador de filas. O valor das propriedades CloseOptions de MQQueue controlam o comportamento da operação de encerramento que ocorre quando o objeto MQQueue é excluído.

O método MQQueueManager AccessQueue abre a fila usando o parâmetro OpenOptions. O método MQQueue.Open abre a fila usando a propriedade OpenOptions. O WebSphere MQ valida o OpenOptions com relação à autorização do usuário como parte do processo de fila aberta

Propriedade AlternateUserId

Leitura/gravação. O ID do usuário alternativo utilizado para validar o acesso à fila quando ela é aberta.

Esta propriedade não pode ser configurado enquanto o objeto está aberto (ou seja, quando IsOpen for TRUE).

Definido em: classe MQQueue

Tipo de dados: sequência de 12 caracteres

Sintaxe: Para obter: *altuser $= MQQueue*.**AlternateUserId** .

Para configurar: *MQQueue.***AlternateUserId** = *altuser $*

Propriedade BackoutRequeueName

Somente leitura. O atributo de MQI BackOutRequeueQName.

Definido em: classe MQQueue

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *backoutrequeename $= MQQueue*.**BackoutRequeueName**

Propriedade BackoutThreshold

Somente leitura. O atributo BackoutThreshold MQI.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- Veja BackoutThreshold (MQLONG).

Sintaxe: Para obter: *backoutthreshold & = MQQueue.***BackoutThreshold**

Propriedade BaseQueueName

Somente leitura. O nome da fila ao qual o alias é resolvido.

Válido apenas para filas de alias.

Definido em: classe MQQueue

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *baseqname $= MQQueue*.**BaseQueueNome**

Propriedade CloseOptions

Leitura/Gravação. As opções utilizadas para controlar o que acontece quando a fila for fechada.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQCO_NONE
- MQCO_DELETE
- MQCO_DELETE_PURGE

MQCO_DELETE e MQCO_DELETE_PURGE são válidos apenas para filas dinâmicas.

Sintaxe: para obter: *closeopt & = MQQueue*.**CloseOptions**

Para configurar: *MQQueue*.**CloseOptions** = *closeopt &*

propriedade CompletionCode

Somente leitura. Retorna o código de conclusão configurados pelo último método ou acesso de propriedade emitidos com relação ao objeto.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

Sintaxe: Para obter: *completioncode & = MQQueue*.**CompletionCode**

Propriedade ConnectionReference

Leitura/gravação. Define o objeto do gerenciador de filas ao qual um objeto da fila pertence. A referência de conexão não pode ser gravada enquanto uma fila está aberta.

Definido em: classe MQQueue

Tipo de dado: MQQueueManager

Valores:

- Uma referência a um objeto do Gerenciador de Filas ativo do WebSphere MQ

Sintaxe: Para configurar: *configure MQQueue*.**ConnectionReference** = *ConnectionReference*

Para obter: *configure ConnectionReference = MQQueue*.**ConnectionReference**

Propriedade CreationDateTime

Somente leitura. Data e hora em que essa fila foi criada.

Definido em: classe MQQueue

Tipo de dado: Variante de tipo 7 (data/hora) ou EMPTY

Sintaxe: Para obter: *datetime = MQQueue*.**CreationDateTime**

Propriedade CurrentDepth

Somente leitura. O número de mensagens que estão atualmente na fila.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *currentdepth & = MQQueue***.CurrentDepth**

Propriedade DefaultInputOpenOption

Somente leitura. Controla o modo que a fila é aberta se o OpenOptions especificar MQOO_INPUT_AS_Q_DEF.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_SHARED

Sintaxe: Para obter: *defaultinop & = MQQueue***.DefaultInputOpenOption**

Propriedade DefaultPersistence

Somente leitura. A persistência padrão para mensagens em uma fila.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *defpersistence & = MQQueue***.DefaultPersistence**

Propriedade DefaultPriority

Somente leitura. A prioridade padrão para mensagens em uma fila.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *defpriority & = MQQueue***.DefaultPriority**

Propriedade DefinitionType

Somente leitura. O tipo de definição de fila.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQQDT_PREDEFINED
- MQQDT_PERMANENT_DYNAMIC
- MQQDT_TEMPORARY_DYNAMIC

Sintaxe: Para obter: *deftype & = MQQueue***.DefinitionType**

Propriedade DepthHighEvent

Somente leitura. O atributo de MQI QDepthHighEvent.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQEVR_DISABLED
- MQEVR_ENABLED

Sintaxe: Para obter: *depthhighevent & = MQQueue.***DepthHighEvento**

Propriedade DepthHighLimit

Somente leitura. O atributo de MQI QDepthHighLimit.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *depthhighlimit & = MQQueue.***DepthHighLimit**

Propriedade DepthLowEvent

Somente leitura. O atributo de MQI QDepthLowEvent.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQEVR_DISABLED
- MQEVR_ENABLED

Sintaxe: Para obter: *depthlowevent & = MQQueue.***DepthLowEvento**

Propriedade DepthLowLimit

Somente leitura. O atributo MQI QDepthLowLimit.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *depthlowlimit & = MQQueue.***DepthLowLimite**

Propriedade DepthMaximumEvent

Somente leitura. O atributo de MQI QDepthMaxEvent.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQEVR_DISABLED
- MQEVR_ENABLED

Sintaxe: Para obter: *depthmaximumevent & = MQQueue.***DepthMaximumEvento**

Propriedade Description

Somente leitura. Uma descrição da fila.

Definido em: classe MQQueue

Tipo de dado: Sequência de 64 caracteres

Sintaxe: Para obter: *description $= MQQueue***Descrição**

Propriedade DynamicQueueName

Leitura/Gravação, somente leitura quando a fila for aberta.

Isso controla o nome da fila dinâmica usado quando uma fila modelo é aberta. Ele pode ser configurado com um curinga pelo usuário como um conjunto de propriedades (apenas quando a fila estiver fechada) ou como um parâmetro para MQQueueManager.AccessQueue().

O nome real da fila dinâmica é localizado pela consulta de QueueName.

Definido em: classe MQQueue

Tipo de dados: sequência de 48 caracteres

Valores:

- Qualquer nome da fila do WebSphere MQ válido

Sintaxe: Para configurar: *MQQueue***.DynamicQueueNome** = *dynamicqueuename $*

Para obter: *dynamicqueuename $ = MQQueue***.DynamicQueueNome**

Propriedade HardenGetBackout

Somente leitura. Se para manter uma contagem de retorno precisa.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQQA_BACKOUT_HARDENED
- MQQA_BACKOUT_NOT HARDENED

Sintaxe: Para obter: *hardengetback & = MQQueue***.HardenGetBackout**

Propriedade InhibitGet

Leitura/gravação. O atributo de MQI InhibitGet.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQQA_GET_INHIBITED
- MQQA_GET_ALLOWED

Sintaxe: Para obter: *getstatus & = MQQueue***.InhibitGet** .

Para configurar: *MQQueue***.InhibitGet** = *getstatus &* .

Propriedade InhibitPut

Leitura/gravação. O atributo InhibitPut MQI.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQQA_PUT_INHIBITED
- MQQA_PUT_ALLOWED

Sintaxe: Para obter: *putstatus & = MQQueue***.InhibitPut**

Para configurar: *MQQueue***.InhibitPut** = *putstatus &*

Propriedade InitiationQueueName

Somente leitura. Nome da fila de iniciação.

Definido em: classe MQQueue

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *initqname $= MQQueue*.**InitiationQueueNome** .

Propriedade IsOpen

Retorna se a fila está aberta.

Somente leitura.

Definido em: classe MQQueue

Tipo de dado: booleano

Valores:

- TRUE (-1)
- FALSE (0)

Sintaxe: Para obter: *open* = *MQQueue*.**IsOpen**

Propriedade MaximumDepth

Somente leitura. Profundidade máxima da fila.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: para obter: *maxdepth & = MQQueue*.**MaximumDepth**

Propriedade MaximumMessageLength

Somente leitura. Comprimento máximo de mensagem permitido em bytes para esta fila.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *maxmlength & = MQQueue*.**MaximumMessageComprimento**

Propriedade MessageDeliverySequence

Somente leitura. Sequência de entrega de mensagens.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQMDS_PRIORITY
- MQMDS_FIFO

Sintaxe: Para obter: *messdelseq & = MQQueue*.**MessageDeliverySequência**

Propriedade Name

Leitura/gravação. O atributo MQI Queue. Esta propriedade não pode ser gravada após o MQQueue estar aberto.

Definido em: classe MQQueue

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *name $= MQQueue*.**name**

Para configurar: *MQQueue*.**name** = *name $*

Nota: O Visual Basic reserva a propriedade "Name" para uso na interface visual. Portanto, ao usar no Visual Basic, utilize letras minúsculas, ou seja, "name".

Propriedade ObjectHandle

Somente leitura. A manipulação do objeto para o objeto da fila do WebSphere MQ

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: para obter: *hobj & = MQQueue.***ObjectHandle**

Propriedade OpenInputCount

Somente leitura. Número de aberturas para entrada.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *openincount & = MQQueue***.OpenInputCount**

Propriedade OpenOptions

Leitura/gravação. Opções a serem usadas para abrir a fila.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- Consulte OpenOptions (MQLONG).

Sintaxe: Para obter: *openopt & = MQQueue***.OpenOptions** .

Para configurar: *MQQueue.***OpenOptions** = *openopt &*

Propriedade OpenOutputCount

Somente leitura. Número de aberturas para saída.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *openoutcount & = MQQueue***.OpenOutputCount**

Propriedade OpenStatus

Somente leitura. Indica se a fila for aberta ou não. O valor inicial é TRUE após o método AccessQueue ou FALSE após New.

Definido em: classe MQQueue

Tipo de dado: booleano

Valores:

- TRUE (-1)
- FALSE (0)

Sintaxe: Obter: *status & = MQQueue***.OpenStatus**

Propriedade ProcessName

Somente leitura. O atributo ProcessName de MQI.

Definido em: classe MQQueue

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *procname $ =MQQueue***.ProcessName**

Propriedade QueueManagerName

Leitura/gravação. O nome do gerenciador de filas do WebSphere MQ .

Definido em: classe MQQueue

Tipo de dados: sequência

Sintaxe: Para obter: *QueueManagerName$ = MQQueue***.QueueManagerName**

Para configurar: *MQQueue***.QueueManagerName** = *QueueManagerName$*

Propriedade QueueType

Somente leitura. O atributo QType MQI.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQQT_ALIAS
- MQQT_LOCAL
- MQQT_MODEL
- MQQT_REMOTE

Sintaxe: Para obter: *queuetype & = MQQueue***.QueueType**

propriedade ReasonCode

Somente leitura. Retorna o código de razão definido pelo último método ou acesso de propriedade emitidos com relação ao objeto.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- Consulte Códigos de razão de API.

Sintaxe: Para obter: *reasoncode & = MQQueue***.ReasonCode**

Propriedade ReasonName

Somente leitura. Retorna o nome simbólico do código de razão mais recente. Por exemplo, "MQRC_QMGR_NOT_AVAILABLE".

Definido em: classe MQQueue

Tipo de dados: sequência

Valores:

- Consulte Códigos de razão de API.

Sintaxe: Para obter: *reasonname $= MQQueue***.ReasonName**

Propriedade RemoteQueueManagerName

Somente leitura. Nome do gerenciador de filas remoto. Válido apenas para filas remotas.

Definido em: classe MQQueue

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *remqmanname $= MQQueue*.**RemoteQueueManagerName** .

Propriedade RemoteQueueName

Somente leitura. O nome da fila como é conhecido no gerenciador de filas remotas. Válido apenas para filas remotas.

Definido em: classe MQQueue

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *remqname $= MQQueue*.**RemoteQueueNome**

Propriedade ResolvedQueueManagerName

Somente leitura. O nome do gerenciador de filas de destino final como conhecido para o gerenciador de filas locais.

Definido em: classe MQQueue

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *resqmanname $= MQQueue*.**ResolvedQueueManagerName**

Propriedade ResolvedQueueName

Somente leitura. O nome da fila de destino final conforme conhecido para o gerenciador de filas locais.

Definido em: classe MQQueue

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *resqname $= MQQueue*.**ResolvedQueueNome** .

Propriedade RetentionInterval

Somente leitura. O período de tempo pelo qual a fila deve ser retida.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *retinterval & = MQQueue*.**RetentionInterval**

Propriedade Scope

Somente leitura. Controla se uma entrada para esta fila também existe em um diretório de células.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQSCO_Q_MGR
- MQSCO_CELL

Sintaxe: Para obter: *scope & = MQQueue*.**Scope**

Propriedade ServiceInterval

Somente leitura. O MQI atributo QServiceInterval.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *serviceinterval & = MQQueue.***ServiceInterval**

Propriedade ServiceIntervalEvent

Somente leitura. O atributo MQI QServiceIntervalEvent.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQQSIE_HIGH
- MQQSIE_OK
- MQQSIE_NONE

Sintaxe: Para obter: *serviceintervalevent & = MQQueue.***ServiceIntervalEvento**

Propriedade Shareability

Somente leitura. Compartilhamento de Filas.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQQA_SHAREABLE
- MQQA_NOT_SHAREABLE

Sintaxe: Para obter: *shareability & = MQQueue***.Shareability**

Propriedade TransmissionQueueName

Somente leitura. Nome da fila de transmissão. Válido apenas para filas remotas.

Definido em: classe MQQueue

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *transqname $= MQQueue***.TransmissionQueueNome**

Propriedade TriggerControl

Leitura/gravação. Acionador de controle.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQTC_OFF
- MQTC_ON

Sintaxe: Para obter: *trigcontrol & = MQQueue***.TriggerControl**

Para configurar: *MQQueue***.TriggerControl** = *trigcontrol &*

Propriedade TriggerData

Leitura/gravação. Dados do acionador.

Definido em: classe MQQueue

Tipo de dado: Sequência de 64 caracteres

Sintaxe: Para obter: *trigdata $= MQQueue***.TriggerData**

Para configurar: *MQQueue***.TriggerData** = *trigdata $*

Propriedade TriggerDepth

Leitura/gravação. O número de mensagens que precisam estar na fila antes de uma mensagem do acionador ser gravada.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *trigdepth & = MQQueue***.TriggerDepth** .

Para configurar: *MQQueue***.TriggerDepth** = *trigdepth &*

Propriedade TriggerMessagePriority

Leitura/gravação. Prioridade da mensagem limite para acionadores.

Definido em: classe MQQueue

Tipo de dados: longo

Sintaxe: Para obter: *trigmesspriority & = MQQueue***.TriggerMessagePriority**

Para configurar: *MQQueue***.TriggerMessagePriority** = *trigmesspriority &*

Propriedade TriggerType

Leitura/gravação. Tipo de acionador.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQTT_NONE
- MQTT_FIRST
- MQTT_EVERY
- MQTT_DEPTH

Sintaxe: Para obter: *trigtype & = MQQueue***.TriggerType** .

Para configurar: *MQQueue***.TriggerType** = *Trigtype &*

Propriedade Usage

Somente leitura. Indica para que a fila é utilizada.

Definido em: classe MQQueue

Tipo de dados: longo

Valores:

- MQUS_NORMAL
- MQUS_TRANSMISSION

Sintaxe: Para obter: *use & = MQQueue***Uso**

método ClearErrorCodes

Reconfigura o CompletionCode para MQCC_OK e o ReasonCode para MQRC_NONE para as classes MQQueue e MQSession.

Definido em: classe MQQueue

Sintaxe: Chamar *MQQueue***.ClearErrorCodes ()**

Método Close

Fecha uma fila utilizando os valores atuais dos CloseOptions.

Definido em: classe MQQueue

Sintaxe: Call *MQQueue*.**Fechar ()**

Método Get

Recupera uma mensagem da fila.

Este método toma um objeto MQMessage como parâmetro, usando alguns dos campos do MQMD do objeto como parâmetros de entrada. Em particular, os campos MessageId e CorrelId são usados, então é importante assegurar que esses campos estejam configurados conforme necessário. Para obter mais informações sobre esses campos, consulte MsgId (MQBYTE24) e CorrelId (MQBYTE24) .

Se o método falhar, o objeto MQMessage permanecerá inalterado. Se for bem-sucedido, as partes de MQMD e de Dados da mensagem do objeto MQMessage serão substituídas pelo MQMD e os Dados da mensagem a partir da mensagem recebida. As propriedades de controle de MQMessage serão configuradas como segue

- **MessageLength** é configurado como comprimento da mensagem WebSphere MQ
- **DataLength** é configurado para o comprimento da mensagem WebSphere MQ
- **DataOffset** é configurado como zero

Definido em:
Classe MQQueue

Sintaxe: Call *MQQueue*.**Obter**(*Mensagem, GetMsgOpções, GetMsgComprimento*)

Parâmetros
Mensagem:

Objeto MQMessage representando a mensagem a ser recuperada.

GetMsgOptions:

Objeto MQGetMessageOptions opcional para controlar a operação get. Se esse parâmetro não for especificado, o MQGetMessageOptions padrão será usado.

GetMsgLength:

Valor de comprimento opcional de 2 ou 4 bytes para controlar o comprimento máximo da mensagem do WebSphere MQ que é recuperada da fila.

Se a opção MQGMO_ACCEPT_TRUNCATED_MSG for especificada, o GET é bem-sucedido com código de conclusão de MQCC_WARNING e um código de razão de MQRC_TRUNCATED_MSG_ACCEPTED se o tamanho da mensagem exceder o comprimento especificado.

MessageData contém os primeiros bytes de dados de GetMsgLength.

Se MQGMO_ACCEPT_TRUNCATED_MSG ***não for*** especificado e o tamanho da mensagem exceder o comprimento especificado, o código de conclusão MQCC_FAILED juntamente ao código de razão MQRC_TRUNCATED_MESSAGE_FAILED serão retornados.

Se o conteúdo do buffer de mensagens estiverem indefinidos, o comprimento total da mensagem será configurado para o comprimento total da mensagem que teria sido recuperada.

Se o parâmetro de comprimento da mensagem não for especificado, o comprimento do buffer de mensagem é ajustado automaticamente para pelo menos o tamanho da mensagem de entrada.

Método Open

Abre uma fila usando os valores atuais de:

1. QueueName
2. QueueManagerName
3. AlternateUserId
4. DynamicQueueName

Definido em:
Classe MQQueue

Sintaxe: Call *MQQueue*.**Abrir ()**

Método Put

Coloca uma mensagem na fila.

Este método utiliza um objeto MQMessage como parâmetro. As propriedades do Descritor de Mensagens (MQMD) desse objeto podem ser alteradas como resultado deste método. Os valores que eles possuem imediatamente após esse método ter sido executado são os valores que foram colocados no WebSphere MQ

Modificações no objeto MQMessage após a entrada ter sido concluída não afetam a mensagem real na fila do WebSphere MQ.

Definido em:
Classe MQQueue

Sintaxe: Call *MQQueue*.**Colocar**(*Mensagem, PutMsgOpções)* .

Parâmetros
Mensagem

Um objeto MQMessage que representa a mensagem a ser colocada.

PutMsgOptions

O objeto MQPutMessageOptions contendo opções para controlar a operação de entrada. Se eles não forem especificados, PutMessageOptions padrão são usadas.

Classe MQMessage

Essa classe representa uma mensagem do WebSphere MQ Ele inclui propriedades para encapsular o descritor de mensagens WebSphere MQ (MQMD) e fornece um buffer para conter os dados da mensagem definidos pelo aplicativo.

A classe inclui métodos de gravação para copiar dados a partir de um aplicativo do ActiveX para um objeto do MQMessage. Da mesma forma, a classe inclui métodos de leitura para copiar dados de um objeto do MQMessage para um aplicativo do ActiveX. A classe gerencia a alocação e desalocação de memória para o buffer automaticamente. O aplicativo não tem de declarar o tamanho do buffer quando um objeto MQMessage é criado porque o buffer aumenta para acomodar os dados gravados nele.

Não será possível colocar uma mensagem em uma fila do WebSphere MQ se o tamanho do buffer exceder a propriedade de Comprimento MaximumMessagedessa fila

Depois de ter sido construído, um objeto MQMessage pode ser Colocado em uma fila do WebSphere MQ usando o método MQQueue.Put . Este método usa uma cópia do MQMD e as partes de dados da mensagem do objeto e coloca essa cópia na fila. Portanto, o aplicativo pode modificar ou excluir um objeto MQMessage após o Put, sem afetar a mensagem na fila do WebSphere MQ . O gerenciador de filas pode ajustar alguns dos campos no MQMD ao copiar a mensagem na fila do WebSphere MQ .

Uma mensagem recebida pode ser lida em um objeto MQMessage usando o método MQQueue.Get. Isso substitui quaisquer dados de MQMD ou de mensagem que podem já ter estado no objeto MQMessage com valores da mensagem que chega. Ele ajusta o tamanho do buffer de dados do objeto MQMessage para corresponder ao tamanho dos dados da mensagem de entrada.

Restrição

As mensagens são contidas pela classe MQSession.

Criação

New cria um objeto MQMessage. Suas propriedades do Descritor de mensagens são inicialmente configuradas como valores padrão e seu buffer de dados da mensagem está vazio.

Sintaxe

```
Dim msg As New MQMessage or Set msg = New MQMessage
```

Propriedades

As propriedades de controle são:

- “propriedade CompletionCode” na página 1085
- “Propriedade DataLength” na página 1085
- “Propriedade DataOffset” na página 1086
- “Propriedade MessageLength” na página 1086
- “propriedade ReasonCode” na página 1086
- “Propriedade ReasonName” na página 1086

As propriedades do Descritor de mensagens são:

- “Propriedade AccountingToken” na página 1087
- “Propriedade AccountingTokenHex” na página 1087
- “Propriedade ApplicationIdData” na página 1087
- “Propriedade ApplicationOriginData” na página 1087
- “Propriedade BackoutCount” na página 1087
- “Propriedade CharacterSet” na página 1088
- “Propriedade CorrelationId” na página 1088
- “Propriedade CorrelationIdHex” na página 1088
- “Propriedade Encoding” na página 1089
- “Propriedade Expiry” na página 1090
- “Propriedade Feedback” na página 1090
- “Propriedade Format” na página 1090
- “Propriedade GroupId” na página 1090
- “Propriedade GroupIdHex” na página 1090
- “Propriedade MessageData” na página 1091
- “Propriedade MessageFlags” na página 1091
- “Propriedade MessageId” na página 1091
- “propriedade MessageIdHex” na página 1092
- “Propriedade MessageSequenceNumber” na página 1092
- “Propriedade MessageType” na página 1092
- “Propriedade Offset” na página 1092
- “Propriedade OriginalLength” na página 1093
- “Propriedade Persistence” na página 1093
- “Propriedade Priority” na página 1093
- “Propriedade PutApplicationName” na página 1093
- “Propriedade PutApplicationType” na página 1093

- “Propriedade PutDateTime” na página 1094
- “Propriedade ReplyToQueueManagerName” na página 1094
- “Propriedade ReplyToQueueName” na página 1094
- “Propriedade Report” na página 1094
- “Propriedade TotalMessageLength” na página 1095
- “Propriedade UserId” na página 1095

Methods

- “método ClearErrorCodes” na página 1095
- “Método ClearMessage” na página 1095
- “Método Read” na página 1095
- “Método ReadBoolean” na página 1095
- “Método ReadByte” na página 1096
- “Método ReadDecimal2” na página 1096
- “Método ReadDecimal4” na página 1096
- “Método ReadDouble” na página 1096
- “Método ReadDouble4” na página 1096
- “Método ReadFloat” na página 1097
- “Método ReadInt2” na página 1097
- “Método ReadInt4” na página 1097
- “Método ReadLong” na página 1097
- “Método ReadNullTerminatedString” na página 1097
- “Método ReadShort” na página 1098
- “Método ReadString” na página 1098
- “Método ReadUInt2” na página 1098
- “Método ReadUnsignedByte” na página 1099
- “Método ReadUTF” na página 1099
- “Método ResizeBuffer” na página 1099
- “Método Write” na página 1099
- “Método WriteBoolean” na página 1100
- “Método WriteByte” na página 1100
- “Método WriteDecimal2” na página 1100
- “Método WriteDecimal4” na página 1100
- “Método WriteDouble” na página 1100
- “Método WriteDouble4” na página 1101
- “Método WriteFloat” na página 1101
- “Método WriteInt2” na página 1101
- “Método WriteInt4” na página 1101
- “Método WriteLong” na página 1102
- “Método WriteNullTerminatedString” na página 1102
- “Método WriteShort” na página 1102
- “Método WriteString” na página 1102
- “Método WriteUInt2” na página 1103

- “Método WriteUnsignedByte” na página 1103
- “Método WriteUTF” na página 1103

Acesso à propriedade

Todas as propriedades podem ser lidas a qualquer momento.

As propriedades de controle são somente leitura, exceto para DataOffset que são de leitura/gravação. As propriedades do Descritor de mensagens são todas de leitura/gravação, exceto BackoutCount e TotalMessageLength, que são somente leitura.

No entanto, observe que algumas das propriedades do MQMD podem ser modificadas pelo gerenciador de fila quando a mensagem é colocada em uma fila do WebSphere MQ . Consulte os campos no MQMD para obter detalhes de como eles podem ser modificados.

Conversão de Dados

É possível transmitir dados binários para uma mensagem do WebSphere MQ configurando a propriedade CharacterSet para o Identificador do conjunto de caracteres codificados do gerenciador de filas (MQCCSI_Q_MGR) e transmitindo uma sequência. É possível usar a função chr $para configurar dados não de caractere na sequência.

Os métodos de leitura e gravação executam a conversão de dados. Eles convertem entre os formatos internos ActiveX e os formatos de mensagens WebSphere MQ conforme definido pelas propriedades Encoding e CharacterSet do descritor de mensagens. Ao gravar uma mensagem, configure os valores de Encoding e CharacterSet que correspondem às características do destinatário da mensagem antes de emitir um método Write. Ao ler uma mensagem, esta etapa não é normalmente necessária porque esses valores terão sido configurados a partir daqueles no MQMD recebido.

Esta é uma etapa de conversão de dados adicionais que ocorre após qualquer conversão executada pelo método MQQueue.Get.

propriedade CompletionCode

Somente leitura. Retorna o código de conclusão do WebSphere MQ configurado pelo acesso de método ou de propriedade mais recente emitido com relação a esse objeto

Definido em: classe MQMessage

Tipo de dados: longo

Valores:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

Sintaxe: Para obter: *completioncode & = MQMessage*.**CompletionCode**

Propriedade DataLength

Somente leitura. Essa propriedade retorna o valor:

```
MQMessage.MessageLength - MQMessage.DataOffset
```

Ele pode ser usado antes de um método Read para verificar se o número esperado de caracteres está realmente presente no buffer.

O valor inicial é zero.

Definido em: classe MQMessage

Tipo de dados: longo

Sintaxe: Para obter: *bytesleft & = MQMessage*.**DataLength**

Propriedade DataOffset

Leitura/gravação. A posição atual na parte Dados da mensagem do objeto de mensagem.

O valor é expresso como um deslocamento de bytes desde o início do buffer de dados da mensagem; o primeiro caractere no buffer corresponde a um valor DataOffset de zero.

Um método de leitura ou gravação inicia sua operação no caractere referenciado pelo DataOffset. Esses métodos processam dados no buffer sequencialmente a partir dessa posição e atualizam o DataOffset para apontar para o byte (se houver) imediatamente após o último byte processado.

DataOffset pode aceitar somente valores no intervalo zero a MessageLength inclusive. Quando DataOffset = MessageLength, ele está apontando para o final, que é o primeiro caractere inválido do buffer. Métodos de gravação são permitidos nesta situação - eles estendem os dados no buffer e aumentam o MessageLength no número de bytes incluídos. Ler além do final do buffer não é válido.

O valor inicial é zero.

Definido em: classe MQMessage

Tipo de dados: longo

Sintaxe: Para obter: *currpos & = MQMessage*.**DataOffset**

Para configurar: *MQMessage*.**DataOffset** = *currpos &*

Propriedade MessageLength

Somente leitura. Retorna o comprimento total da parte Dados da Mensagem do objeto de mensagem em caracteres, independentemente do valor de DataOffset.

O valor inicial é zero. É definido para inserir o Comprimento da Mensagem após uma chamada de método Get que referenciou este objeto de mensagem. Ele é incrementado se o aplicativo usa um método de gravação para incluir dados no objeto. Ele não é afetado por métodos Read.

Definido em: classe MQMessage

Tipo de dados: longo

Sintaxe: Para obter: *msglength & = MQMessage*.**MessageLength** .

propriedade ReasonCode

Somente leitura. Retorna o código de razão definido pelo método mais recente ou acesso de propriedade emitidos para este objeto.

Definido em: classe MQMessage

Tipo de dados: longo

Valores:

- Consulte Códigos de razão de API.

Sintaxe: para obter: *reasoncode & = MQMessage*.**ReasonCode**

Propriedade ReasonName

Somente leitura. Retorna o nome simbólico do código de razão mais recente. Por exemplo, "MQRC_QMGR_NOT_AVAILABLE". **Definido em:** classe MQMessage

Tipo de dados: sequência

Valores:

- Consulte Códigos de razão de API.

Sintaxe: Para obter: *reasonname $= MQMessage*.**ReasonName** .

Propriedade AccountingToken

Leitura/gravação. O AccountingToken MQMD – parte do Contexto de identidade da mensagem.

Seu valor inicial é todos nulos.

Definido em: classe MQMessage

Tipo de dado: sequência de 32 caracteres

Sintaxe: Para obter: *actoken $= MQMessage*.**AccountingToken**

Para configurar: *MQMessage.***AccountingToken** = *actoken $*

Consulte “Propriedades do Descritor de Mensagens” na página 1047 para obter mais informações sobre quando deve-se usar AccountingTokenHex no lugar da propriedade AccountingToken.

Propriedade AccountingTokenHex

Leitura/gravação. O AccountingToken MQMD – parte do Contexto de identidade da mensagem.

Cada dois caracteres representam o equivalente hexadecimal de um caractere ASCII único. Por exemplo, o par de caracteres "6" e "1" representa o caractere único "A", o par de caracteres "6" e "2" representa o caractere único "B" e assim por diante.

Deve-se fornecer 64 caracteres hexadecimais válidos.

Seu valor inicial é "0...0"

Definido em: classe MQMessage

Tipo de dados: sequência de 64 caracteres hexadecimais representando 32 caracteres ASCII

Sintaxe: Para obter: *actokenh $= MQMessage*.**AccountingTokenHex**

Para configurar: *MQMessage.***AccountingTokenHex** = *actokenh $*

Consulte “Propriedades do Descritor de Mensagens” na página 1047 para obter mais informações sobre quando deve-se usar AccountingTokenHex no lugar da propriedade AccountingToken.

Propriedade ApplicationIdData

Leitura/gravação. O MQMD ApplIdentityData - parte do Contexto de Identidade da mensagem.

Seu valor inicial é todos os espaços em branco.

Definido em: classe MQMessage

Tipo de dado: sequência de 32 caracteres

Sintaxe: Para obter: *applid $= MQMessage*.**ApplicationIdDados**

Para configurar: *MQMessage*.**ApplicationIdData** = *applid $*

Propriedade ApplicationOriginData

Leitura/gravação. O MQMD ApplOriginData - parte do contexto de origem da mensagem.

Seu valor inicial é todos os espaços em branco.

Definido em: classe MQMessage

Tipo de dado: Sequência de 4 caracteres

Sintaxe: Para obter: *applor $= MQMessage*.**ApplicationOriginData**

Para configurar: *MQMessage*.**ApplicationOriginData** = *applor $*

Propriedade BackoutCount

Somente leitura. O BackoutCount de MQMD.

Seu valor inicial é 0

Definido em: classe MQMessage

Tipo de dados: longo

Sintaxe: Para obter: *backoutct & = MQMessage*.**BackoutCount**

Propriedade CharacterSet

Leitura/gravação. O MQMD CodedCharSetId.

Seu valor inicial é o valor especial **MQCCSI_Q_MGR** .

Se o CharacterSet for definido como **MQCCSI_Q_MGR**, a página de códigos para o código de idioma atual será utilizada para conversão de caracteres no método WriteString Para aplicativos do servidor, a página de códigos usada é a página de códigos do gerenciador de filas. Para aplicativos clientes, é a página de códigos do idioma atual padrão.

Por exemplo:

```
msg.CharacterSet = MQCCSI_Q_MGR
msg.WriteString(chr$(n))
```

em que 'n' é maior ou igual a zero e menor ou igual a 255, resulta na gravação de um único byte de valor 'n' no buffer.

Definido em: classe MQMessage

Tipo de dados: longo

Sintaxe: Para obter: *:30ccid& = MQMessage* **.CharacterSet**

Para configurar: *MQMessage*.**CharacterSet**= *ccid &*

exemplo

Se desejar que a sequência seja gravada na página de códigos 437, emita:

```
Message.CharacterSet = 437
Message.WriteString ("string to be written")
```

Configure o valor que deseja em CharacterSet antes de emitir quaisquer chamadas WriteString.

Propriedade CorrelationId

Leitura/gravação. O CorrelationId a ser incluído no MQMD de uma mensagem quando colocada em uma fila. Além disso, o ID a ser correspondido em relação ao obter uma mensagem de uma fila.

Seu valor inicial é nulo.

Definido em: classe MQMessage

Tipo de dado: sequência de 24 caracteres

Sintaxe: Para obter: *correlid $= MQMessage*.**CorrelationId** Para configurar: *MQMessage*.**CorrelationId** = *correlid $*

Veja “Propriedades do Descritor de Mensagens” na página 1047 para obter mais informações sobre quando se deve usar CorrelationIdHex em vez da propriedade CorrelationId.

Propriedade CorrelationIdHex

Leitura/gravação. O CorrelationId a ser incluído no MQMD de uma mensagem quando colocada em uma fila. Além disso, o CorrelationId a ser correspondido em relação ao obter uma mensagem de uma fila.

Cada dois caracteres da sequência reprsentam o equivalente hexadecimal de um único caractere ASCII. Por exemplo, o par de caracteres "6" e "1" representa o caractere único "A", o par de caracteres "6" e "2" representa o caractere único "B" e assim por diante.

Deve-se fornecer 48 caracteres hexadecimais válidos.

Seu valor inicial é "0...0".

Definido em: classe MQMessage

Tipo de dados: sequência de 48 caracteres hexadecimais representando 24 caracteres ASCII

Sintaxe: Para obter: *correlidh $= MQMessage*.**CorrelationIdHex**

Para configurar: *MQMessage*.**CorrelationIdHex** = *correlidh $*

Veja “Propriedades do Descritor de Mensagens” na página 1047 para obter uma discussão de quando se deve usar CorrelationIdHex em vez da propriedade CorrelationId.

Propriedade Encoding

Leitura/gravação. O campo MQMD que identifica a representação usada para valores numéricos nos dados de mensagem do aplicativo.

Seu valor inicial é o valor especial MQENC_NATIVE, que varia por plataforma.

Essa propriedade é usada pelos métodos a seguir:

- Método ReadDecimal2
- Método ReadDecimal4
- Método ReadDouble
- Método ReadDouble4
- Método ReadFloat
- Método ReadInt2
- Método ReadInt4
- Método ReadLong
- Método ReadShort
- Método ReadUInt2
- Método WriteDecimal2
- Método WriteDecimal4
- Método WriteDouble
- Método WriteDouble4
- Método WriteFloat
- Método WriteInt2
- Método WriteInt4
- Método WriteLong
- Método WriteShort
- Método WriteUInt2

Definido em: classe MQMessage

Tipo de dados: longo

Sintaxe: Para obter: *encoding & = MQMessage***Codificação** Para configurar: *MQMessage*.**Codificação** = *codificação &*

Se estiver se preparando para gravar dados no buffer de mensagem, é necessário configurar esse campo para que corresponda às características da plataforma do gerenciador de filas de recebimento se o gerenciador de filas de recebimento for incapaz de executar sua própria conversão de dados.

Propriedade Expiry

Leitura/gravação. O campo de tempo de expiração do MQMD, esperado em décimos de segundo.

Seu valor inicial é o valor especial MQEI_UNLIMITED

Definido em: classe MQMessage

Tipo de dados: longo

Sintaxe: Para obter: *expirar & = MQMessage***Expiração**

Para configurar: *MQMessage*.**Expiração** = *expiração &*

Propriedade Feedback

Leitura/gravação. O campo feedback MQMD.

Seu valor inicial é o valor especial MQFB_NONE.

Definido em: classe MQMessage

Tipo de dados: longo

Valores:

- Veja Feedback.

Sintaxe: Para obter: *feedback & = MQMessage***Feedback**

Para configurar: *MQMessage*.**Feedback** = *feedback &*

Propriedade Format

Leitura/gravação. O campo de formato MQMD. Fornece o nome de um formato integrado ou definido pelo usuário que descreve a natureza dos dados da mensagem.

Seu valor inicial é o valor especial MQFMT_NONE.

Definido em: classe MQMessage

Tipo de dados: Sequência de 8 caracteres

Sintaxe: Para obter: *format $= MQMessage***.Formato**

Para configurar: *MQMessage*.**Formato** = *formato $*

Propriedade GroupId

Leitura/gravação. O GroupId a ser incluído na MQPMR de uma mensagem quando colocar em uma fila. Além disso, o ID a ser correspondido em relação ao obter uma mensagem de uma fila. Seu valor inicial é todos nulos.

Definido em:
Classe MQMessage

Tipo de dados:
Sequência de 24 caracteres

Sintaxe: Para obter: *groupid $= MQMessage*.**GroupId**

Para configurar: *MQMessage*.**GroupId** = *groupid $*

Veja “Propriedades do Descritor de Mensagens” na página 1047 para obter mais informações sobre quando se deve usar GroupIdHex em vez da propriedade GroupId.

Propriedade GroupIdHex

Leitura/gravação. O GroupId a ser incluído na MQPMR de uma mensagem quando colocar em uma fila. Além disso, o ID a ser correspondido em relação ao obter uma mensagem de uma fila.

Cada dois caracteres da sequência reprsentam o equivalente hexadecimal de um único caractere ASCII. Por exemplo, o par de caracteres "6" e "1" representam o caractere único "A", o par de caracteres "6" e "2" representam o caractere único "B" e assim por diante.

Deve-se fornecer 48 caracteres hexadecimais válidos.

Seu valor inicial é "0...0".

Definido em:
Classe MQMessage

Tipo de dados:
Sequência de 48 caracteres hexadecimais que representam 24 caracteres ASCII.

Sintaxe: Para obter: *groupidh $= MQMessage*.**GroupIdHex**

Para configurar: *MQMessage*.**GroupIdHex** = *groupidh $*

Veja “Propriedades do Descritor de Mensagens” na página 1047 para obter mais informações sobre quando se deve usar GroupIdHex em vez da propriedade GroupId.

Propriedade MessageData

Leitura/gravação. Recupera ou define todo o conteúdo de uma mensagem como uma sequência de caracteres.

Definido em: classe MQMessage

Data Type: Variant

Nota: O tipo de dados usado por esta propriedade é variante, mas o MQAX espera que esta seja um tipo variante de sequência. Se você passar em uma variante de tipo diferente desta, o erro MQRC_OBJECT_TYPE_ERROR será retornado.

Sintaxe: Obter: *String$* = *MQMessage*.**MessageData**

Para configurar: *MQMessage*.**MessageData** = *String$*

Propriedade MessageFlags

Leitura/Gravação. Sinalizações de mensagem especificando informações de controle de segmentação. O valor inicial é 0.

Definido em:
Classe MQMessage

Tipo de dados:
Long

Valores:
Veja MsgFlags (MQLONG).

Sintaxe: Para obter: *messageflag & = MQMessage*.**MessageFlags** .

Para configurar: *MQMessage*.**MessageFlags** = *messageflags &*

Propriedade MessageId

Leitura/gravação. O MessageId a ser incluído no MQMD de uma mensagem quando colocada em uma fila. Além disso, o ID a ser correspondido em relação ao obter uma mensagem de uma fila.

Seu valor inicial é todos nulos.

Definido em: classe MQMessage

Tipo de dado: sequência de 24 caracteres

Sintaxe: Para obter: *messageid $= MQMessage*.**MessageId** .

Para configurar: *MQMessage*.**MessageId** = *messageid $*

Consulte “Propriedades do Descritor de Mensagens” na página 1047 para obter mais informações sobre quando deve-se usar MessageIdHex no lugar da propriedade MessageId.

propriedade MessageIdHex

Leitura/gravação. O MessageId a ser incluído no MQMD de uma mensagem quando colocada em uma fila. Além disso, o MessageId a ser correspondido em relação a quando obter uma mensagem de uma fila.

Cada dois caracteres da sequência reprsentam o equivalente hexadecimal de um único caractere ASCII. Por exemplo, o par de caracteres "6" e "1" representa o caractere único "A", o par de caracteres "6" e "2" representa o caractere único "B" e assim por diante.

Deve-se fornecer 48 caracteres hexadecimais válidos.

Seu valor inicial é "0...0".

Definido em: classe MQMessage

Tipo de dados: sequência de 48 caracteres hexadecimais representando 24 caracteres ASCII

Sintaxe: Para obter: *messageidh $= MQMessage*.**MessageIdHex**

Para configurar: *MQMessage*.**MessageIdHex** = *messageidh $*

Consulte “Propriedades do Descritor de Mensagens” na página 1047 para obter mais informações sobre quando deve-se usar MessageIdHex no lugar da propriedade MessageId.

Propriedade MessageSequenceNumber

Leitura/Gravação. Informações de sequência que identificam uma mensagem em um grupo. O valor inicial é 1.

Definido em:
Classe MQMessage

Tipo de dados:
Long

Valores:
Consulte MsgSeqNumber (MQLONG).

Sintaxe: Para obter: *sequencenumber & = MQMessage*.**SequenceNumber**

Para configurar: *MQMessage*.**SequenceNumber** = *sequencenumber &*

Propriedade MessageType

Leitura/gravação. O campo MQMD MsgType.

Seu valor inicial é MQMT_DATAGRAM.

Definido em: classe MQMessage

Tipo de dados: longo

Valores:

- Consulte MsgType (MQLONG).

Sintaxe: Para obter: *msgtype & = MQMessage*.**MessageType**

Para configurar: *MQMessage*.**MessageType** = *msgtype &*

Propriedade Offset

Leitura/Gravação. O deslocamento em uma mensagem segmentada. O valor inicial é 0.

Definido em:
Classe MQMessage

Tipo de dados:
Long

Valores:
Veja Offset (MQLONG).

Sintaxe: Para obter: *offset & = MQMessage*.**Offset**

Para configurar: *MQMessage*.**Offset** = *offset &*

Propriedade OriginalLength

Leitura/Gravação. O comprimento original de uma mensagem segmentada. O valor inicial é MQOL_UNDEFINED.

Definido em:
Classe MQMessage

Tipo de dados:
Long

Valores:
Veja OriginalLength (MQLONG).

Sintaxe: para obter: *originallength & = MQMessage*.**OriginalLength**

Para configurar: *MQMessage*.**OriginalLength** = *originallength &*

Propriedade Persistence

Leitura/gravação. A configuração de persistência da mensagem.

Seu valor inicial é MQPER_PERSISTENCE_AS_Q_DEF.

Definido em: classe MQMessage

Tipo de dados: longo

Sintaxe: para obter: *persist & = MQMessage***.Persistência**

Para configurar: *MQMessage*.**Persistência** = *persistente &*

Propriedade Priority

Leitura/gravação. A prioridade da mensagem.

Seu valor inicial é o valor especial MQPRI_PRIORITY_AS_Q_DEF

Definido em: classe MQMessage

Tipo de dados: longo

Sintaxe: para obter: *priority & = MQMessage***Prioridade**

Para configurar: *MQMessage*.**Prioridade** = *prioridade &*

Propriedade PutApplicationName

Leitura/gravação. O MQMD PutApplName - parte do contexto de Origem da Mensagem.

Seu valor inicial é todos os espaços em branco.

Definido em: classe MQMessage

Tipo de dado: Sequência de 28 caracteres

Sintaxe: Para obter: *putapplnm $= MQMessage***.PutApplicationNome**

Para configurar: *MQMessage***.PutApplicationName** = *putapplnm $*

Propriedade PutApplicationType

Leitura/gravação. O MQMD PutApplType – parte do contexto de origem da mensagem.

Seu valor inicial é MQAT_NO_CONTEXT

Definido em: classe MQMessage

Tipo de dados: longo

Valores:

- Consulte PutApplType (MQLONG).

Sintaxe: Para obter: *putappltp & = MQMessage*.**PutApplicationTipo**

Para configurar: *MQMessage*.**PutApplicationType** = *putappltp &*

Propriedade PutDateTime

Leitura/gravação. Esta propriedade combina os campos MQMD PutDate e PutTime. Esses são partes do contexto de Origem de mensagens que indicam quando a mensagem foi colocada.

A extensão ActiveX converte entre o formato de data / hora ActiveX e os formatos de Data e Hora usados em um MQMD WebSphere MQ . Se for recebida uma mensagem que possui um PutDate ou PutTime inválido, a propriedade PutDateTime após o método get é configurada como EMPTY.

Seu valor inicial é EMPTY.

Definido em: classe MQMessage

Tipo de Dado: Variante de tipo 7 (data/hora) ou EMPTY.

Sintaxe: Para obter: *datetime = MQMessage*.**PutDateTime**

Para configurar: *MQMessage*.**PutDateTime** = *datetime*

Propriedade ReplyToQueueManagerName

Leitura/gravação. O campo MQMD ReplyToQMgr.

Seu valor inicial é todos os espaços em branco

Definido em: classe MQMessage

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *replytoqmgr $= MQMessage*.**ReplyToQueueManagerNome** .

Para configurar: *MQMessage*.**ReplyToQueueManagerNome** = *replytoqmgr $*

Propriedade ReplyToQueueName

Leitura/gravação. O campo MQMD ReplyToQ.

Seu valor inicial é todos os espaços em branco

Definido em: classe MQMessage

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *replytoq $= MQMessage*.**ReplyToQueueName**

Para configurar: *MQMessage*.**ReplyToQueueName** = *replytoq $*

Propriedade Report

Leitura/gravação. As opções de relatório da mensagem.

Seu valor inicial é MQRO_NONE.

Definido em: classe MQMessage

Tipo de dados: longo

Valores:

- Consulte Relatório.

Sintaxe: Para obter: *report & = MQMessage***Relatório**

Para configurar: *MQMessage*.**Relatório** = *relatório &*

Propriedade TotalMessageLength

Somente leitura. Recupera o comprimento da última mensagem recebida pela chamada MQGET. Se a mensagem não foi truncada, este valor é igual ao valor da propriedade MessageLength.

Definido em: classe MQMessage

Tipo de dados: longo

Sintaxe: Para obter: *totalmessagelength & = MQMessage***.TotalMessageComprimento**

Propriedade UserId

Leitura/gravação. O MQMD UserIdentifier - parte do Contexto de Identidade da mensagem.

Seu valor inicial é todos os espaços em branco.

Definido em: classe MQMessage

Tipo de dados: sequência de 12 caracteres

Sintaxe: Para obter: *userid $= MQMessage***.UserId**

Para configurar: *MQMessage***.UserId** = *userid $*

método ClearErrorCodes

Reconfigura o CompletionCode para MQCC_OK e MQRC_NONE para ReasonCode para a classe MQMessage e a classe MQSession.

Definido em: classe MQMessage

Sintaxe: Chamar *MQMessage***.ClearErrorCodes ()**

Método ClearMessage

Esse método limpa a parte do buffer de dados do objeto MQMessage. Quaisquer Dados de Mensagem no buffer de dados que forem perdidos, como MessageLength, DataLength e DataOffset, são todos configurados para zero.

A parte do Descritor de mensagens (MQMD) não é afetada; um aplicativo pode precisar modificar alguns dos campos do MQMD antes de reutilizar o objeto MQMessage. Para definir os campos de MQMD, use Novo de volta para substituir o objeto por uma nova instância.

Definido em: classe MQMessage

Sintaxe: Chamar *MQMessage***.ClearMessage()**

Método Read

Lê uma sequência de bytes do buffer de mensagem em uma matriz de bytes. O DataOffset é aumentado e o Data Lenght diminuído no número de bytes lidos.

Definido em:
Classe MQMessage

Sintaxe: Dados = MQMessage.**Read**(len &)

Parâmetros:
len &: longo. Comprimento de dados em bytes a serem lidos.

Método ReadBoolean

Lê um valor booleano de 1 byte da posição atual no buffer de mensagem e retorna um valor booleano de 2 bytes TRUE(-1)/FALSE(0). DataOffset é incrementado em um e dados de Comprimento será diminuído por um.

Definido em:
Classe MQMessage

Sintaxe: *value = MQMessage*.**ReadBoolean**

Método ReadByte

Esse método lê 1 byte do buffer de Dados da Mensagem, iniciando com o caractere referido por DataOffset e retorna como um valor de número inteiro Integer (2-byte assinado) no intervalo de -128 a 127.

O método falha se MQMessage.DataLength for inferior a 1 quando for emitido.

DataOffset é incrementado em 1 e DataLength é decrementado em 1 se o método for bem-sucedido.

O byte de dados da mensagem é assumido para ser um número inteiro binário assinado.

Definido em: classe MQMessage

Sintaxe: *integerv% = MQMessage*.**ReadByte**

Método ReadDecimal2

Lê um número decimal compactado de 2 bytes e retorna-o como um valor de número inteiro de 2 bytes assinado. DataOffset é incrementado por dois e Data Length é reduzida por dois.

Definido em:
Classe MQMessage

Sintaxe: *value% = MQMessage*.**ReadDecimal2**

Método ReadDecimal4

Lê um número decimal compactado de 4 bytes e retorna-o como um valor de número inteiro de 4 bytes assinado. DataOffset é incrementado em quatro e Dados Comprimento será diminuído por quatro.

Definido em:
Classe MQMessage

Sintaxe: Valor de *chamada & = MQMessage*.**ReadDecimal4**

Método ReadDouble

Esse método lê 8 bytes do buffer de Dados da Mensagem, começando com o byte referido por DataOffset e retorna como um valor de ponto flutuante duplo (8 bytes assinados).

O método falha se MQMessage.DataLength for menor que 8 quando ele é emitido.

DataOffset é incrementado em 8 e DataLength é decrementado em 8 se o método for bem-sucedido.

Os 8 caracteres de dados da mensagem são considerados como sendo um número de vírgula flutuante binário. A codificação é especificada pela propriedade MQMessage.Encoding. Observe que a conversão do formato System/360 não é suportada.

Definido em: classe MQMessage

Sintaxe: *doublev# = MQMessage*.**ReadDouble**

Método ReadDouble4

Os métodos ReadDouble4 e WriteDouble4 são alternativas para ReadFloat e WriteFloat. Isso ocorre porque eles suportam valores de mensagens de ponto flutuante System/390 de 4 bytes que são muito grandes para converter em formato de ponto flutuante IEEE de 4 bytes.

Este método lê 4 bytes do buffer de Dados da Mensagem, começando com o byte referido por DataOffset e retorna como um valor de ponto flutuante duplo (8 bytes assinados).

O método falhará se MQMessage.DataLength for menor que 4 quando emitido.

DataOffset será incrementado em 4 e DataLength será decrementado em 4 se o método for bem-sucedido.

Os 4 caracteres de dados da mensagem são considerados como sendo um número de vírgula flutuante binário. A codificação é especificada pela propriedade MQMessage.Encoding. Observe que a conversão do formato System/360 não é suportada.

Definido em: classe MQMessage

Sintaxe: *doublev# = MQMessage.***ReadDouble4**

Método ReadFloat

Esse método lê 4 bytes do buffer de Dados da Mensagem, começando com o byte referido por DataOffset e retorna como um valor de ponto flutuante único (4 bytes assinados).

O método falhará se MQMessage.DataLength for menor que 4 quando emitido.

DataOffset será incrementado em 4 e DataLength será decrementado em 4 se o método for bem-sucedido.

Os 4 caracteres de dados da mensagem são considerados como sendo um número de vírgula flutuante. A codificação é especificada pela propriedade MQMessage.Encoding. Observe que a conversão do formato System/360 não é suportada.

Definido em: classe MQMessage

Sintaxe: *singlev! = MQMessage***.ReadFloat**

Método ReadInt2

O método é idêntico ao método ReadShort.

Sintaxe: *integerv% = MQMessage***.ReadInt2**

Método ReadInt4

Este método é idêntico ao método ReadLong.

Sintaxe: *bigint & = MQMessage***.ReadInt4**

Método ReadLong

Esse método lê 4 bytes do buffer de Dados da Mensagem, começando com o byte referido por DataOffset e retorna como um valor de número inteiro longo (4 bytes assinados).

O método falhará se MQMessage.DataLength for menor que 4 quando emitido.

DataOffset será incrementado em 4 e DataLength será decrementado em 4 se o método for bem-sucedido.

Os quatro caracteres de dados da mensagem são considerados como sendo um número inteiro binário. A codificação é especificada pela propriedade MQMessage.Encoding.

Definido em: classe MQMessage

Sintaxe: *bigint & = MQMessage***.ReadLong**

Método ReadNullTerminatedString

Esse método é para ser usado no lugar de ReadString se a sequência possivelmente contiver caracteres nulos integrados.

Esse método lê o número especificado de bytes do buffer de dados de mensagem começando com o byte referido por DataOffset e retorna o mesmo como uma sequência do ActiveX. Se a sequência contiver um nulo integrado antes do fim, então, o comprimento da sequência retornada é reduzido para refletir somente os caracteres antes do nulo.

DataOffset é incrementado e DataLength é diminuído pelo valor especificado, independentemente de se a sequência contém caracteres nulos integrados.

Os caracteres nos dados da mensagem são considerados como uma sequência na página de códigos especificada pela propriedade MQMessage.CharacterSet. A conversão para representação do ActiveX é executada para o aplicativo.

Definido em:
Classe MQMessage

Sintaxe: *string $ = MQMessage*.**ReadNullTerminatedString(length &)**

Parâmetros:
comprimento & *Longo.* O comprimento do campo de sequência em bytes.

Método ReadShort

Este método lê 2 bytes do buffer de Dados da Mensagem, iniciando com o byte referido por DataOffset e retorna-o como um valor Integer (assinado 2 bytes).

O método falhará se MQMessage.DataLength for menor que 2 quando emitido.

DataOffset será incrementado em 2 e DataLength será decrementado em 2 se o método for bem-sucedido.

Os dois caracteres de dados da mensagem são considerados como sendo um número inteiro binário. A codificação é especificada pela propriedade MQMessage.Encoding.

Definido em: classe MQMessage

Sintaxe: *integerv% = MQMessage***.ReadShort**

Método ReadString

Esse método lê n bytes do buffer de Dados de mensagem começando com o byte referido por DataOffset e o retorna como uma sequência do ActiveX.

O método falha se MQMessage.DataLength for menor que n quando emitido.

DataOffset é incrementado por n e DataLength é reduzido por n se o método for bem-sucedido.

Os n caracteres dos dados da mensagem são considerados como uma sequência na página de códigos especificada pela propriedade MQMessage.CharacterSet. A conversão para representação do ActiveX é executada para o aplicativo.

Definido em: classe MQMessage

Sintaxe: *stringv $= MQMessage***.ReadString***(comprimento &*)

Parâmetro
length & longo. O comprimento do campo de sequência em bytes.

Método ReadUInt2

Esse método lê 2 bytes do buffer de Dados da Mensagem, começando com o byte referido por DataOffset e retorna como um valor de número inteiro longo (4 bytes assinados).

O método falhará se MQMessage.DataLength for menor que 2 quando emitido.

DataOffset será incrementado em 2 e DataLength será decrementado em 2 se o método for bem-sucedido.

Os 2 bytes de dados da mensagem são considerados como sendo um número inteiro binário não assinado. A codificação é especificada pela propriedade MQMessage.Encoding.

Definido em: classe MQMessage

Sintaxe: *bigint & = MQMessage*.**ReadUInt2**

Método ReadUnsignedByte

Esse método lê 1 byte do buffer de Dados da Mensagem, iniciando com o byte referido por DataOffset e retorna como um valor de número inteiro inteiro (2 bytes assinados) no intervalo de 0 a 255.

O método falha se MQMessage.DataLength for inferior a 1 quando for emitido.

DataOffset é incrementado em 1 e DataLength é decrementado em 1 se o método for bem-sucedido.

O 1 caractere dos dados da mensagem é assumido para ser um número inteiro binário não assinado.

Definido em: classe MQMessage

Sintaxe: *integerv% = MQMessage*.**ReadUnsignedByte**

Método ReadUTF

Este método lê uma sequência de formato UTF da mensagem que começa com o byte referido por DataOffset e retorna como uma sequência ActiveX . A sequência na mensagem consiste em um comprimento de 2 bytes seguido pelos dados de caractere..

O método falhará se MQMessage.DataLength for menor que o comprimento da sequência quando ele for emitido.

DataOffset é incrementado pelo comprimento da sequência e DataLength é decrementado pelo comprimento da sequência se o método for bem-sucedido

Definido em:
Classe MQMessage

Sintaxe: *valor $= MQMessage*.**ReadUTF**

Método ResizeBuffer

Esse método altera a quantia de armazenamento atualmente alocada internamente para conter o buffer de Dados de mensagem. Ela fornece ao aplicativo algum controle sobre o gerenciamento de buffer automático, em que, se o aplicativo souber que irá lidar com uma mensagem grande, poderá assegurar que um buffer grande o suficiente esteja alocado. O aplicativo não precisa usar essa chamada – se não usar, o código de gerenciamento de buffer automático aumentará o tamanho do buffer para ajustar.

Se você redimensionar o buffer a ser menor que o MessageLength atual, corre o risco de perder dados. Se perder dados, o método retorna um CompletionCode igual a MQCC_WARNING e um ReasonCode igual a MQRC_DATA_TRUNCATED.

Se redimensionar o buffer para ser menor que o valor da propriedade **DataOffset**, a:

- Propriedade **DataOffset** mudará para apontar para o fim do novo buffer
- Propriedade **DataLength** será configurada para zero
- Propriedade **MessageLength** mudará para o novo tamanho do buffer

Definido em:
Classe MQMessage

Sintaxe: *MQMessage*.**ResizeBuffer**(*Comprimento &*)

Parâmetro:
Length& Long. Tamanho necessário em caracteres.

Método Write

Grava uma sequência de bytes para o buffer de mensagem a partir de uma matriz de byte na posição referida por Deslocamento de dados. Se necessário, o comprimento do buffer

(MQMessage.MQMessageLength) é estendido para acomodar o comprimento total da matriz de bytes. DataOffset é incrementado pelo número de bytes gravados se o método for bem-sucedido.

Definido em:
Classe MQMessage

Sintaxe: Chamar *MQMessage*.**Gravar**(valor)

Parâmetros:
data: uma matriz de bytes ou uma referência variante para uma matriz de bytes

Método WriteBoolean

Grava um valor booleano de 1 byte na posição atual no buffer de mensagem de um valor booliano de 2 bytes. DataOffset é incrementado em um.

Definido em:
Classe MQMessage

Sintaxe: Chamar *MQMessage*.**WriteBoolean**(valor)

Parâmetro:
value: Boolean (2-bytes). Valor a ser gravado.

Método WriteByte

Este método usa um valor inteiro de 2 bytes assinados e o grava no buffer Dados da mensagem como um número binário de 1 byte na posição referenciada por DataOffset. Ele substitui quaisquer dados já na posição no buffer, e estende o comprimento do buffer (MQMessage.MessageLength), se necessário.

DataOffset é incrementado em um se o método for bem-sucedido.

O valor especificado deve estar no intervalo de -128 a 127. Se não estiver, o método retorna com CompletionCode MQCC_FAILED e ReasonCode MQRC_WRITE_VALUE_ERROR.

Definido em: classe MQMessage

Sintaxe: Chamar *MQMessage*.**WriteByte**(*valor%*)

Parameter: *value%* Integer. Valor a ser gravado.

Método WriteDecimal2

Grava um número inteiro de 2 bytes assinado como um número decimal compactado de 2 bytes. DataOffset é incrementado por dois.

Definido em:
Classe MQMessage

Sintaxe: Chame *MQMessage*.**WriteDecimal2**(valor%)

Parâmetro:
value% Integer. Valor a ser gravado.

Método WriteDecimal4

Grava um número inteiro de 4 bytes assinado como um número decimal compactado de 4 bytes. DataOffset é incrementado por quatro.

Definido em:
Classe MQMessage

Sintaxe: Chamar *MQMessage*.**WritedDecimal4**(valor &)

Parâmetro:
valor & Long. Valor a ser gravado.

Método WriteDouble

Este método usa um valor de vírgula flutuante de 8 bytes assinados e o grava no buffer Dados da mensagem como um número de vírgula flutuante de 8 bytes começando na posição referenciada por DataOffset. Ele substitui quaisquer dados que já estejam nessas posições no buffer e estende o comprimento do buffer (MQMessage.MessageLength), se necessário.

DataOffset é incrementado em 8 se o método for bem-sucedido.

O método converte para a representação de ponto flutuante especificada pela propriedade MQMessage.Encoding. *Conversão para formato System/360 não é suportado.*

Definido em: classe MQMessage

Sintaxe: Chamar *MQMessage*.**WriteDouble**(*value#*)

Parâmetro:
value# Double. Valor a ser gravado.

Método WriteDouble4

Consulte "Método ReadDouble4" na página 1096 para obter uma descrição de quando ReadDouble4 e WriteDouble4 devem ser usados no lugar de ReadFloat e WriteFloat.

Este método obtém um valor de ponto flutuante de 8 bytes assinado e o grava no buffer de Dados de Mensagem como um número flutuante de 4 bytes começando na posição referida por DataOffset.

DataOffset será incrementado em 4 se o método for bem-sucedido.

Ele substitui quaisquer dados que já estejam nessas posições no buffer e estende o comprimento do buffer (MQMessage.MessageLength), se necessário.

O método converte para a representação de ponto flutuante especificada pela propriedade MQMessage.Encoding. *Conversão para formato System/360 não é suportado.*

Definido em: classe MQMessage

Sintaxe: Chamar *MQMessage*.**WriteDouble4 (value#)**

Parâmetro: *value#* Double. Valor a ser gravado.

Método WriteFloat

Este método usa um valor de vírgula flutuante de 4 bytes assinados e o grava no buffer Dados da mensagem como um número de vírgula flutuante de 4 bytes iniciando no caractere referido por DataOffset. Ele substitui quaisquer dados que já estejam nessas posições no buffer e estende o comprimento do buffer (MQMessage.MessageLength), se necessário.

DataOffset será incrementado em 4 se o método for bem-sucedido.

O método converte para a representação binária especificada pela propriedade MQMessage.Encoding. *Conversão para formato System/360 não é suportado.*

Definido em: classe MQMessage

Sintaxe: Chame *MQMessage*.**WriteFloat**(*valor!*)

Parameter *value!* Flutuante. Valor a ser gravado.

Método WriteInt2

Este método é idêntico ao método WriteShort.

Sintaxe: Chamar *MQMessage*.**WriteInt2**(*value%*)

Parameter *value%* Integer. Valor a ser gravado.

Método WriteInt4

Este método é idêntico ao método WriteLong.

Sintaxe: Chamar *MQMessage*.**WriteInt4**(*value &*)

Parâmetro *valor &* longo. Valor a ser gravado.

Método WriteLong

Este método usa um valor inteiro de 4 bytes assinado e grava-o no buffer de Dados da mensagem como um número binário de 4 bytes iniciando no byte referenciado por DataOffset. Ele substitui quaisquer dados que já estejam nessas posições no buffer e estende o comprimento do buffer (MQMessage.MessageLength), se necessário.

DataOffset será incrementado em 4 se o método for bem-sucedido.

O método converte para a representação binária especificada pela propriedade MQMessage.Encoding.

Definido em: classe MQMessage

Sintaxe: Chamar *MQMessage*.**WriteLong**(*valor &*)

Parâmetro *valor &* longo. Valor a ser gravado.

Método WriteNullTerminatedString

Este método executa uma WriteString normal e preenche os bytes restantes até o comprimento especificado com nulo. Se o número de bytes gravados pela sequência de gravação inicial for igual ao comprimento especificado, nenhum nulo será gravado. Se o número de bytes exceder o comprimento especificado, um erro (código de razão MQRC_WRITE_VALUE_ERROR) será configurado.

DataOffset será incrementado pelo comprimento especificado se o método for bem-sucedido.

Definido em: classe MQMessage

Sintaxe: Chamar *MQMessage*.**WriteNullTerminatedString***(value$, length &)*

Parâmetros:
value$ String. Valor a ser gravado.

length& Long. O comprimento do campo de sequência em bytes.

Método WriteShort

Este método usa um valor inteiro de 2 bytes assinado e grava-o no buffer de Dados da mensagem como um número binário de 2 bytes iniciando no byte referenciado por DataOffset. Ele substitui quaisquer dados que já estão nessas posições no buffer e estenderá o comprimento do buffer (MQMessage.MessageLength) se necessário.

DataOffset é incrementado em 2 se o método for bem-sucedido.

O método converte para a representação binária especificada pela propriedade MQMessage.Encoding.

Definido em: classe MQMessage

Sintaxe: Call *MQMessage*.**WriteShort**(*value%*)

Parameter *value%* Integer. Valor a ser gravado.

Método WriteString

Esse método usa uma sequência de ActiveX e a grava no buffer de Dados da mensagem iniciando no byte referido por DataOffset. Ele substitui quaisquer dados que já estão nessas posições no buffer e estenderá o comprimento do buffer (MQMessage.MessageLength) se necessário.

DataOffset é incrementado pelo comprimento da sequência em bytes se o método for bem-sucedido.

O método converte os caracteres na página de códigos especificada pela propriedade MQMessage.CharacterSet.

Definido em: classe MQMessage

Sintaxe: Chamar *MQMessage*.**WriteString**(*valor $*)

Parameter *value$* String. Valor a ser gravado.

Método WriteUInt2

Este método usa um valor de número inteiro de 4 bytes assinados e o grava no buffer Dados da mensagem como um número binário de 2 bytes não assinados, iniciando no byte referido por DataOffset. Ele substitui quaisquer dados que já estejam nessas posições no buffer e estende o comprimento do buffer (MQMessage.MessageLength), se necessário.

DataOffset é incrementado em 2 se o método for bem-sucedido.

O método converte para a representação binária especificada pela propriedade MQMessage.Encoding. O valor especificado deve estar no intervalo de 0 a 2**16-1. Se não for, o método retorna com CompletionCode MQCC_FAILED e ReasonCode MQRC_WRITE_VALUE_ERROR.

Definido em: classe MQMessage

Sintaxe: Chamar *MQMessage*.**WriteUInt2**(*valor &*)

Parâmetro *valor &* longo. Valor a ser gravado.

Método WriteUnsignedByte

Esse método pega um valor inteiro de 2 bytes assinado e grava no buffer de Dados da mensagem como um número binário não assinado de 1 byte iniciando no caractere referido por DataOffset. Ele substitui quaisquer dados que já estejam nessas posições no buffer e estende o comprimento do buffer (MQMessage.MessageLength), se necessário.

DataOffset é incrementado em 1 se o método for bem-sucedido.

O valor especificado deve estar no intervalo de 0 a 255. Se não for, o método retorna com CompletionCode MQCC_FAILED e ReasonCode MQRC_WRITE_VALUE_ERROR.

Definido em:
Classe MQMessage

Sintaxe: Chamar *MQMessage*.**WriteUnsignedByte**(*value%*)

Parameter *value%* Integer. Valor a ser gravado.

Método WriteUTF

Este método obtém uma sequência ActiveX e a grava no buffer de dados de mensagem na posição atual no formato UTF. Os dados gravados consistem em um comprimento de 2 bytes seguido pelos dados de caracteres. DataOffset será incrementado pelo comprimento da sequência se o método for bem-sucedido.

Definido em:
Classe MQMessage

Sintaxe: Call *MQMessage*.**WriteUTF**(value$)

Parâmetro:
value$ String. Valor a ser gravado.

Classe MQPutMessageOptions

Esta classe encapsula as várias opções que controlam a ação de colocar uma mensagem em uma Fila do WebSphere MQ ..

Restrição

A classe MQPutMessageOptions é contida pela classe MQSession.

Criação

New cria um novo objeto MQPutMessageOptions e define todas as suas propriedades para os valores iniciais.

Alternativamente, use o método AccessPutMessageOptions da classe MQSession.

Sintaxe

Dim *pmo* **As New MQPutMessageOptions** ou

Set *pmo* = **New MQPutMessageOptions**

Propriedades

- "propriedade CompletionCode" na página 1104.
- "Propriedade Options" na página 1104.
- "propriedade ReasonCode" na página 1104.
- "Propriedade ReasonName" na página 1105.
- "Propriedade RecordFields" na página 1105.
- "Propriedade ResolvedQueueManagerName" na página 1105.
- "Propriedade ResolvedQueueName" na página 1105.

Methods

- "método ClearErrorCodes" na página 1105.

propriedade CompletionCode

Somente leitura. Retorna o código de conclusão configurados pelo último método ou acesso de propriedade emitidos com relação ao objeto.

Definido em: classe MQPutMessageOptions

Tipo de dados: longo

Valores:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

Sintaxe: Para obter: *completioncode & = PutOpts***.CompletionCode**

Propriedade Options

Leitura/gravação. O campo Opções MQPMO. O valor inicial deste campo é MQPMO_NONE. Para obter mais informações, consulte Opções MQPMO.

Definido em: classe MQPutMessageOptions.

Tipo de dados: longo

Sintaxe: Para obter: *opções & = PutOpts***Opções**

Para configurar: *PutOpts***.Opções** = *opções &*

As opções MQPMO_PASS_IDENTITY_CONTEXT e MQPMO_PASS_ALL_CONTEXT não são suportadas.

propriedade ReasonCode

Somente leitura. Retorna o código de razão definido pelo último método ou acesso de propriedade emitidos com relação ao objeto.

Definido em: classe MQPutMessageOptions

Tipo de dados: longo

Valores:

- Consulte Códigos de razão de API.

Sintaxe: Obter: *Reasoncode & = PutOpts*.**ReasonCode**

Propriedade ReasonName

Somente leitura. Retorna o nome simbólico do código de razão mais recente. Por exemplo, "MQRC_QMGR_NOT_AVAILABLE".

Definido em: classe MQPutMessageOptions

Tipo de dados: sequência

Valores:

- Consulte Códigos de razão de API.

Sintaxe: Para obter: *reasonname $= PutOpts*.**ReasonName**

Propriedade RecordFields

Leitura/gravação. Sinalizadores que indicam quais campos devem ser customizados em uma base por fila ao colocar uma mensagem em uma lista de distribuição. O valor inicial é zero.

Esta propriedade corresponde ao sinalizadores PutMsgRecFields na estrutura MQPMO do MQI. No MQI, esses sinalizadores controlam quais campos (na estrutura MQPMR) estão presentes e são usados pelo MQPUT. Em um objeto MQPutMessageOptions, esses campos estão sempre presentes, e os sinalizadores, portanto, afetam apenas os campos que são usados pelo Put. Consulte o *WebSphere MQ Application Programming Reference* para obter detalhes adicionais.

Definido em:
Classe MQPutMessageOptions

Tipo de dados:
Long

Sintaxe: Para obter: *recordfields & = PutOpts*.**RecordFields**

Para configurar: *PutOpts*.**RecordFields** = *recordfields &*

Propriedade ResolvedQueueManagerName

Somente leitura. O campo MQPMO ResolvedQMgrName. Consulte ResolvedQMgrName (MQCHAR48) para obter detalhes. O valor inicial é todos os espaços em branco.

Definido em: classe MQPutMessageOptions

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *qmgr $= PutOpts*.**ResolvedQueueManagerName**

Propriedade ResolvedQueueName

Somente leitura. O campo MQPMO ResolvedQName. Veja ResolvedQName (MQCHAR48) para obter detalhes. O valor inicial é todos os espaços em branco.

Definido em: classe MQPutMessageOptions

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *qname $= PutOpts*.**ResolvedQueueNome**

método ClearErrorCodes

Reconfigura o CompletionCode para MQCC_OK e MQRC_NONE para ReasonCode para a classe MQPutMessageOptions e a classe MQSession.

Definido em: classe MQPutMessageOptions

Sintaxe: Call *PutOpts***.ClearErrorCodes ()**

Classe MQGetMessageOptions

Essa classe encapsula as várias opções que controlam a ação de obter uma mensagem de uma fila do WebSphere MQ .

Restrição

A classe MQGetMessageOptions está contido pela classe MQSession.

Criação

New cria um novo objeto MQGetMessageOptions e define todas as suas propriedades para valores iniciais.

Como alternativa, use o método AccessGetMessageOptions da classe MQSession.

Propriedades

- "propriedade CompletionCode" na página 1106
- "Propriedade MatchOptions" na página 1107
- "Propriedade Options" na página 1107
- "propriedade ReasonCode" na página 1107
- "Propriedade ReasonName" na página 1107
- "Propriedade ResolvedQueueName" na página 1107
- "Propriedade WaitInterval" na página 1107

Methods

- "método ClearErrorCodes" na página 1108

Sintaxe

Dim *gmo* **As New MQGetMessageOptions** ou

Set *gmo* = **New MQGetMessageOptions**

propriedade CompletionCode

Somente leitura. Retorna o código de conclusão configurados pelo último método ou acesso de propriedade emitidos com relação ao objeto.

Definido em: classe MQGetMessageOptions.

Tipo de dados: longo

Valores:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

Sintaxe: Para obter: *completioncode & = GetOpts***.CompletionCode** .

Propriedade MatchOptions

Leitura/gravação. Opções que controlam os critérios de seleção usados para MQGET. O valor inicial é MQMO_MATCH_MSG_ID MQMO_MATCH_CORREL_ID.

Definido em:
Classe MQGetMessageOptions

Tipo de dados:
Long

Valores:
Veja MatchOptions (MQLONG).

Sintaxe: Para obter: *matchoptions & = GetOpts*.**MatchOptions** .

Para configurar: *GetOpts*.**MatchOptions** =*matchoptions &*

Propriedade Options

Leitura/gravação. O campo Options de MQGMO. Consulte Opções para obter detalhes. valor inicial é MQGMO_NO_WAIT.

Definido em: classe MQGetMessageOptions.

Tipo de dados: longo

Sintaxe: Para obter: *opções & = GetOpts***Opções** Para configurar: *GetOpts***Opções** = *opções &*

propriedade ReasonCode

Somente leitura. Retorna o código de razão definido pelo último método ou acesso de propriedade emitidos com relação ao objeto.

Definido em: classe MQGetMessageOptions

Tipo de dados: longo

Valores:

- Consulte Códigos de razão de API.

Sintaxe: Para obter: *reasoncode & = GetOpts***.ReasonCode** .

Propriedade ReasonName

Somente leitura. Retorna o nome simbólico do código de razão mais recente. Por exemplo, "MQRC_QMGR_NOT_AVAILABLE". **Definido em:** classe MQGetMessageOptions

Tipo de dados: sequência

Valores:

- Consulte Códigos de razão de API.

Sintaxe: Para obter: *reasonname $= MQGetMessageOpções***.ReasonName**

Propriedade ResolvedQueueName

Somente leitura. O campo MQGMO ResolvedQName. Veja ResolvedQName (MQCHAR48) para obter detalhes. O valor inicial é todos os espaços em branco.

Definido em: classe MQGetMessageOptions

Tipo de dados: sequência de 48 caracteres

Sintaxe: Para obter: *qname $= GetOpts***.ResolvedQueueNome** .

Propriedade WaitInterval

Leitura/gravação. O campo MQGMO WaitInterval. O tempo máximo, em milissegundos, que o Get aguarda uma mensagem adequada chegar - se a ação de espera foi solicitada pela propriedade Options. Este campo tem um valor inicial de 0. Para obter detalhes de opções MQGMO, consulte MQGMO.

Definido em: classe MQGetMessageOptions

Tipo de dados: longo

Sintaxe: Para obter: *wait & = GetOpts*.**WaitInterval**

Para configurar: *GetOpts*.**WaitInterval** = wait &

método ClearErrorCodes

Reconfigura o CompletionCode para MQCC_OK e MQRC_NONE para ReasonCode para a classe MQGetMessageOptions e a classe MQSession.

Definido em: classe MQGetMessageOptions

Sintaxe: Chamada *GetOpts*.**ClearErrorCódigos ()**

Classe MQDistributionList

Esta classe encapsula uma coleta de filas – local, remota ou de alias para saída.

Criação

new cria um objeto MQDistributionList novo.

Como alternativa, utilize o método AddDistributionList da classe MQQueueManager

Propriedades

Método

Sintaxe

Dim *distlist.***Ums Novo MQDistributionList** ou **Configurar** *distlist* = **Novo MQDistributionList**

Propriedade AlternateUserId

Leitura/gravação. O ID do usuário alternativo utilizado para validar o acesso à lista de filas quando elas forem abertas.

Definido em:
Classe MQDistributionList

Tipo de dados:
Sequência de 12 caracteres

Sintaxe: Para obter: *altuser $= MQDistributionList*.**AlternateUserId**

Para configurar: *MQDistributionList*.**AlternateUserId** = *altuser $*

Propriedade CloseOptions

Leitura/gravação. As opções usadas para controlar o que acontece quando a lista de distribuição é fechada. O valor inicial é MQCO_NONE.

Definido em:
Classe MQDistributionList

Tipo de dados:
Long

Valores:

- MQCO_NONE
- MQCO_DELETE
- MQCO_DELETE_PURGE

Sintaxe: Para obter: *closeopt & = MQDistributionList*.**CloseOptions**

Para configurar: *MQDistributionList*.**CloseOptions** = *closeopt &*

propriedade CompletionCode

Somente leitura. O código de conclusão configurados pelo último método ou acesso de propriedade emitidos com relação ao objeto.

Definido em:
Classe MQDistributionList

Tipo de dados:
Long

Valores:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

Sintaxe: para obter: *completioncode & = MQDistributionList*.**CompletionCode**

Propriedade ConnectionReference

Leitura/gravação. O gerenciador de filas ao qual a lista de distribuição pertence.

Definido em:
Classe MQDistributionList

Tipo de dados:
MQQueueManager

Sintaxe: Para obter: *configure queuemanager = MQDistributionList*.**ConnectionReference**

Para configurar: *set MQDistributionList*. **ConnectionReference** = *queuemanager*

Propriedade FirstDistributionListItem

Somente leitura. O objeto de item da lista de distribuição primeira associado à lista de distribuição.

Definido em:
Classe MQDistributionList

Tipo de dados:
MQDistributionListItem

Valores:

Sintaxe: Para obter: *set distributionlistitem = MQDistributionList*.**FirstDistributionListItem**

Propriedade IsOpen

Somente leitura.

Definido em:
Classe MQDistributionList

Tipo de dados:
Booleana

Valores:

- TRUE (-1)
- FALSE (0)

Sintaxe: Para obter: *IsOpen = MQDistributionList*.**IsOpen**

Propriedade OpenOptions

Leitura/gravação. Opções a serem utilizadas quando a lista de distribuição é aberta.

Definido em:
Classe MQDistributionList

Tipo de dados:
Long

Valores:
Veja opções de MQPMO.

Sintaxe: Para obter: *openopt & = MQDistributionList*.**OpenOptions**

Para configurar: *MQDistributionList.***OpenOptions** = openopt &

propriedade ReasonCode

Somente leitura. O código de razão definido pelo último método ou acesso de propriedade emitidos com relação ao objeto.

Definido em:
Classe MQDistributionList

Tipo de dados:
Long

Valores:
Consulte Códigos de razão de API.

Sintaxe: Para obter: *reasoncode & = MQDistributionList*.**ReasonCode** .

Propriedade ReasonName

Somente leitura. O nome simbólico para a ReasonCode. Por exemplo "MQRC_QMGR_NOT_AVAILABLE".

Definido em:
Classe MQDistributionList

Tipo de dados:
Sequência

Valores:
Consulte Códigos de razão de API.

Sintaxe: Para obter: *reasonname $= MQDistributionList*.**ReasonName**

Método AddDistributionListItem

Cria um novo objeto de Item MQDistributionListe associa-o ao objeto da lista de distribuição O parâmetro de nome de fila é obrigatório.

A propriedade DistributionList do item da lista de distribuições é configurada para a lista de distribuição de propriedade e a propriedade FirstDistributionListItem da lista de distribuição é configurada para referenciar esse novo item da lista de distribuição.

Para o novo item da lista de distribuição, a propriedade PreviousDistributionListItem é configurada como nada e a propriedade NextDistributionListItem é configurada para fazer referência a qualquer item da lista de distribuição que foi anteriormente primeiro ou nada se não havia nenhum anteriormente (ou seja, o novo é inserido na frente daqueles que já existem).

Isso retornará um erro se a lista de distribuições estiver aberta

Definido em:
Classe MQDistributionList

Sintaxe: configure distributionlistitem = *MQDistributionList*.**AddDistributionListItem** (QName$, QMgrName$)

Parâmetros:

QName$ String. Nome da fila do WebSphere MQ .

QMgrName$ String. Nome do gerenciador de filas do WebSphere MQ .

método ClearErrorCodes

Reconfigura o CompletionCode para MQCC_OK e MQRC_NONE para ReasonCode para a classe MQDistributionList e a classe MQSession.

Definido em:
Classe MQDistributionList

Sintaxe: Chamada *MQDistributionList*.**ClearError**()

Método Close

Fecha uma lista de distribuição utilizando o valor atual de opções de fechamento.

Definido em:
Classe MQDistributionList

Sintaxe: Call *MQDistributionList*.**Close**()

Método Open

Abre cada uma das filas especificadas pelas propriedades QueueName e (onde apropriado) QueueManagerName dos itens da lista de distribuição associados ao objeto atual usando o valor atual do ID AlternateUser.

Definido em:
Classe MQDistributionList

Sintaxe: Chamar *MQDistributionList*.**Abrir**()

Método Put

Coloca uma mensagem em cada uma das filas identificado pelo itens da lista de distribuição associado com a lista de distribuição.

Definido em:

Classe MQDistributionList

Sintaxe

Chame MQDistributionList.**Put**(Mensagem, PutMsgOpções &)

Parâmetros

Message Objeto MQMessage que representa a mensagem a ser colocada.

PutMsgOptions Objeto MQPutMessageOptions que contém opções para controlar a operação de entrada. Se não for especificado, PutMessageOptions padrão são utilizados.

Este método utiliza um objeto MQMessage como parâmetro. A seguinte lista de distribuição de item propriedades podem ser alteradas como resultado deste método:

- CompletionCode
- ReasonCode
- ReasonName
- MessageId
- MessageIdHex
- CorrelationId
- CorrelationIdHex
- GroupId
- GroupIdHex
- Feedback
- AccountingToken
- AccountingTokenHex

Classe MQDistributionListItem

Esta classe encapsula a MQOR, MQRR, e MQPMR as estruturas e os associa com uma lista de distribuição de propriedade.

Criação

Use o método AddDistributionListItem da classe MQDistributionList

Propriedades

Methods

- “Propriedade AccountingToken” na página 1113.
- “Propriedade AccountingTokenHex” na página 1114.
- “propriedade CompletionCode” na página 1114.
- “Propriedade CorrelationId” na página 1114.
- “Propriedade CorrelationIdHex” na página 1114.
- “Propriedade DistributionList” na página 1115.
- “Propriedade Feedback” na página 1115.
- “Propriedade GroupId” na página 1115.
- “Propriedade GroupIdHex” na página 1115.

Propriedades:

- Propriedade AccountingToken
- Propriedade AccountingTokenHex
- propriedade CompletionCode
- Propriedade CorrelationId
- Propriedade CorrelationIdHex
- Propriedade DistributionList
- Propriedade Feedback
- Propriedade GroupId
- Propriedade GroupIdHex
- Propriedade MessageId
- propriedade MessageIdHex
- Propriedade NextDistributionListItem
- Propriedade PreviousDistributionListItem
- Propriedade QueueManagerName
- Propriedade QueueName
- propriedade ReasonCode
- Propriedade ReasonName

Métodos:

- método ClearErrorCodes

Criação:

Use o método AddDistributionListItem da classe MQDistributionList

Propriedade AccountingToken

Leitura/gravação. O AccountingToken a serem incluídos na MQPMR de uma mensagem quando colocada em uma fila. Seu valor inicial é todos nulos.

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Sequência de 32 caracteres

Sintaxe: Para obter: *accountingtoken $= MQDistributionListItem*.**AccountingToken**

Para configurar: *MQDistributionListItem*.**AccountingToken** = *accountingtoken $*

Propriedade AccountingTokenHex

Leitura/gravação. O AccountingToken a serem incluídos na MQPMR de uma mensagem quando colocada em uma fila.

Cada dois caracteres da sequência reprsentam o equivalente hexadecimal de um único caractere ASCII. Por exemplo, o par de caracteres "6" e "1" representam o caractere único "A", o par de caracteres "6" e "2" representam o caractere único "B" e assim por diante.

Deve-se fornecer 64 caracteres hexadecimais válidos.

Seu valor inicial é "0...0".

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Sequência de 64 caracteres hexadecimais reqpresenting 32 caracteres ASCII.

Sintaxe: Para obter: *accountingtokenh $= MQDistributionListItem*.**AccountingTokenHex**

Para configurar: *MQDistributionListItem*.**AccountingTokenHex** = *accountingtokenh $*

propriedade CompletionCode

Somente leitura. O código de conclusão configurados pela última abertura ou pedido put emitidos com relação ao objeto de lista de distribuição de propriedade.

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Long

Valores:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

Sintaxe: Para obter: *completioncode $= MQDistributionListItem*.**CompletionCode**

Propriedade CorrelationId

Leitura/gravação. O CorrelId a serem incluídas na MQPMR de uma mensagem quando colocada em uma fila. Seu valor inicial é todos nulos.

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Sequência de 24 caracteres

Sintaxe: Para obter: *correlid $= MQDistributionListItem*.**CorrelationId**

Para configurar: *MQDistributionListItem*.**CorrelationId** = *correlid $*

Propriedade CorrelationIdHex

Leitura/gravação. O CorrelId a serem incluídas na MQPMR de uma mensagem quando colocada em uma fila.

Cada dois caracteres da sequência reprsentam o equivalente hexadecimal de um único caractere ASCII. Por exemplo, o par de caracteres "6" e "1" representam o caractere único "A", o par de caracteres "6" e "2" representam o caractere único "B" e assim por diante.

Deve-se fornecer 48 caracteres hexadecimais válidos.

Seu valor inicial é "0..0".

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Sequência de 48 caracteres hexadecimais que representam 24 caracteres ASCII.

Sintaxe: Para obter: *correlidh $= MQDistributionListItem*.**CorrelationIdHex**

Para configurar: *MQDistributionListItem*.**CorrelationIdHex** = *correlidh $*

Propriedade DistributionList

Somente leitura. A lista de distribuição com o qual este item da lista de distribuição está associado.

Definido em:
Classe MQDistributionListItem

Tipo de dados:
MQDistributionList

Sintaxe: Para obter: *set distributionlist = MQDistributionListItem*.**DistributionList**

Propriedade Feedback

Leitura/gravação. O valor de feedback a serem incluídas na MQPMR de uma mensagem quando colocado em uma fila.

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Long

Valores:
Veja Feedback (MQLONG).

Sintaxe: Para obter: *feedback & = MQDistributionListItem*.**Feedback**

Para configurar: *MQDistributionListItem*.**Feedback** = *feedback &*

Propriedade GroupId

Leitura/gravação. O GroupId a ser incluído na MQPMR de uma mensagem quando colocar em uma fila. Seu valor inicial é todos nulos.

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Sequência de 24 caracteres

Sintaxe: Para obter: *groupid $= MQDistributionListItem*.**GroupId**

Para configurar: *MQDistributionListItem*.**GroupId** = *groupid $*

Propriedade GroupIdHex

Leitura/gravação. O GroupId a ser incluído na MQPMR de uma mensagem quando colocar em uma fila.

Cada dois caracteres da sequência reprsentam o equivalente hexadecimal de um único caractere ASCII. Por exemplo, o par de caracteres "6" e "1" representam o caractere único "A", o par de caracteres "6" e "2" representam o caractere único "B" e assim por diante.

Deve-se fornecer 48 caracteres hexadecimais válidos.

Seu valor inicial é "0..0".

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Sequência de 48 caracteres hexadecimais reqpresenting 24 caracteres ASCII.

Sintaxe: Para obter: *groupidh $= MQDistributionListItem*.**GroupIdHex**

Para configurar: *MQDistributionListItem*.**GroupIdHex** = *groupidh $*

Propriedade MessageId

Leitura/gravação. O MessageId a serem incluídas na MQPMR de uma mensagem quando colocado em uma fila. Seu valor inicial é todos nulos.

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Sequência de 24 caracteres

Sintaxe: Para obter: *messageid $= MQDistributionListItem*.**MessageId**

Para configurar: *MQDistributionListItem*.**MessageId** = *messageid $*

propriedade MessageIdHex

Leitura/gravação. O MessageId a serem incluídas na MQPMR de uma mensagem quando colocado em uma fila.

Cada dois caracteres da sequência reprsentam o equivalente hexadecimal de um único caractere ASCII. Por exemplo, o par de caracteres "6" e "1" representam o caractere único "A", o par de caracteres "6" e "2" representam o caractere único "B" e assim por diante.

Deve-se fornecer 48 caracteres hexadecimais válidos.

Seu valor inicial é "0..0".

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Sequência de 48 caracteres hexadecimais que representam 24 caracteres ASCII.

Sintaxe: Para obter: *messageidh $= MQDistributionListItem*.**MessageIdHex**

Para configurar: *MQDistributionListItem*.**MessageIdHex** = *messageidh $*

Propriedade NextDistributionListItem

Somente leitura. O objeto de item da lista de distribuição próxima associado à lista de distribuição da mesma.

Definido em:
Classe MQDistributionListItem

Tipo de dados:
MQDistributionListItem

Sintaxe: Para obter: *set distributionlistitem = MQDistributionListItem.***NextDistributionListItem**

Propriedade PreviousDistributionListItem

Somente leitura. O objeto de item da lista de distribuição anterior associado à lista de distribuição da mesma.

Definido em:
Classe MQDistributionListItem

Tipo de dados:
MQDistributionListItem

Sintaxe: Para obter: *configure distributionlistitem = MQDistributionListItem*.**PreviousDistributionListItem**

Propriedade QueueManagerName

Leitura/gravação. O nome do gerenciador de filas do WebSphere MQ .

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Sequência de 48 caracteres.

Sintaxe: Para obter: *qmname $= MQDistributionListItem*.**QueueManagerNome**

Para configurar: *MQDistributionListItem*.**QueueManagerNome** = *qmname $*

Propriedade QueueName

Leitura/gravação. O nome da fila do WebSphere MQ .

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Sequência de 48 caracteres.

Sintaxe: Para obter: *qname $= MQDistributionListItem*.**QueueName**

Para configurar: *MQDistributionListItem*.**QueueName** = *qname $*

propriedade ReasonCode

Somente leitura. O código de conclusão definido pela última abertura ou colocação emitida para o objeto da lista de distribuições de propriedade

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Long

Valores:
Consulte Códigos de razão de API.

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

Sintaxe: Para obter: *reasoncode & = MQDistributionListItem*.**ReasonCode**

Propriedade ReasonName

Somente leitura. O nome simbólico para a ReasonCode. Por exemplo "MQRC_QMGR_NOT_AVAILABLE".

Definido em:
Classe MQDistributionListItem

Tipo de dados:
Sequência

Valores:
Consulte Códigos de razão de API.

Sintaxe: Para obter: *reasonName $= MQDistributionListItem*.**ReasonName**

método ClearErrorCodes

Reconfigura o CompletionCode para MQCC_OK e MQRC_NONE para ReasonCode para a classe MQDistributionListItem e a classe MQSession.

Definido em:
Classe MQDistributionListItem

Sintaxe: Chamada *MQDistributionListItem*.**ClearError**

Resolução de Problemas

Informações sobre o recurso de rastreio fornecido, armadilhas comuns e ajuda sobre como evitá-las.

A seção a seguir explica o recurso de rastreio fornecido e detalha armadilhas comuns, com a ajuda sobre como evitá-las:

Usando rastreio

MQAX inclui um recurso de rastreio para ajudar a organização de serviço a identificar o que está acontecendo quando você tem um problema. Ele mostra os caminhos tomados quando você executa o script MQAX. A menos que você tenha um problema, execute com o rastreio configurado para evitar qualquer uso desnecessário de recursos do sistema.

Existem três variáveis de ambiente que você configura para controlar o rastreio:

- OMQ_TRACE
- OMQ_TRACE_PATH
- OMQ_TRACE_LEVEL

Observe que especificar *qualquer* valor para OMQ_TRACE alterna o recurso de rastreio. Mesmo se você configurar OMQ_TRACE como OFF, o rastreio ainda estará ativo

Para desativar o rastreio, não especifique um valor para OMQ_TRACE.

1. Clique em **Iniciar**
2. Clique em **Painel de Controle**
3. Clique duas vezes em **Sistema**
4. Clique em **Avançado**
5. Clique em **Ambiente**
6. Na seção intitulada "Variáveis de usuário para (nome do usuário)", clique em **Novo**
7. Insira o nome da variável e um valor válido nos campos apropriados e clique em **OK**
8. Clique em **OK** para fechar a janela Variáveis de ambiente
9. Clique em **OK** para fechar a janela Propriedades do sistema
10. Feche a janela Painel de Controle

Ao decidir onde deseja que os arquivos de rastreio sejam gravados, assegure-se de que você tenha autoridade suficiente para gravar no disco, não apenas para ler a partir dele.

Com o rastreio ativado, ele diminui a execução do MQAX, mas não afeta o desempenho de seus ambientes ActiveX ou WebSphere MQ . Quando você não precisar mais de um arquivo de rastreio, será possível excluí-lo.

Deve-se parar a execução do MQAX para mudar o status da variável OMQ_TRACE

Nome e diretório do arquivo de rastreio

O nome do arquivo de rastreio usa o formato OMQnnnnn.trc, em que nnnnn é o ID do processo ActiveX em execução no momento.

Tabela 157. Comandos e seus efeitos

Comando:	Efeito
SET OMQ_TRACE_PATH = drive:\directory	Configura o diretório de rastreio onde o arquivo de rastreio será gravado.
SET OMQ_TRACE_PATH =	Remove a variável de ambiente OMQ_PATH em que o diretório ativo atual (quando ActiveX é iniciado) é usado
ECHO %OMQ_TRACE_PATH%	Exibe a configuração atual do diretório de rastreio no Windows..
SET OMQ_TRACE = xxxxxxxx	Isso configura o rastreio como ON (ON) Você ativa o rastreio colocando um ou mais caracteres após o sinal '='. Por exemplo: SET OMQ_TRACE=yes SET OMQ_TRACE = no. Em ambos os exemplos, o rastreio será configurado como ON (ON). Isso é efetivo apenas para uma única janela / sessão
SET OMQ_TRACE=	Configura o rastreio OFF
ECHO %OMQ_TRACE%	Exibe o conteúdo da variável de ambiente no Windows.
SET	Exibe o conteúdo das variáveis de ambiente no Windows.
SET OMQ_TRACE_LEVEL = 9	Configura o nível de rastreio para 9 Valores maiores que 9 não produzem informações adicionais no arquivo de rastreio.

Quando o script do WebSphere MQ Automation Classes para ActiveX falhar

Se o script do WebSphere MQ Automation Classes para ActiveX falhar, haverá várias fontes de informações..

Primeiro relatório de sintoma da falha

Independentemente do recurso de rastreio, para erros inesperados e internos, um relatório de sintoma da Primeira falha pode ser produzido.

Esse relatório é localizado em um arquivo denominado OMQnnnnn.fdc, em que nnnnn é o número do processo de ActiveX que está em execução no momento. Localize esse arquivo no diretório ativo a partir do qual você iniciou o ActiveX ou no caminho especificado na variável de ambiente OMQ_PATH.

Outras fontes de informações

O WebSphere MQ fornece vários registros de erro e informações de rastreio, dependendo da plataforma envolvida Consulte o log de eventos do aplicativo do Windows NT

Códigos de Razão

Os códigos de razão a seguir podem ocorrer além daqueles documentados para o MQI do WebSphere MQ . Para outros códigos, consulte o log de eventos do aplicativo WebSphere MQ .

Tabela 158. Códigos de razão e o que eles significam

Código de razão	Explanation
MQRC_LIBRARY_LOAD_ERROR (6000)	Uma ou mais bibliotecas WebSphere MQ não puderam ser carregadas. Verifique se todas as bibliotecas do WebSphere MQ estão no caminho da procura correto no sistema que você está usando Por exemplo, certifique-se de que os diretórios que contêm as bibliotecas do WebSphere MQ estejam no PATH
MQRC_CLASS_LIBRARY_ERROR (6001)	Uma das chamadas da biblioteca de classes WebSphere MQ retornou um ReasonCode/CompletionCodeinesperado. Verifique o First Failure Symptom Report para obter detalhes. Anote o último método / propriedade e a classe que está sendo usada e informe o Suporte IBM do problema

Tabela 158. Códigos de razão e o que eles significam (continuação)

Código de razão	Explanation
MQRC_STRING_LENGTH_TOO_BIG (6002)	Foi feita uma tentativa de gravar uma sequência em formato UTF com um comprimento maior que 65.535 bytes no buffer de mensagem.
MQRC_WRITE_VALUE_ERROR (6003)	É utilizado um valor fora do intervalo; por exemplo, msg.WriteByte (240).
MQRC_PACKED_DECIMAL_ERROR (6004)	Foi feita uma tentativa de ler um número decimal compactado do buffer da mensagem, mas os dados no ponteiro de dados não estão em um formato de dados compactados válido.
MQRC_FLOAT_CONVERSION_ERROR (6005)	Foi feita uma tentativa de ler um número de vírgula flutuante único ou duplo do buffer de mensagem, mas os dados no ponteiro de dados não estão em um formato de vírgula flutuante apropriado.
MQRC_REOPEN_EXCL_INPUT_ERROR (6100)	Um objeto aberto não possui o **OpenOptionscorreto** e requer uma ou mais opções adicionais. Uma reabertura implícita é requerida, mas o fechamento foi impedido. Configure o **OpenOptions**explicitamente para cobrir todas as eventualidades para que a reabertura implícita não seja necessária. O fechamento foi evitado porque a fila está aberta para entrada exclusiva e o fechamento apresentaria uma janela de oportunidade para outros potencialmente obterem acesso à fila..
MQRC_REOPEN_INQUIRE_ERROR (6101)	Um objeto aberto não possui as OpenOptions corretas e requer uma ou mais opções adicionais. Uma reabertura implícita é requerida, mas o fechamento foi impedido. Configure o OpenOptions explicitamente para incluir MQOO_INQUIRE O encerramento foi evitado porque uma ou mais características do objeto precisam ser verificadas dinamicamente antes do fechamento e o OpenOptions ainda não inclui MQOO_INQUIRE
MQRC_REOPEN_SAVED_CONTEXT_ERR (6102)	Um objeto aberto não possui as OpenOptions corretas e requer uma ou mais opções adicionais. Uma reabertura implícita é requerida, mas o fechamento foi impedido. Configure OpenOptions explicitamente para cobrir todas as eventualidades para que a reabertura implícita não seja necessária. O fechamento foi impedido porque a fila está aberta com MQOO_SAVE_ALL_CONTEXT e um get destrutivo foi executado anteriormente. Isso fez com que informações de estado retidas fossem associadas à fila aberta e essas informações seriam destruídas pelo fechamento.
MQRC_REOPEN_TEMPORARY_Q_ERROR (6103)	Um objeto aberto não possui o **OpenOptionscorreto** e requer uma ou mais opções adicionais. Uma reabertura implícita é necessária, mas o encerramento foi evitado Configure o **OpenOptions**explicitamente para cobrir todas as eventualidades para que a reabertura implícita não seja necessária. O encerramento foi evitado porque a fila é uma fila local do tipo de definição MQQDT_TEMPORARY_DYNAMIC, que seria destruída pelo encerramento
MQRC_ATTRIBUTE_LOCKED (6104)	Foi feita uma tentativa para mudar o valor ou o atributo de um objeto enquanto o objeto está aberto. Determinados atributos, como **AlternateUserId**, não podem ser mudados enquanto um objeto é aberto.
MQRC_CURSOR_NOT_VALID (6105)	O cursor de procura de uma fila aberta foi invalidado desde que foi usado pela última vez por uma reabertura implícita. Configure OpenOptions explicitamente para cobrir todas as eventualidades para que a reabertura implícita não seja necessária.
MQRC_ENCODING_ERROR (6106)	A codificação do próximo item de mensagem precisa ser MQENC_NATIVE para leitura..
MQRC_STRUCID_ERROR (6107)	A estrutura do ID do próximo item de mensagem, que é derivada dos 4 caracteres começando no ponteiro de dados, está ausente ou é inconsistente com o tipo de variável em que o item está sendo lido.

Tabela 158. Códigos de razão e o que eles significam (continuação)

Código de razão	Explanation
MQRC_NULL_POINTER (6108)	Um ponteiro nulo é fornecido quando um ponteiro não nulo for necessário ou implícito. Isso pode ser causado usando declarações explícitas para objetos WebSphere MQ usados do VBA como parâmetros para chamadas (por exemplo, dim msg como Object is ok, dim msg como MqMessage pode causar problemas). Por exemplo, no Excel, com q definido e configurado dim msg como MqMessageq.put msg fornece reasonCode MQRC_NULL_POINTER. Ele opera corretamente do VisualBasic.
MQRC_NO_CONNECTION_REFERENCE (6109)	O objeto **MQQueue** perdeu sua conexão com **MQQueueManager**. Isso ocorrerá se o **MQQueueManager**estiver desconectado Exclua o objeto **MQQueue**.
MQRC_NO_BUFFER (6110)	Nenhum buffer está disponível. Para um objeto **MQMessage** , não é possível alocar um, denotando uma inconsistência interna no estado do objeto que não deve ocorrer
MQRC_BINARY_DATA_LENGTH_ERROR (6111)	O comprimento dos dados binários é inconsistente com o comprimento do atributo de destino. Zero é um comprimento correto para todos os atributos. 24 é um comprimento correto para um **CorrelationId** e para um **MessageId** 32 é um comprimento correto para um **AccountingToken**
MQRC_BUFFER_NOT_AUTOMATIC (6112)	Um buffer definido pelo usuário e gerenciado não pode ser redimensionado. Como buffers de mensagens são gerenciados pelo sistema, isso indica uma inconsistência interna.
MQRC_INSUFFICIENT_BUFFER (6113)	Não há espaço disponível suficiente no buffer após o ponteiro de dados para acomodar a solicitação. Isso pode ser porque o buffer não pode ser redimensionado.
MQRC_INSUFFICIENT_DATA (6114)	Há dados insuficientes após o ponteiro de dados para acomodar a solicitação de leitura. Reduza o buffer para o tamanho correto e leia os dados novamente.
MQRC_DATA_TRUNCATED (6115)	Dados foram truncados ao serem copiados de um buffer para outro. Isso pode ser porque o buffer de destino não pode ser redimensionado ou porque há um problema direcionando um ou outro buffer ou porque um buffer está sendo reduzido por uma substituição menor.
MQRC_ZERO_LENGTH (6116)	Um comprimento zero é fornecido quando um comprimento positivo for necessário ou implícito.
MQRC_NEGATIVE_LENGTH (6117)	Um comprimento negativo é fornecido quando um comprimento zero ou positivo for necessário.
MQRC_NEGATIVE_OFFSET (6118)	Um deslocamento negativo foi fornecido quando um deslocamento zero ou positivo é necessário.
MQRC_INCONSISTENT_FORMAT (6119)	O formato do próximo item de mensagem é inconsistente com o tipo de variável no qual o item está sendo lido.
MQRC_INCONSISTENT_OBJECT_STATE (6120)	Há uma inconsistência entre esse objeto, que está aberto, e o objeto MQQueueManager referido, que não está conectado.
MQRC_CONTEXT_OBJECT_NOT_VALID (6121)	A referência de contexto de Opções MQPutMessagenão referencia um objeto MQQueue válido. O objeto foi previamente destruído.
MQRC_CONTEXT_OPEN_ERROR (6122)	A referência de contexto de Opções MQPutMessagereferencia um objeto MQQueue que não pôde ser aberto para estabelecer um contexto. Isso pode ser porque o objeto MQQueue possui opções de abertura inapropriadas Inspecione o código de razão do objeto de referência para estabelecer a causa.
MQRC_STRUC_LENGTH_ERROR (6123)	O comprimento de uma estrutura de dados interna é inconsistente com seu conteúdo. Para um MQRMH, o comprimento é insuficiente para conter os campos fixos e todos os dados de deslocamento.
MQRC_NOT_CONNECTED (6124)	Um método falhou porque uma conexão necessária para um gerenciador de fila não estava disponível, e uma conexão não pode ser estabelecida implicitamente

Tabela 158. Códigos de razão e o que eles significam (continuação)	
Código de razão	**Explanation**
MQRC_NOT_OPEN (6125)	Um método falhou porque um objeto do WebSphere MQ não foi aberto e a abertura não pode ser realizada implicitamente
MQRC_DISTRIBUTION_LIST_EMPTY (6126)	Uma MQDistributionList falhou ao abrir porque não há nenhum objeto de Item MQDistributionList na lista de distribuição Ação corretiva: Inclua pelo menos um objeto de item MQDistributionListna lista de distribuição.
MQRC_INCONSISTENT_OPEN_OPTIONS (6127)	Um método falhou porque o objeto está aberto e as opções abertas são inconsistentes com a operação necessária. Ação corretiva: Abra o objeto com as opções de abertura apropriadas e tente novamente.
MQRC_WRONG_VERSION (6128)	Um método falhou porque um número de versão especificado ou encontrado é incorreto ou não é suportado.

Ferramenta de nível do código

Você pode ser solicitado pela Equipe de Serviço IBM qual nível de código você instalou.

Para descobrir isso, execute o programa utilitário 'MQAXLEV'.

No prompt de comandos, mude para o diretório que contém o MQAX200.dll ou inclua o comprimento do caminho completo e insira:

```
MQAXLev MQAX200.dll > MQAXLEV.OUT
```

em que MQAXLEV.OUT é o nome do arquivo de saída.

Se você não especificar um arquivo de saída, o detalhe será exibido na tela.

Um arquivo de saída de exemplo da ferramenta de nível de código é detalhado no exemplo a seguir:

Arquivo de saída de exemplo da ferramenta de nível de código

```
5639-B43 (C) Copyright IBM Corp. 1996, 2024. ALL RIGHTS RESERVED.
     ***** Code Level is 5.1 ***** lib/mqole/mqole.cpp, mqole, p000, p000 L981119 1.8 98/08/21
lib/mqlsx/gmqdyn0a.c, mqlsx, p000, p000 L990212  1.6 99/02/11 16:40:24
lib/mqlsx/pc/gmqdyn1p.c, mqlsx, p000, p000 L990212  1.6 99/02/11 16:44:14
lib/mqlsx/xmqcsa.c, mqole, p000, p000 L990216             1.3 99/02/15 13:24:34
lib/mqlsx/xmqfdca.c, mqlsx, p000, p000 L990212               1.3 99/02/11 16:40:35
lib/mqlsx/xmqtrca.c, mqlsx, p000, p000 L990212             1.5 99/02/11 16:12:02
lib/mqlsx/xmqutila.c, mqlsx, p000, p000 L990212           1.3 99/02/11 16:40:40
lib/mqlsx/xmqutl1a.c, mqlsx, p000, p000 L990212           1.4 99/02/11 16:40:30
lib/mqlsx/xmqcnv1a.c, mqlsx, p000, p000 L990212           1.9 99/02/11 16:40:56
lib/mqlsx/xmqmsg.c, mqole, p000, p000 L990219             1.11 99/02/18 12:12:59
```

Interface do ActiveX para a MQAI

Para obter uma visão geral breve de interfaces COM e seu uso na MQAI, consulte “Usando o Component Object Model Interface (WebSphere MQ Classes de Automação para ActiveX)” na página 1045.

A MQAI permite que os aplicativos construam e enviem comandos Programmable Command Format (PCF) sem obter e formatar diretamente os buffers de comprimento variável necessários para PCF. Para obter mais informações sobre o MQAI, consulte Introdução à Interface de Administração do WebSphere MQ (MQAI) A classe MQAI ActiveX MQBag encapsula os pacotes de dados suportados pelo MQAI de maneira que seja possível usar em qualquer linguagem que suporte a criação de objetos COM; por exemplo, Visual Basic, C + +, Java e outros clientes de script ActiveX .

A interface MQAI ActiveX é para uso com as classes MQAX que fornecem uma interface COM para a MQI. Para obter mais informações sobre as classes MQAX, consulte “Projetando aplicativos MQAX que acessam aplicativos não ActiveX” na página 1046.

A interface ActiveX fornece uma única classe chamada MQBag. Essa classe é usada para criar pacotes de dados MQAI e suas propriedades e métodos são usados para criar e trabalhar com itens de dados dentro de cada pacote. O método MQBag Execute envia os dados do pacote para um gerenciador de filas WebSphere MQ como uma mensagem PCF e coleta as respostas.

Para obter mais informações sobre a classe MQBag, suas propriedades e seus métodos, consulte “A classe MQBag” na página 1123.

A mensagem PCF é enviada para o objeto de gerenciador de filas especificado, opcionalmente usando filas de resposta e solicitação especificadas. As respostas são retornadas em um novo objeto MQBag. O conjunto completo de comandos e respostas é descrito em Definições dos Programmable Command Formats. Os comandos podem ser enviados para qualquer gerenciador de filas na rede do WebSphere MQ selecionando as filas de solicitação e de resposta apropriadas

A classe MQBag

A classe MQBag é usada para criar objetos MQBag, conforme necessário. Quando instanciada, a classe MQBag retornará uma nova referência de objeto MQBag.

Crie um objeto MQBag em Visual Basic como a seguir:

```
Dim mqbag As MQBag
Set mqbag = New MQBag
```

Propriedade MQBag

As propriedades de objetos MQBag são explicadas na lista a seguir:

- “Propriedade Item” na página 1124.
- “Propriedade Count” na página 1125.
- “Propriedade Options” na página 1125.

Métodos MQBag

Os métodos dos objetos MQBag são explicados na lista a seguir:

- “Método Add” na página 1126.
- “Método AddInquiry” na página 1127.
- “Método Clear” na página 1127.
- “Método Execute” na página 1127.
- “Método FromMessage” na página 1128.
- “Método ItemType” na página 1128.
- “Método Remove” na página 1129.
- “Método Selector” na página 1130.
- “Método ToMessage” na página 1130.
- “Método Truncate” na página 1131.

Manipulação de Erros

Se um erro for detectado durante uma operação em um objeto MQBag, incluindo os erros retornados para o pacote por um objeto subjacente MQAX ou MQAI, uma exceção de erro será levantada. A classe MQBag

suporta a interface COM ISupportErrorInfo de forma que as informações a seguir estejam disponíveis para sua rotina de manipulação de erros:

- Número do erro: composto do código de razão do WebSphere MQ para o erro detectado e um código do recurso COM O campo recurso, como padrão para COM, indica a área de responsabilidade para o erro. Para erros detectados pelo WebSphere MQ é sempre FACILITY_ITF.
- Erro de origem: identifica o tipo e a versão do objeto que detectou o erro. Para erros detectados durante as operações de MQBag, a origem do erro é sempre MQBag.MQBag1.
- Descrição do erro: a sequência que fornece o nome simbólico para o código de razão do WebSphere MQ .

Como você acessa as informações de erro depende de sua linguagem de script; por exemplo, no Visual Basic as informações são retornadas no objeto Err e o código de razão WebSphere MQ é obtido subtraindo a constante vbObjectErro de Err.Number.

ReasonCode = Err.Number - vbObjectError

Se a mensagem MQBag Execute enviar uma mensagem PCF e uma resposta for recebida, a operação será considerada bem-sucedida embora o comando enviado possa ter falhado. Nesse caso, o próprio pacote de respostas contém os códigos de razão de conclusão e de erro, conforme descrito em Definições dos Formatos de Comando Programáveis

Propriedade Item

Finalidade

A propriedade Item representa um item em um pacote. Ela é usada para configurar ou consultar sobre o valor de um item. O uso dessa propriedade corresponde às seguintes chamadas MQAI:

- "mqSetString"
- "mqSetInteger"
- "mqInquireInteger"
- "mqInquireString"
- "mqInquireBag"

no Referência de formatos de comando programáveis.

Formato

Item (Selector, ItemIndex, Value)

Parâmetros

Selector **(VARIANT) - input**
Seletor do item a ser configurado ou consultado.

Ao consultar sobre um item, MQSEL_ANY_USER_SELECTOR será o padrão. Ao configurar um item, MQIA_LIST ou MQCA_LIST serão o padrão.

Se `Selector` não for do tipo longo, resultará em MQRC_SELECTOR_TYPE_ERROR.

Esse parâmetro é opcional.

ItemIndex **(LONG) – input**
Este valor identifica a ocorrência do item do seletor especificado que deve ser configurado ou consultado. MQIND_NONE é o padrão.

Esse parâmetro é opcional.

***Value* (VARIANT) - input/output**
O valor retornado ou o valor a ser configurado. Ao consultar sobre um item, o valor de retorno pode ser de tipo longo, sequência ou MQBag. No entanto, ao configurar um item, o valor deve ser do tipo longo ou sequência, caso contrário resulta em MQRC_ITEM_VALUE_ERROR.

Chamada de linguagem Visual Basic

Ao consultar sobre um valor de um item em um pacote:

```
Value = mqbag[.Item]([Selector],
[ItemIndex])
```

Para referências MQBag:

```
Set abag = mqbag[.Item]([Selector].
[ItemIndex])
```

Para configurar o valor de um item em um pacote:

```
mqbag[.Item]([Selector],
[ItemIndex]) = Value
```

Propriedade Count

Finalidade

A propriedade Count representa o número de itens de dados dentro de um pacote. Esta propriedade corresponde à chamada MQAI, "mqCountItems," no Referência de formatos de comando programáveis.

Formato

Contagem (*Selector, Value*)

Parâmetros

***Selector* (VARIANT) - input**
Seletor dos itens de dados a serem incluídos na contagem.

MQSEL_ALL_USER_SELECTORS é o padrão.

Se `Selector` não for do tipo longo, MQRC_SELECTOR_TYPE_ERROR será retornado.

***Value* (LONG) - saída**
O número de itens no pacote incluído por *Selector*.

Chamada de linguagem Visual Basic

Para retornar o número de itens em um pacote:

```
ItemCount = mqbag.Count([Selector])
```

Propriedade Options

Finalidade

A propriedade Options define as opções para o uso de um pacote. Esta propriedade corresponde ao parâmetro `Options` da chamada MQAI, "mqCreateBag," no Referência de formatos de comando programáveis.

Formato

Opções (*`Options`*)

Parâmetros

***`Options`* (LONG) - entrada/saída**
As opções de pacote.

Nota: As opções de pacote devem ser configuradas ***antes*** de os itens de dados serem incluídos ou definidos dentro do pacote. Se as opções forem mudadas quando o pacote não estiver vazio, MQRC_OPTIONS_ERROR será o resultado. Isso se aplica mesmo se o pacote for subsequentemente removido.

Chamada de linguagem Visual Basic

Ao consultar sobre as opções de um item em um pacote:

```
Options = mqbag.Options
```

Para configurar uma opção de um item em um pacote:

```
mqbag.Options = Options
```

Métodos MQBag

Os métodos dos objetos MQBag são explicados nas seguintes páginas.

Método Add

Finalidade

O método Add inclui um item de dados em um pacote. Este método corresponde às chamadas MQAI, “mqAddInteger” e "mqAddString," no Referência de formatos de comando programáveis.

Formato

Incluir (*`Value, Selector`*)

Parâmetros

***`Value`* (VARIANT) – entrada**
Valor de número inteiro ou sequência do item de dados.

***`Selector`* (VARIANT) – entrada**
O seletor que identifica o item a ser incluído.

Dependendo do tipo de `Value`, MQIA_LIST ou MQCA_LIST é o padrão. Se o parâmetro `Selector` não for do tipo longo, MQRC_SELECTOR_TYPE_ERROR será o resultado.

Chamada de linguagem Visual Basic

Para incluir um item em um pacote:

```
mqbag.Add(Value,[Selector])
```

Método AddInquiry

Finalidade

O método AddInquiry inclui um seletor que especifica o atributo a ser retornado quando um pacote de administração é enviado para executar um comando INQUIRE. Este método corresponde à chamada MQAI, "mqAddInquiry,", no Referência de formatos de comando programáveis.

Formato

AddInquiry *(Inquiry)*

Parâmetros

Inquiry **(LONG) – entrada**
Seletor do atributo do WebSphere MQ a ser retornado pelo comando de administração INQUIRE.

Chamada de linguagem Visual Basic

Para usar o método AddInquiry:

```
mqbag.AddInquiry(Inquiry)
```

Método Clear

Finalidade

O método Limpar exclui todos os itens de dados de um pacote. Esse método corresponde à chamada MQAI, "mqClearBag,", no Referência de formatos de comando programáveis.

Formato

Remover

Chamada de linguagem Visual Basic

Para excluir todos os itens de dados de um pacote:

```
mqbag.Clear
```

Método Execute

Finalidade

O método Execute envia uma mensagem de comando de administração para o servidor de comandos e aguarda por quaisquer mensagens de resposta. Este método corresponde à chamada MQAI, "mqExecute,", no Referência de formatos de comando programáveis.

Formato

Executar *(QueueManager, Command, OptionsBag, RequestQ, ReplyQ, ReplyBag*)

Parâmetros

QueueManager **(MQQueueManager) - input**
O gerenciador de filas ao qual o aplicativo está conectado.

Command **(LONG) - input**
O comando a ser executado.

OptionsBag **(MQBag) - entrada**
O pacote que contém opções que afetam o processamento da chamada.

RequestQ **(MQQueue) - input**
A fila na qual a mensagem de comando de administração será colocada.

ReplyQ **(MQQueue) - input**
A fila na qual todas as mensagens de resposta são recebidas.

ReplyBag **(MQBag) - output**
Uma referência de pacote que contém dados de mensagens de resposta.

Chamada de linguagem Visual Basic

Para enviar uma mensagem de comando de administração e aguardar a quaisquer mensagens de resposta:

```
Set ReplyBag = mqbag.Execute(QueueManager, Command,
[OptionsBag],[RequestQ],[ReplyQ])
```

Método FromMessage

Finalidade

O método FromMessage carrega dados a partir de uma mensagem em um pacote. Este método corresponde à chamada MQAI, "mqBufferToBag," no Referência de formatos de comando programáveis.

Formato

FromMessage *(Message, OptionsBag*)

Parâmetros

Message **(MQMessage) - entrada**
A mensagem que contém os dados a serem convertidos.

OptionsBag **(MQBag) - entrada**
Opções para controlar o processamento da chamada.

Chamada de linguagem Visual Basic

Para carregar dados de uma mensagem para um pacote:

```
mqbag.FromMessage(Message,[OptionsBag])
```

Método ItemType

Finalidade

O método ItemType retorna o tipo do valor em um item especificado em um pacote. Este método corresponde à chamada MQAI, "mqInquireItemInfo," no Referência de formatos de comando programáveis.

Formato

ItemType *(Selector, ItemIndex, ItemType*)

Parâmetros

Selector **(VARIANT) - input**
Seletor identificando o item a ser consultado.

MQSEL_ANY_USER_SELECTOR é o padrão. Se o parâmetro `Selector` não for do tipo longo, MQRC_SELECTOR_TYPE_ERROR será o resultado.

ItemIndex **(LONG) – input**
Índice de itens a serem consultados.

MQIND_NONE é o padrão.

ItemType **(LONG) - output**
Tipo de dados do item especificado.

Nota: Deve-se especificar o parâmetro `Selector`, o parâmetro `ItemIndex` ou ambos. Se nenhum parâmetro estiver presente, resulta em MQRC_PARAMETER_MISSING.

Chamada de linguagem Visual Basic

Para retornar o tipo de um valor:

```
ItemType = mqbag.ItemType([Selector],
[ItemIndex])
```

Método Remove

Finalidade

O método remove exclui um item de um pacote. Este método corresponde à chamada MQAI, "mqDeleteItem," no Referência de formatos de comando programáveis.

Formato

Remover *(Selector, ItemIndex*)

Parâmetros

Selector **(VARIANT) - input**
Seletor identificando o item a ser excluído.

MQSEL_ANY_USER_SELECTOR é o padrão. Se o parâmetro `Selector` não for do tipo longo, MQRC_SELECTOR_TYPE_ERROR será o resultado.

ItemIndex **(LONG) – input**
Índice do item a ser excluído.

MQIND_NONE é o padrão.

Nota: Deve-se especificar o parâmetro `Selector`, o parâmetro `ItemIndex` ou ambos. Se nenhum parâmetro estiver presente, resulta em MQRC_PARAMETER_MISSING.

Chamada de linguagem Visual Basic

Para excluir um item de um pacote:

```
mqbag.Remove([Selector],[ItemIndex])
```

Método Selector

Finalidade

O método Selector retorna o seletor de um item especificado dentro de um pacote. Este método corresponde à chamada MQAI, "mqInquireItemInfo," no Referência de formatos de comando programáveis.

Formato

Seletor ***(Selector, ItemIndex, OutSelector)***

Parâmetros

Selector **(VARIANT) - input**
Seletor identificando o item a ser consultado.

MQSEL_ANY_USER_SELECTOR é o padrão. Se o parâmetro `Selector` não for do tipo longo, MQRC_SELECTOR_TYPE_ERROR será o resultado.

ItemIndex **(LONG) – input**
Índice do item a ser consultado.

MQIND_NONE é o padrão.

OutSelector **(VARIANT) - output**
Seletor do item especificado.

Nota: Deve-se especificar o parâmetro `Selector`, o parâmetro `ItemIndex` ou ambos. Se nenhum parâmetro estiver presente, resulta em MQRC_PARAMETER_MISSING.

Chamada de linguagem Visual Basic

Para retornar o seletor de um item:

```
OutSelector = mqbag.Selector([Selector],
[ItemIndex])
```

Método ToMessage

Finalidade

O método ToMessage retorna uma referência a um objeto MQMessage. A referência contém dados de um pacote. Este método corresponde à chamada MQAI, "mqBagToBuffer”, no Referência de formatos de comando programáveis.

Formato

ToMessage *(OptionsBag, Message)*

Parâmetros

OptionsBag **(MQBag) - entrada**
Um pacote que contém opções que controlam o processamento do método.

***Message* (MQMessage) - saída**
Uma referência do objeto MQMessage que contém dados do pacote.

Chamada de linguagem Visual Basic

Para usar o Método ToMessage:

```
Set Message = mqbag.ToMessage([OptionsBag])
```

Método Truncate

Finalidade

O método Truncar reduz o número de itens do usuário em um pacote. Este método corresponde à chamada MQAI, "mqTruncateBag," no Referência de formatos de comando programáveis.

Formato

Truncar (*ItemCount*)

Parâmetros

***ItemCount* (LONG) – entrada**
O número de itens do usuário permanece no pacote após o truncamento ocorrer.

Chamada de linguagem Visual Basic

Para reduzir o número de itens do usuário em um pacote:

```
mqbag.Truncate(ItemCount)
```

Sobre as Classes de Automação do WebSphere MQ para amostras do Iniciador do ActiveX

Este apêndice descreve as classes de automação do WebSphere MQ para amostras do Iniciador ActiveX e explica como usá-las.

WebSphere MQ para Windows fornece os seguintes programas de amostra do Visual Basic:

- MQAXTRIV.VBP
- MQAXBSRV.VBP
- MQAXDLST.VBP
- MQAXCLSS.VBP

Essas amostras são executadas no Visual Basic 4 ou no Visual Basic 5. Você os encontrará no diretório ... \tools\mqax\samples\vb.

No mesmo diretório, você também encontrará amostras para Microsoft Excel e html. São elas:

- MQAX.XLS
- MQAXTRIV.XLS
- MQAXTRIV.HTM

Nota: Se estiver usando o Visual Basic 5, **deve-se** selecionar e instalar o Visual Basic do componente grid32.ocx.

O que é demonstrado nas amostras

As amostras demonstram como usar as Classes de Automação do WebSphere MQ para ActiveX para:

- Conecta-se a um gerenciador de filas
- Acessar uma fila
- Colocar uma mensagem em uma fila
- Obter uma mensagem de uma fila

A parte central da amostra do Visual Basic é mostrada nas páginas a seguir.

"Preparando para executar as amostras" na página 1132 e

"Manipulação de erros nas amostras" na página 1133

Executando as amostras do ActiveX Starter

Antes de executar as amostras do WebSphere MQ Automation Classes para ActiveX Starter, verifique se você tem um gerenciador de filas padrão em execução e se criou as definições de fila necessárias. Para obter detalhes de criação e execução de um gerenciador de filas e criar uma fila, consulte Administrando. A amostra usa a fila SYSTEM.DEFAULT.LOCAL.QUEUE , que deve ser definido em qualquer servidor WebSphere MQ configurado normalmente

As diferentes maneiras de usar pacotes de dados são conforme o mostrado na lista a seguir:

- Conecta-se a um gerenciador de filas
- Acessar uma fila
- Colocar uma mensagem em uma fila
- Obter uma mensagem de uma fila

Para obter informações sobre as amostras do iniciador MQAX para Microsoft Basic Versão 4 ou posterior, consulte "Executando a amostra MQAXTRIV" na página 1133

Para obter informações sobre uma amostra que permite procurar propriedades e métodos de gerenciadores de filas e objetos de fila, consulte "Iniciando a amostra MQAXCLSS" na página 1134

Para obter informações sobre a amostra MQAXDLST, "A amostra MQAXDLST" na página 1135

Para obter informações sobre como executar a amostra do iniciador MQAX para Microsoft Excel 95 ou posterior, MQAXTRIV.XLS, consulte "Executando a amostra MQAXTRIV.XLS" na página 1135..

Para obter informações adicionais sobre como executar a demonstração de Banco com o MQAX.XLS, consulte "Executando a demonstração do Banco com MQAX.XLS" na página 1135

Para obter informações sobre amostras iniciais usando um navegador WWW compatível com ActiveX, consulte "Amostra do Starter usando um navegador WWW compatível com ActiveX" na página 1135

Preparando para executar as amostras

Para executar qualquer uma das amostras, é necessário um dos seguintes, dependendo de qual das amostras que você pretende executar.

- Microsoft Visual Basic Versão 4 (ou posterior)
- Microsoft Excel 95 (ou posterior)
- Um navegador da web

Também é necessário:

- Um gerenciador de filas do WebSphere MQ em execução.
- Uma fila WebSphere MQ já definida.

Manipulação de erros nas amostras

A maioria das amostras fornecidas no pacote WebSphere MQ Automation Classes para ActiveX exibe pouca ou nenhuma manipulação de erros. Para obter mais informações sobre a manipulação de erros, consulte “Manipulação de Erros” na página 1050.

Executando a amostra MQAXTRIV

1. Inicie o gerenciador de filas.
2. No Windows Explorer ou File Manager, selecione o ícone da amostra, MQAXTRIV.VBP (arquivo Visual Basic Project) e abra o arquivo.

   O programa Visual Basic é iniciado e abre o arquivo, MQAXTRIV.VBP.
3. No Visual Basic, pressione a tecla de função 5 (F5) para executar a amostra.
4. Clique em qualquer lugar na janela, "MQAX trivial tester".

Se tudo estiver funcionando corretamente, o plano de fundo da janela deverá ser mudado para verde. Se houver um problema com sua configuração, o plano de fundo da janela deverá ser mudado para vermelho e as informações sobre o erro serão exibidas.

A figura a seguir mostra a parte central da amostra do Visual Basic.

```
Option Explicit

Private Sub Form_Click()

'*****************************************************************************
'* This simple example illustrates how to put and get a WebSphere MQ message to
'* and from a WebSphere MQ message queue. The data from the message returned by the
'*get is read and compared with that from the original message.
'*****************************************************************************
Dim MQSess As MQSession                 '* session object
Dim QMgr As MQQueueManager              '* queue manager object
Dim Queue As MQQueue                    '* queue object
Dim PutMsg As MQMessage                 '* message object for put
Dim GetMsg As MQMessage                 '* message object for get
Dim PutOptions As MQPutMessageOptions   '* get message option

Dim GetOptions As MQGetMessageOptions   '* put message options
Dim PutMsgStr As String                 '* put message data string
Dim GetMsgStr As String                 '* get message data string
'*****************************************************************************
'* Handle errors
'*****************************************************************************
On Error GoTo HandleError

'*****************************************************************************
'* Initialize the current position for the form
'*****************************************************************************
CurrentX = 0
CurrentY = 0

'*****************************************************************************
'* Create the MQSession object and access the MQQueueManager and (local) MQQueue
'*****************************************************************************
Set MQSess = New MQSession
Set QMgr = MQSess.AccessQueueManager("")
Set Queue = QMgr.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE", _
                      MQOO_OUTPUT Or MQOO_INPUT_AS_Q_DEF)

'*****************************************************************************
'* Create a new MQMessage object for use with put, add some data then create an
'* MQPutMessageOptions object and put the message
'*****************************************************************************
Set PutMsg = MQSess.AccessMessage()
PutMsgStr = "12345678 " & Time
PutMsg.MessageData = PutMsgStr
Set PutOptions = MQSess.AccessPutMessageOptions()
Queue.Put PutMsg, PutOptions

'*****************************************************************************
'* Create a new MQMessage object for use with get, set the MessageId (to that of
```

```
'* the message that was put), create an MQGetMessageOptions object and get the
'* message.
'*
'* Note: Setting the MessageId ensures that the get returns the MQMessage
'* that was put earlier.
'*******************************************************************************

Set GetMsg = MQSess.AccessMessage()
GetMsg.MessageId = PutMsg.MessageId
Set GetOptions = MQSess.AccessGetMessageOptions()
Queue.Get GetMsg, GetOptions
'*******************************************************************************
'* Read the data from the message returned by the get, compare it with
'* that from the original message and output a suitable message.
'*******************************************************************************
GetMsgStr = GetMsg.MessageData
Cls
If GetMsgStr = PutMsgStr Then
    BackColor = RGB(127, 255, 127)  '* set to green for ok
    Print
    Print "Message data comparison was successful."
    Print "Message data: """ & GetMsgStr & """"

Else
    BackColor = RGB(255, 255, 127)  '* set to amber for compare error
    Print "Compare error: "
    Print "The message data returned by the get did not match the " &_
    "input data from the original message that was put."
    Print
    Print "Input message data:        """ & PutMsgStr & """"
    Print "Returned message data: """ & GetMsgStr & """"
End If

Exit Sub
'*******************************************************************************
'* Handle errors
'*******************************************************************************
HandleError:
Dim ErrMsg As String
Dim StrPos As Integer

Cls
BackColor = RGB(255, 0, 0)  '* set to red for error
Print "An error occurred as follows:"
Print ""
If MQSess.CompletionCode <> MQCC_OK Then
    ErrMsg = Err.Description
    StrPos = InStr(ErrMsg, " ")            '* search for first blank
    If StrPos > 0 Then
        Print Left(ErrMsg, StrPos)         '* print offending MQAX object name
    Else
        Print Error(Err)                   '* print complete error object
    End If
    Print ""
    Print "WebSphere MQ Completion Code = " & MQSess.CompletionCode
    Print "WebSphere MQ Reason Code = " & MQSess.ReasonCode
    Print "(" & MQSess.ReasonName & ")"
Else
    Print "Visual Basic error:  " & Err
    Print Error(Err)
End If

Exit Sub

End Sub
```

Iniciando a amostra MQAXCLSS

Esta amostra permite navegar pelas propriedades e métodos de gerenciadores de filas e objetos de fila.

1. Inicie o gerenciador de filas.
2. Abra o arquivo, MQAXCLSS.VBP, dando um clique duplo no ícone do documento no Windows Explorer ou clicando em Arquivo-Abrir no menu do arquivo no Visual Basic
3. Inicie a amostra.

4. Insira o gerenciador de filas apropriado e os nomes de filas e, em seguida, clique nos botões correspondentes.

A amostra MQAXDLST

A amostra em Visual Basic MQAXDLST demonstra o uso de uma lista de distribuição para enviar a mesma mensagem para duas filas com um put. Para executar a amostra, faça o mesmo que para a amostra MQAXCLSS.

Amostra do MQAX Starter para Microsoft Excel 95 ou mais recente

Esta seção explica como executar a amostra do iniciador MQAX para Microsoft Excel 95 ou posterior, MQAXTRIV.XLS.

Executando a amostra MQAXTRIV.XLS

1. Inicie o gerenciador de filas.
2. No Explorer ou File Manager, selecione o ícone para a amostra MQAX MQAXTRIV.XLS.
3. Clique no botão na planilha.
4. A tela será atualizada com uma mensagem de sucesso (ou de falha).

Executando a demonstração do Banco com MQAX.XLS

Siga estas etapas para executar a demonstração do Banco.

1. Inicie o gerenciador de filas.
2. Execute o arquivo de comandos MQSC do IBM WebSphere MQ, BANK.TST. Isso configura as definições de filas necessárias do IBM WebSphere MQ.

   Para descobrir como usar um arquivo de comandos MQSC, consulte Comandos de script (MQSC).
3. Execute MQAXBSRV.VBP. Este programa de amostra é o servidor simulando um aplicativo backend e precisa ser executado com o Microsoft Excel.
4. Execute MQAX.XLS. Esta amostra é a demonstração do cliente do IBM WebSphere MQ
5. Selecione um cliente na lista.
6. Clique em **Enviar**.

Após uma pausa curta (aproximadamente 3 segundos), os campos serão preenchidos com valores e um gráfico de barras será exibido.

Amostra do Starter usando um navegador WWW compatível com ActiveX

Nota: Para executar essa amostra, deve-se estar executando um navegador da web compatível com ActiveX. Microsoft Internet Explorer (mas não Netscape Navigator) é um navegador da Web compatível.

Executando a amostra HTML

Esta amostra demonstra como é possível chamar MQAX a partir de VBScript e JavaScript.

1. Inicie o gerenciador de filas.
2. Abra o arquivo, "MQAXTRIV.HTM", em seu navegador da web compatível com ActiveX.

   É possível fazer isso clicando duas vezes no ícone do arquivo no Windows Explorer ou escolher Arquivo-Abrir no menu Arquivo de seu navegador da web compatível com ActiveX .
3. Siga as instruções na tela.

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:

Relações Comerciais e Industriais da IBM
Av. Pasteur, 138-146
Botafogo
Rio, RJ 10504-1785
U.S.A.

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:

licença de propriedade intelectual
IBM World Trade Asia Corporation Licensing
IBM Japan, Ltd.
Minato-ku
Tóquio 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 websites não IBM são fornecidas apenas por conveniência e não representam de forma alguma um endosso a esses websites. 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:

Av. Pasteur, 138-146
Av. Pasteur, 138-146

Botafogo
Rio de Janeiro, RJ
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 somente 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: