MQSeries® Adapter Kernel para Multiplataformas

Iniciação Rápida

Versão 1 Release 1

G517-7169-05

MQSeries® Adapter Kernel para Multiplataformas

Iniciação Rápida

Versão 1 Release 1

G517-7169-05

Nota: Antes de utilizar estas informações e o produto suportado por elas, leia as informações contidas na seção "Avisos" na página 127.

Sexta Edição (Abril 2001)

Esta edição se aplica à versão 1, release 1, modificação de nível 1 do MQSeries Adapter Kernel para Multiplataformas (número do produto 5648-D75) e a todos os releases e modificações subseqüentes, até que seja indicado de forma diferente em novas edições.

A IBM aprecia seus comentários. Você pode enviar comentários sobre estas informações por e-mail para idrcf@hursley.ibm.com.

Índice

Figuras

Tabelas

Bem-vindo MQSeries Adapter Kernel Iniciação Rápida

Este documento descreve o MQSeries® Adapter Kernel e explica como planejá-lo, instalá-lo e utilizá-lo.

Para tornar o kernel pronto para ser utilizado, execute as seguintes etapas gerais:

1. Leia o "Capítulo 1. Sobre o Pacote MQSeries Adapter" na página 1.
2. Prepare a instalação. Consulte a seção "Preparação para a Instalação" na página 41 para obter detalhes.
3. Instale o kernel. Consulte a seção "Instalação do kernel" na página 43 para obter detalhes.
4. Verifique a instalação. Consulte a seção "Verificação da Instalação" na página 48 para obter detalhes.
5. Configure o kernel. Consulte a seção "Configuração do kernel" na página 60 para obter detalhes.
6. Se desejar, configure o software opcional para que este funcione com o kernel. Consulte a seção "Configuração do MQSeries e do MQSeries Integrator" na página 92 para obter detalhes.
7. Crie seus adaptadores utilizando o MQSeries Adapter Builder e, em seguida, teste e implemente-as. Consulte a documentação do MQSeries Adapter Builder para obter detalhes.
8. Inicie o kernel. Consulte a seção "Início do kernel" na página 93 para obter detalhes.

Para utilizar estas informações, você também precisa saber sobre os pré-requisitos e os produtos opcionais. Consulte o "Capítulo 2. Planejamento da instalação do kernel" na página 33. Consulte também a seção "Referências" na página 101.

Quem deve utilizar estas informações

Estas informações destinam-se a quem precisa planejar, instalar ou utilizar o MQSeries Adapter Kernel.

Informações relacionadas

Para obter informações adicionais, consulte o seguinte:

- o arquivo `readme.txt`. Este arquivo contém possivelmente as informações que foram disponibilizados após a conclusão deste manual. Antes da instalação, o arquivo `readme.txt` está localizado no diretório raiz do CD-ROM do produto. Depois da instalação, o arquivo `readme.txt` está localizado no diretório raiz da instalação do MQSeries Adapter Kernel.
- O *Manual de Determinação de Problemas*, número do formulário G517-7170, que descreve as ferramentas, incluindo o rastreamento, para solucionar problemas específicos com o MQSeries Adapter Kernel. O *Manual de Determinação de Problemas* está disponível no Centro de Informações do MQSeries Adapter Kernel, instalado com o produto.
- A documentação online da interface de programação de aplicativo (API) é fornecida no Centro de Informações do MQSeries Adapter Kernel. Estas informações são fornecidas apenas como um auxílio para entender as funções kernel e o auxílio para os diagnósticos. Consulte o "Capítulo 5. Utilização de APIs do MQSeries Adapter Kernel" na página 99.
- Informações sobre o MQSeries Adapter Builder, incluindo manuais e sistema de ajuda.
- O site Web da família do produto MQSeries no endereço www.ibm.com/software/ts/mqseries/.

  Seguindo os links neste site Web, você pode:

  – Obter as últimas informações sobre a família de produtos MQSeries, incluindo o Pacote do MQSeries Adapter.

  – Acessar os manuais MQSeries nos formatos HTML e PDF, incluindo possivelmente a edição mais recente deste manual.

  – Fazer download do MQSeries SupportPacs.

Convenções

A documentação do MQSeries Adapter Kernel utiliza as seguintes convenções tipográficas e de digitação.

Tabela 1. Convenções utilizadas neste manual

Convenção	Significado
Negrito	Indica os nomes de comandos. Ao mencionar as interfaces gráficas com o usuário (GUIs), indica os menus, os itens de menu, as etiquetas e os botões.
`Fonte Monoespaçada`	Indica o texto que deve ser digitado em um prompt de comandos e valores que você deve utilizar literalmente, como nomes de arquivos, caminhos e elementos de linguagens de programação como funções, classes e métodos. A fonte monoespaçada indica o texto na tela e os exemplos de códigos.
Itálico	Indica os valores das variáveis que você deve fornecer (por exemplo, você fornece o nome de um arquivo para *fileName*). O itálico também indica ênfase e títulos de manuais.
`%`	Representa o prompt do shell de comando UNIX para um comando que não exige privilégios raiz.
`#`	Representa o prompt do shell de comandos UNIX para um comando que exige privilégios raiz.
`C:\>`	Representa os prompts de comandos nos sistemas Windows®.
>	Quando utilizado para descrever um menu, mostra uma série de seleções de menu. Por exemplo, "Clicar em **Arquivo > Novo**" significa "No menu **Arquivo**, clique no comando **Novo**."
Inserção de comandos	Quando for instruído a "digitar" ou a "emitir" um comando, digite o comando e, em seguida, pressione Return. Por exemplo, a instrução "Enter the **ls** command" significa digite **ls** no prompt de comandos e, em seguida, pressione Return.
[ ]	Engloba itens opcionais nas descrições de sintaxe.
{ }	Abrange listas a partir das quais você deve escolher um item nas descrições de sintaxe.
|	Itens separados em uma lista de opções entre chaves ({ }) nas descrições de sintaxe.
...	As reticências nas descrições de sintaxe indicam que você pode repetir um item anterior uma ou mais vezes. As reticências em exemplos indicam que as informações foram omitidas do exemplo para fins de redução.

Nota: O termo Epic aparece em alguns valores e nomes no software kernel e neste manual. Com relação ao Pacote do MQSeries Adapter, este termo não tem significado próprio.

Resumo das alterações

A sexta edição (a edição atual) inclui as seguintes alterações comparadas com a quinta edição:

- Atualização da discussão sobre o fluxo em tempo de execução para concretizar em várias alterações. Consulte o "Fluxo em tempo de execução" na página 14.
- Informações da utilização do MQSeries Adapter Kernel com o WebSphere® Business Integrator. Consulte o "Utilização do MQSeries Adapter Kernel com o WebSphere Business Integrator e o WebSphere Application Server" na página 29 para obter detalhes.
- Informações do suporte do nível de idioma nacional fornecido com os diferentes tipos de adaptadores. Consulte o "Suporte do Idioma Nacional" na página 31 para obter detalhes.
- Esclarecimento das instruções da instalação. Consulte "Instalação do kernel" na página 43.
- Informações da instalação silenciosa. Consulte o "Utilização da instalação silenciosa" na página 53 para obter detalhes.
- Uma visão geral do conceito da configuração para assistir as configurações do kernel. Consulte o "Visão geral da configuração" na página 60 para obter detalhes.
- Informações dos valores do cabeçalho novo. Consulte o "Cabeçalho descritor de mensagem do MQSeries Adapter Kernel" na página 111 para obter detalhes.

A quinta edição inclui as seguintes alterações a partir da quarta edição:

- Informações sobre a utilização do kernel nas plataformas Windows® 2000, OS/400®, HP-UX e Solaris. Suporte para essas plataformas é novo no MQSeries Adapter Kernel versão 1.1. O kernel estava previamente disponível apenas no Windows NT® e AIX®.
- Atualizações de todas as instruções de instalação para que sejam condizentes com o MQSeries Adapter Kernel versão 1.1.
- Informações sobre a utilização do arquivo `aqmconfig.xml` para configurar o MQSeries Adapter Kernel. O kernel foi previamente configurado com o arquivo `aqmconfig.properties`. Consulte o "O arquivo de configuração" na página 66 para obter detalhes.
- Informações sobre os novos modos de comunicação do MQ e JMS (Java Message Service). Consulte o "Apêndice A. Modos de Comunicação" na página 103 para obter detalhes.

- Informações sobre o rastreamento que foi movido deste documento para o novo documento *Manual de Determinação de Problemas*. Consulte em *Manual de Determinação de Problemas* para obter mais informações.

Capítulo 1. Sobre o Pacote MQSeries Adapter

O IBM MQSeries Adapter Kernel faz parte de um conjunto de produtos de integração de aplicativo que juntos são chamados de Pacotes do IBM MQSeries Adapter. O Pacote do MQSeries Adapter opera com outros serviços de mensagem e mensagem MQSeries, para permitir que você reduza o risco, complexidade e custos de gerenciamento da integração ponto a ponto dos seus processos de negócio.

Na integração *ponto a ponto*, cada aplicativo individualmente se comunica com cada um dos outros aplicativos. Cada interface é diferente e existem muitas interfaces diferentes. Uma alteração em um aplicativo geralmente exige mudanças em diversas interfaces. À medida que o número de aplicativos aumenta, o custo da integração ponto-a-ponto também aumenta rapidamente. Geralmente, a integração de cada novo aplicativo exige mais trabalho do que a integração anterior.

Com o Pacote MQSeries Adapter, você pode migrar a partir da integração ponto a ponto para utilizar a integração *um para todos*. Existem várias vantagens na integração um para todos, incluindo o seguinte:

- Todos os aplicativos podem utilizar uma interface comum.
- Os dados de um *aplicativo de origem*, na forma de uma *mensagem* são *encaminhados* para um ou mais *aplicativos de destino*.
- Uma alteração em um aplicativo geralmente afeta apenas aquela interface.
- Utilizar uma interface comum, ou seja, um aplicativo neutro; por exemplo, um padrão industrial como a extensible markup language (XML); pode ser ainda mais eficiente. Mais aplicativos podem ser suportados com menos esforço.
- À medida que o número de aplicativos aumenta, a integração um para todos torna-se ainda mais eficaz em termos de custo. A inclusão de um novo aplicativo geralmente não exige mudanças significativas nas interfaces de todos os outros aplicativos.
- O trabalho de integração pode ser automatizado e pode estar baseado em gabaritos.

O Pacote do MQSeries Adapter pode ser implementado sem alterar os aplicativos ou os processos comerciais. Geralmente, todo o trabalho de integração é realizado no Pacote do MQSeries Adapter, reduzindo a necessidade de gravar o código personalizado.

No Pacote do MQSeries Adapter, a interface para ou de um aplicativo é fornecida por um *adaptador*. Todos os aplicativos precisam de um adaptador pelo menos para fornece uma interface entre o ambiente de aplicativo e o ambiente de mensagem. Cada adaptador é específico a um aplicativo e para um tipo de mensagem.

O MQSeries Adapter Kernel pode ser implementado opcionalmente, com o MQSeries Integrator para desempenhar a função de intermediário e a transformação da mensagem. O Pacote do MQSeries Adapter pode ser complementado pelos pacotes de serviço da IBM e outros.

Os exemplos de utilizações de adaptadores incluem o seguinte:

- Incluir um pedido de vendas.
- Sincronizar um registro de clientes.
- Sincronizar um registro do inventário.
- Sincronizar um item.
- Sincronizar um pedido de vendas.

Tempo de produção e tempo de execução

O Pacote do MQSeries Adapter é composto por dois componentes principais, o Adapter Builder (também chamado de builder) e o Adapter Kernel (também chamado de kernel). Esta seção descreve esses componentes, bem como os adaptadores que são criados e executados pelo Pacote do Adapter.

adaptador

O software que fornece uma interface para ou de um aplicativo. Os adaptadores são criados utilizando o MQSeries Adapter Builder. Normalmente, cada adaptador é criado para ser específico para uma mensagem que é enviado a partir ou para um aplicativo. Os adaptadores não fazem parte do Pacote do MQSeries Adapter.

Um adaptador consiste no código de origem do C ou Java™ que compila uma biblioteca compartilhada. Quando os adaptadores e o MQSeries Adapter Kernel são executados juntos, eles realizam a funcionalidade em tempo de execução do Pacote do MQSeries Adapter.

Dependendo de como foi modelado no MQSeries Adapter Builder, o adaptador pode contém uma ampla variedade de funcionalidade como fluxo de controle; fluxo de dados, navegação seqüencial, ramificação condicional, incluindo decisão e integração; digitando os dados; armazenamento de contexto de dados; transformação de elementos de dados; controle de transação; operações lógica e código personalizado.

Os adaptadores podem ser utilizados novamente.

Existem dois tipos primários de adaptadores:

- Adaptadores de origem, para aplicativos que envia os dados.
- Adaptadores de destino, para aplicativos que recebem os dados.

Enviar um tipo de mensagem de um aplicativo para um segundo aplicativo normalmente, exige um adaptador de origem e um adaptador de destino. Se o segundo aplicativo tiver que enviar um tipo de mensagem para o primeiro aplicativo, são necessários outro adaptador de origem e outro adaptador de destino. Dessa forma, para enviar um tipo de mensagem a partir do primeiro aplicativo para o segundo aplicativo e, em seguida, enviar outros tipos de mensagem do segundo aplicativo novamente para o primeiro aplicativo, quatro adaptadores são normalmente disponibilizados.

É necessário um adaptador separado para cada tipo de mensagem.

Um terceiro tipo de adaptador, o adaptador de bean da sessão de serviço do Java, é utilizado, quando o IBM WebSphere Appplication Server e os bean incorporativos são utilizados no lado de destino do kernel. A implementação do WebSphere Application Server do Sun Microsystems Enterprise JavaBeans (EJB) ativa a especificação de uso dos adaptadores de bean da sessão de serviço do Java e outros beans incorporativos. Consulte a documentação para obter mais informações do "Utilização do MQSeries Adapter Kernel com o WebSphere Business Integrator e o WebSphere Application Server" na página 29 MQSeries Adapter Builder.

MQSeries Adapter Builder
Uma interface gráfica com o usuário (GUI) permite que você crie um adaptador virtualmente para qualquer aplicativo. A interface do usuário é semelhante à interface do usuário do MQSeries Integrator. Para obter mais informações, consulte o Centro de Informações do MQSeries Adapter Builder.

MQSeries Adapter Kernel
Um conjunto de interface de programação de aplicativo (APIs - application programming interfaces), vários programas executáveis no C e Java e vários arquivos de configuração. O kernel ativa a implementação e execução de adaptadores. Além disso, para suportar diretamente os adaptadores, o kernel desempenha funções relacionadas, incluindo o roteamento simples de mensagens. Fornece também, serviços de infra-estrutura como a construção de mensagem, controle de transação, rastreamento e interface com o MQSeries ou outros softwares de mensagem.

O kernel é instalado em cada computador no qual um adaptador de origem ou um adaptador de destino é executado.

Com o Pacote do MQSeries Adapter, os processos comerciais e cada aplicativo podem permanecer isolados de especificações de middleware, detalhes de mensagens e outros aplicativos. Uma interface comum para mensagens permite a inclusão de novos aplicativos, sem alterar os aplicativos existentes ou os processos comerciais.

O MQSeries Adapter Kernel pode ser implementado em duas camadas. Uma camada é o lado de origem do tempo de execução; a outra camada é o lado de destino do tempo de execução. Uma implementação de duas camadas oferece uma operação eficiente e baixas despesas indiretas administrativas. Uma terceira camada para roteamento e entrega não é exigida para residir entre os dois lados do tempo de execução. No entanto, o MQSeries Integrator pode, opcionalmente, ser incluído para servir como intermediário, como roteamento complexo, transformação de dados e mediação de dados.

Exceto onde especificado, o restante deste documento pertence somente ao MQSeries Adapter Kernel. Para obter informações detalhadas sobre o MQSeries Adapter Builder, consulte o Centro de Informações do produto.

Sobre o kernel

Em resumo, o tempo de execução; ou seja, o kernel e os adaptadores constituídos; têm os seguintes objetivos:

1. Transferir dados de um aplicativo de origem para um aplicativo de destino.
2. Converter os dados de um aplicativo de origem para uma mensagem, geralmente no formato de um aplicativo neutro, ou seja, encaminhado pelo kernel, utilizando o MQSeries ou outro software de mensagens.
3. Encaminhar a mensagem para o aplicativo de destino.
4. Determinar como obter os dados para o aplicativo de destino.
5. Converter os dados do formato da mensagem que é roteada pelo kernel por um adaptador para o formato do aplicativo de destino.

Nesta seção, a funcionalidade do kernel é discutida em alto nível. A funcionalidade é discutida com mais detalhes no “Fluxo em tempo de execução” na página 14.

Existem dois lados do kernel:

- O *lado de origem*, que começa quando a mensagem é recebida a partir do aplicativo de origem, e encerra quando a mensagem é colocada na fila de mensagem.

- O *lado do destino*, que começa quando a mensagem é recebida a partir da fila de mensagem e encerra quando a mensagem é enviada para o destino.

Geralmente, cada lado reside em um computador diferente, mas eles podem residir no mesmo computador.

Consulte a Figura 1 na página 6. Ela descreve a seqüência a seguir.

Lado de origem do kernel

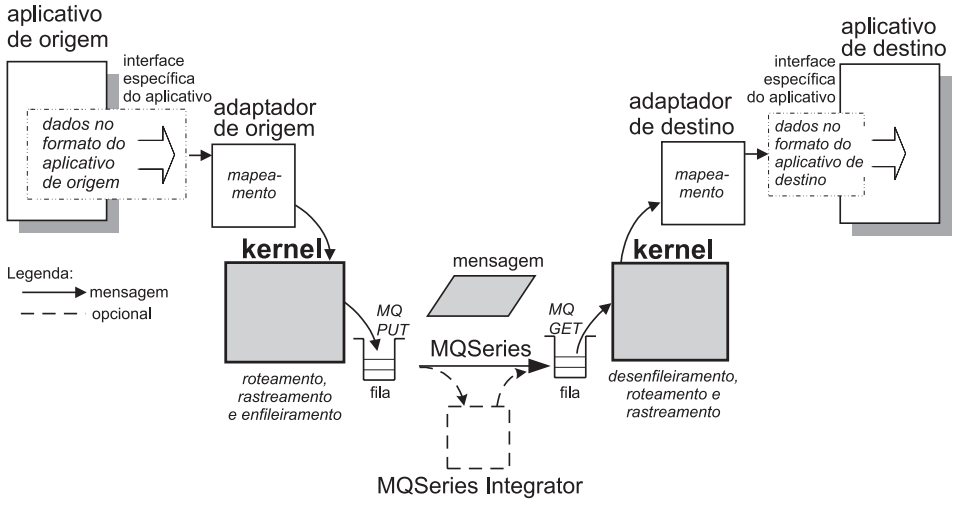
1. No lado de origem do kernel, o aplicativo de origem envia os dados no *formato do aplicativo de origem,* utilizando uma *interface específica do aplicativo,* para um adaptador de origem criado no MQSeries Adapter Builder. É necessário um adaptador de origem diferente para cada tipo de mensagem, por exemplo, para "incluir um pedido de vendas" ou para "sincronizar um registro do clientes."

   A interface específica do aplicativo deve ser desenvolvida fora do Pacote do MQSeries Adapter. O uso exato da interface específica do aplicativo depende das características do aplicativo de origem ou do aplicativo de destino. Os exemplos incluem chamadas de API e saídas do usuário, leituras e gravações de arquivos, disparos de bancos de dados e filas de mensagens.

   Observe que o adaptador de origem é executado no processo do aplicativo de origem. Qualquer daemon ou servidor que contém o adaptador de origem precisa para executar o adaptador de origem para a função.
2. O adaptador de origem realiza sua função de acordo com a forma que foi criada. Uma função comum é a transformação dos elementos de dados, ou seja, mapeamento de elementos a partir do formato do aplicativo de origem para um *formato de mensagem de integração* para os dados do corpo. Os dados do corpo e os meta-dados adicionais que representam os valores de controle são colocados em um *objeto de retenção de mensagem*.
3. Quando o adaptador de origem passa o objeto de retenção de mensagem para o kernel utilizando o *adaptador nativo*, valores de controle do objeto de retenção da mensagem (*valores de controle de mensagem*) são utilizados pelo kernel para controlar a organização do objeto de retenção de mensagem no formato de mensagem de comunicação e roteamento de mensagens de comunicação.

   Se a mensagem não contiver determinados valores de controle de mensagem, o kernel poderá utilizar o padrão ou os valores de controle de mensagem obtidos a partir do arquivo de configuração. Para obter as definições dos valores de controle de mensagem, consulte "valores de controle de mensagem" na página 16.
4. O kernel realiza suas funções, incluindo *enfileiramento de mensagens*, roteamento *simples* e, opcionalmente, *rastreamento*. Consulte as seções "Mensagem e formato de mensagem" na página 12, "Roteamento e entrega" na página 13 e "Rastreamento" na página 29.

Figura 1. Visão geral do Pacote do MQSeries Adapter.



Entrega do lado de origem para o lado de destino do kernel

5. O kernel, utilizando seu adaptador nativo, coloca a mensagem na fila de mensagem apropriada.

   Existem dois métodos de envio utilizados no lado de origem:

   - `sendMsg`, que envia a mensagem e retorna imediatamente. O método `sendMsg` também pode ser utilizado com os métodos `begin`, `commit` e `rollback` para enviar mensagens *por transações*; ou seja, as mensagens podem ser enviadas se (e apenas se) outras operações forem concluídas com êxito. Consulte a seção "Recursos transacionais" na página 28 para obter mais informações.
   - `sendRequestResponse`, que envia a mensagem e aguarda uma resposta. O método `sendRequestResponse` não pode ser emitido por transação.

     Observe que o terceiro método, `sendResponse`, é utilizado no lado de destino do kernel quando o emissor solicita uma resposta.

   O MQSeries ou outro software de mensagem transporta a mensagem. Consulte a seção "Função do MQSeries ou de outro software de mensagens" na página 8. Observe que o software de mensagem já deve ser configurado para suportar o Pacote MQSeries Adapter.

   Opcionalmente, se o MQSeries Integrator foi configurado no kernel como o destino, o MQSeries Integrator poderá desempenhar as funções de intermediário. Consulte a seção "Função do MQSeries Integrator" na página 8. Se o destino final, uma fila de mensagem, for configurada nas

regras ou no fluxo de mensagem do MQSeries Integrator, assim sendo, o MQSeries Integrator envia a mensagem para a fila de mensagem.

A mensagem chega na fila de mensagens adequada.

Lado de destino do kernel

6. No lado de destino do kernel, existem dois *modelos de entrega* possíveis para a interface entre o tempo de execução e o aplicativo de destino.
   - O modelo mais comum é *push*, no qual o kernel é responsável por iniciar e gerenciar a entrega da mensagem para o aplicativo de destino. Geralmente, o modelo push não exige alteração do aplicativo de destino para suportar o Pacote do MQSeries Adapter.
   - No modelo *pull*, o aplicativo de destino é responsável por gerenciar a recepção da mensagem. O modelo pull exige a alteração do aplicativo de destino para suportar o Pacote do MQSeries Adapter. O aplicativo de destino deve gerenciar a interface do kernel para o aplicativo de destino.

   No modelo push, observe que, no lado de destino, os processos kernel devem ser inicializados pelo usuário anteriormente para obter e entregar a mensagem. Consulte a seção "Início do kernel" na página 93.

   No modelo push, o kernel tira a mensagem da fila de mensagens. Ele realiza o rastreamento caso este esteja ativado. Ele continua a rotear a mensagem selecionando o adaptador de destino adequado. No geral, é necessário um adaptador de destino diferente para cada tipo de mensagem.
7. O kernel entrega a mensagem para o adaptador de destino adequado. O adaptador de destino realiza a funcionalidade que foi criada nele. Um função comum é o mapeamento de elementos do formato de mensagem de integração para os elementos no *formato do aplicativo de destino*.

   Os adaptadores de destino podem ser hospedados pelo daemon do adaptador MQSeries Adapter Kernel ou pelo WebSphere Application Server. Consulte "Utilização do MQSeries Adapter Kernel com o WebSphere Business Integrator e o WebSphere Application Server" na página 29, para obter uma discussão sobre esse arquivo posteriormente.
8. O adaptador de destino envia os dados para o aplicativo de destino no formato do aplicativo de destino utilizando uma interface específica de aplicativo desenvolvido fora do Pacote MQSeries Adapter.
9. Quando o adaptador de destino entregar a mensagem, a mensagem é consolidada a partir da fila de mensagem. Isso faz com que a mensagem seja removida da fila.
10. Se o adaptador de origem tiver definido um valor de controle de mensagem para solicitar uma confirmação do kernel entrega uma

confirmação da entrega da mensagem ou saída do adaptador de destino para o adaptador de origem utilizando o método `sendResponse`.

11. Em caso de erro, o kernel coloca a mensagem original na fila de erros. Se o kernel não puder colocar a mensagem original na fila de erros, não ocorrerá uma consolidação.

Função do MQSeries ou de outro software de mensagens

As mensagens de comunicação do Pacote do MQSeries Adapter são transportadas pelas filas de mensagens. As filas de mensagens são fornecidas pelo software como MQSeries ou do Java Message Service (JMS). As mensagens transportadas pelo Pacote do MQSeries Adapter utilizam os seguintes tipos de filas:

- *Filas de recepção*, na terminologia do Pacote do MQSeries Adapter. São utilizadas como filas principais de entrada para receber as mensagens. Podem existir diversas filas de recepção por aplicativo de destino.
- *Filas de erros*, na terminologia do Pacote do MQSeries Adapter. São utilizadas quando uma mensagem obtida de uma fila de recepção não pode ser processada.
- Como opção, *filas de resposta*. São utilizadas com o método `sendRequestResponse`.

O Pacote do MQSeries Adapter utiliza determinados recursos do MQSeries, como os seguintes tipos de mensagens:

- Datagramas, utilizados pelo método `sendMsg`.
- Pedido, utilizado pelo método `sendRequestResponse`.
- Resposta, utilizado pelos métodos `sendRequestResponse` e `sendResponse`.

O MQSeries pode, opcionalmente, agir como uma interface específica do aplicativo.

Consulte o “Apêndice B. Configurações válidas” na página 109 para obter uma lista de configurações válidas do MQSeries e do Pacote do MQSeries Adapter. Consulte a seção “Software” na página 34 para obter uma lista de versões suportadas do MQSeries e de outro software.

Função do MQSeries Integrator

O MQSeries Integrator pode, opcionalmente, ser implementado com o MQSeries Adapter Kernel. Pode ser utilizado para atender a diversos possíveis requisitos para função de intermediário:

- O roteamento complexo, ou seja, roteamento baseado no conteúdo do cabeçalho da mensagem ou no corpo da mensagem. O roteamento pode ser alterado dinamicamente, à medida que o conteúdo do corpo da mensagem é alterado. Consulte a seção “Roteamento e entrega” na página 13 para obter mais informações sobre os roteamentos complexo e simples.

- Transformação de dados, ou seja, alteração para um tipo de documento diferente.
- Mediação de dados, ou seja, alteração do conteúdo do corpo da mensagem. Por exemplo, se o aplicativo de origem fornece o valor `each` em um campo, mas o aplicativo de destino espera que o valor do campo seja `ea`, a mediação de dados substitui o valor fornecido pelo valor esperado.

Pode-se utilizar o MQSeries Integrator para executar a maior parte do roteamento em suas instalações; você pode diminuir também a utilização da funcionalidade de roteamento do MQSeries Adapter Kernel.

Consulte o "Apêndice B. Configurações válidas" na página 109 para obter uma lista de configurações válidas do MQSeries Integrator e do Pacote do MQSeries Adapter. Consulte a seção "Software" na página 34 para obter uma lista das versões suportadas do MQSeries Integrator e de outros softwares.

Como o kernel funciona

Os itens a seguir serão abordados nesta seção:

Componentes do tempo de execução do kernel

Quando os adaptadores são criados, o código personalizado desenvolvido e o MQSeries Adapter Kernel são executados juntos, fornecendo a funcionalidade do Pacote do MQSeries Adapter.

Os principais componentes do tempo de execução do kernel são os seguintes:

adaptador de origem
: Software criado para um aplicativo específico (geralmente utilizando o MQSeries Adapter Builder) para converter os dados desse aplicativo em um formato de mensagem de integração (dados do corpo). Geralmente, os adaptadores de origem são executados na mesma máquina que o aplicativo de origem, dentro do processo do aplicativo, ou como um processo separado. Os exemplos de dados de origem incluem arquivos, estruturas C e objetos Java. Um exemplo de um formato de mensagem de integração é o XML, geralmente seguindo um padrão industrial como OAG ou RosettaNet.

retenção de mensagens
: Um contêiner para meta-dados utilizado pelo kernel para encapsular a mensagem de integração e outros dados de controle utilizados pelo

kernel. Os exemplos de meta-dados incluem identificadores de aplicativos (identificadores lógicos) dos aplicativos de origem e de destino, a categoria de mensagem (por exemplo, OAG), o tipo de mensagem (por exemplo, "Pedido de Compra") e a mensagem de comunicação (dados do corpo) que está sendo enviada ou recebida.

adaptador nativo
O Software utilizado para enviar e receber os objetos de retenção de mensagem. Ao enviar as mensagens, o adaptador nativo fornece o roteamento de dados simples e a capacidade de suportar um ou mais mecanismos de transporte de comunicação. O roteamento de dados simples é baseado em meta-dados no objeto de retenção de mensagem como a categoria de mensagem e tipo de mensagem. As mensagens podem ser enviadas de forma assíncrona ou síncrona. Se o mecanismo de comunicação sublinhado suportar a mensagem transacional, as mensagens poderão ser enviadas pelo controle transacional da fase simples. O suporte transacional é limitado aos recursos do mecanismo de transporte utilizado. O objeto de retenção de mensagem é enfileirado no formato de mensagem de comunicação utilizado pelo mecanismo de transporte. Quando uma mensagem de comunicação é recebida, o adaptador tira a mensagem da fila e a retorna para o objeto de retenção da mensagem.

daemon do adaptador
Um processo que instancia os operadores do adaptador. Depois de iniciado, o daemon do adaptador permanece ativo. Para cada aplicativo de destino, pode haver um daemon do adaptador para cada fila de recepção do aplicativo.

operador do adaptador
Um processo que entrega cada mensagem para o adaptador de destino adequado. Cada operador gerencia um adaptador nativo. O daemon do adaptador cria e inicia os operadores.

O objetivo de ter diversos operadores é permitir a *entrega de mensagem com múltiplos encadeamentos* para os adaptadores de destino. Cada operador, com seu adaptador nativo, pode cuidar de um encadeamento. Se houver apenas um operador, a entrega de mensagens para o adaptador de destino e, ainda, para o aplicativo de destino, terá um encadeamento simples.

Além de gerenciar um adaptador nativo, o operador também executa as seguintes tarefas:

- Ele instancia o cliente de rastreamento, caso este esteja ativado.
- Ele instancia a classe de início de sessão adequada para cada aplicativo de destino.
- Ele seleciona o adaptador de destino com base no tipo de corpo e na categoria do corpo da mensagem.

- Ele envia a mensagem para o adaptador de destino selecionado.
- Se não puder executar uma consolidação, ele executará um recuo, definirá um sinalizador para os outros operadores naquele daemon do adaptador e encerrará ele próprio e o seu adaptador nativo. Isso significa que a mensagem tem um problema. Desativar todos os operadores evita o reprocessamento da mesma mensagem do problema com o mesmo resultado.
- Quando ele reconhece o sinalizador definido por outro operador para que encerre, ele encerra a si próprio e o seu adaptador nativo.

adaptador de destino
O software criado para um aplicativo específico (normalmente utilizando o MQSeries Adapter Builder) para converter os dados a partir de um formato de mensagem de integração (dados do corpo) para os tipos de dados requeridos pelo aplicativo de destino. O adaptador de destino chama as APIs necessárias no aplicativo de destino para entregar a mensagem. Os adaptadores de destino são executados na mesma máquina que o aplicativo ou o cliente do aplicativo.

adaptador bean da sessão de serviço Java
Um tipo de adaptador EJB da linguagem Java que está hospedado em um servidor EJB como WebSphere Application Server.

componente de configuração
Dados utilizados para resolver os identificadores lógicos nos objetos como nomes de filas. Os dados da configuração podem ser especificados em um arquivo ou em uma estrutura LDAP do produto WebSphere Business Integrator. Os dados controlam os seguintes aspectos da configuração do kernel:

- Enfileiramento e roteamento de mensagens
- Verificação de instalação
- Modo de comunicação
- Rastreamento

Consulte a seção “O arquivo de configuração” na página 66 para obter uma descrição completa do arquivo de configuração. Consulte a documentação WebSphere Business Integrator para obter mais informações da configuração que o produto opera com o kernel.

componente de rastreamento
O software que grava as mensagens de rastreamento. A maioria dos componentes do kernel utilizam o componente de rastreamento. Consulte a seção “Rastreamento” na página 29 para obter uma visão geral do rastreamento e o *Manual de Determinação de Problemas* para obter detalhes sobre o rastreamento.

Mensagem e formato de mensagem

No MQSeries e no Pacote do MQSeries Adapter, uma *mensagem* é uma coleção de dados enviada por um programa e destinada a outro programa. O formato da mensagem, em qualquer momento, depende do local da mensagem no fluxo de mensagem que o momento particular. O MQSeries Adapter Kernel especifica três tipos de mensagens, conforme a seguir:

- A *Mensagem de integração*—Uma mensagem consiste em dados de um aplicativo de origem convertido em outro formato como XML para enviar ao aplicativo de destino. A mensagem de integração é inserida no objeto de retenção da mensagem como os dados do corpo de mensagem. O XML é o padrão para a representação de dados. Quando o formato é XML, ele é definido por uma *Definição de Tipo de Documento* (DTD). Um DTD é um ou mais arquivos que contêm uma definição formal de um documento; neste caso, do corpo da mensagem. Assim sendo, é recomendado, o corpo de mensagem não precisa estar no corpo de mensagem para ser um formato do aplicativo neutro. O formato do corpo de mensagem pode ser proprietário ou ainda especializado; portanto, este tipo de formato não é recomendado.

  O *Business Object Documents* (BODs) pode ser utilizado pelo Pacote MQSeries Adapter para definir os corpos de mensagem nas suas mensagens de integração. Um BOD é uma representação de um processo comercial padrão que flui em uma organização ou entre organizações. Entre os exemplo estão “incluir o pedido de compra,” “mostrar disponibilidade do produto” e “incluir o pedido de venda.” Os BODs são definidos no XML pelo Grupo de Aplicativos Abertos (OAG - Open Applications Group). A utilização dos BODs é recomendável, mas não obrigatória.
- O *objeto de retenção da mensagem*—Um objeto contém a mensagem de integração e o meta-dados do cabeçalho adicional representando os valores de controle que são específicos para o MQSeries Adapter Kernel. O adaptador de origem cria o objeto de retenção da mensagem, define as informações de controle apropriado e se existir uma mensagem de integração a ser enviada, defina os dados do corpo. Os adaptadores de destino recebem os objetos de retenção de mensagem, obtém os dados de corpo e convertem os dados do corpo para os dados que são específicos para o aplicativo de destino. Os adaptadores de origem e os de destino são criados utilizando o MQSeries Adapter Builder.
- A *Mensagem de comunicação*—Quaisquer informações específicas do transporte de comunicação, mais o objeto de retenção da mensagem, convertidos no formato de mensagem específico para o transporte de comunicação sendo utilizado. Alguns transportes de comunicação suportam mais de um formato de mensagem. Normalmente, os valores dos meta-dados do cabeçalho do kernel, combinado com a mensagem de comunicação são considerados como dados do aplicativo pelo transporte de comunicação. Para obter mais informações, consulte o “Apêndice A. Modos

de Comunicação" na página 103. Os exemplos para o transporte MQSeries consiste em cabeçalho da mensagem específica do MQSeries, mais o objeto de retenção da mensagem enfileirada. Os formatos específicos do MQSeries incluem:

– O cabeçalho de mensagem do MQSeries incluído pelo MQSeries
– Se o MQSeries Integrator for utilizado, o cabeçalho da mensagem específico da versão:
  - O cabeçalho da mensagem do MQSeries Integrator versão 1, se o MQSeries Integrator versão 1.1 for utilizado
  - O cabeçalho da mensagem do MQSeries Integrator versão 2, se o MQSeries Integrator versão 2 for utilizado
– Os meta-dados do cabeçalho específicos do kernel representando os valores de controle
– A mensagem de integração (dados do corpo)

Consulte o "Apêndice C. Cabeçalhos de mensagens" na página 111 para obter uma lista de campos relevantes utilizados nos cabeçalhos da mensagem do Pacote do MQSeries Adapter e suas descrições.

Roteamento e entrega

O kernel roteia cada mensagem do adaptador de origem e entrega-a para o adaptador de destino adequado. O roteamento é executado em duas etapas:

1. O lado de origem do kernel coloca a mensagem na fila de mensagens adequada.
2. O lado de destino do kernel pega a mensagem da fila de mensagens e chama o adaptador de destino adequado.

O roteamento é determinado por diversos fatores:

- Filas de mensagens. No nível mais básico, as filas de mensagens devem estar configuradas para suportar o roteamento do Pacote do MQSeries Adapter.
- Os valores do controle de mensagem na mensagem. Incluem o identificador lógico de origem, identificador lógico de destino, identificador lógico de resposta, categoria do corpo, tipo do corpo, identificador de transação, identificador de mensagem, confirmação solicitada e registros de tempo. Consulte a seção "valores de controle de mensagem" na página 16 para obter detalhes. O identificador lógico de destino na mensagem pode substituir o arquivo de configuração do kernel. O roteamento pode ser alterado automaticamente à medida que esses valores de controle de mensagem em cada cabeçalho de mensagem mudam. No entanto, o conteúdo dos dados do corpo da mensagem (mensagem de integração) não pode determinar o roteamento.

- Os valores do controle da mensagem no arquivo de configuração do kernel. O arquivo pode especificar os identificadores lógicos de destino, nomes de fila e adaptadores de destino associados. Determine e modifique a configuração editando este arquivo. Consulte a seção "O arquivo de configuração" na página 66 para obter informações adicionais.
- Como opção, o MQSeries Integrator, o qual pode ser utilizado para interromper as mensagens, incluindo o roteamento complexo. O roteamento pode ser alterado dinamicamente, à medida que o conteúdo do corpo da mensagem é alterado. Consulte a seção "Função do MQSeries Integrator" na página 8. Em contraste, o próprio Pacote do MQSeries Adapter pode executar apenas o roteamento simples. O roteamento simples é baseado em uma combinação de valores de controle de mensagem na mensagem e nos valores de controle de mensagem associado no arquivo de configuração. Não baseia-se no conteúdo do corpo da mensagem.

O kernel pode ser solicitado a confirmar a entrega de mensagens. Isso é uma confirmação do nível do aplicativo.

Fluxo em tempo de execução

Esta seção discute o fluxo em tempo de execução em detalhes; como o kernel envia, roteia, rastreia e entrega uma mensagem em um ambiente de produção comum. Consulte a Figura 2 na página 15 para obter um diagrama do fluxo em tempo de execução.

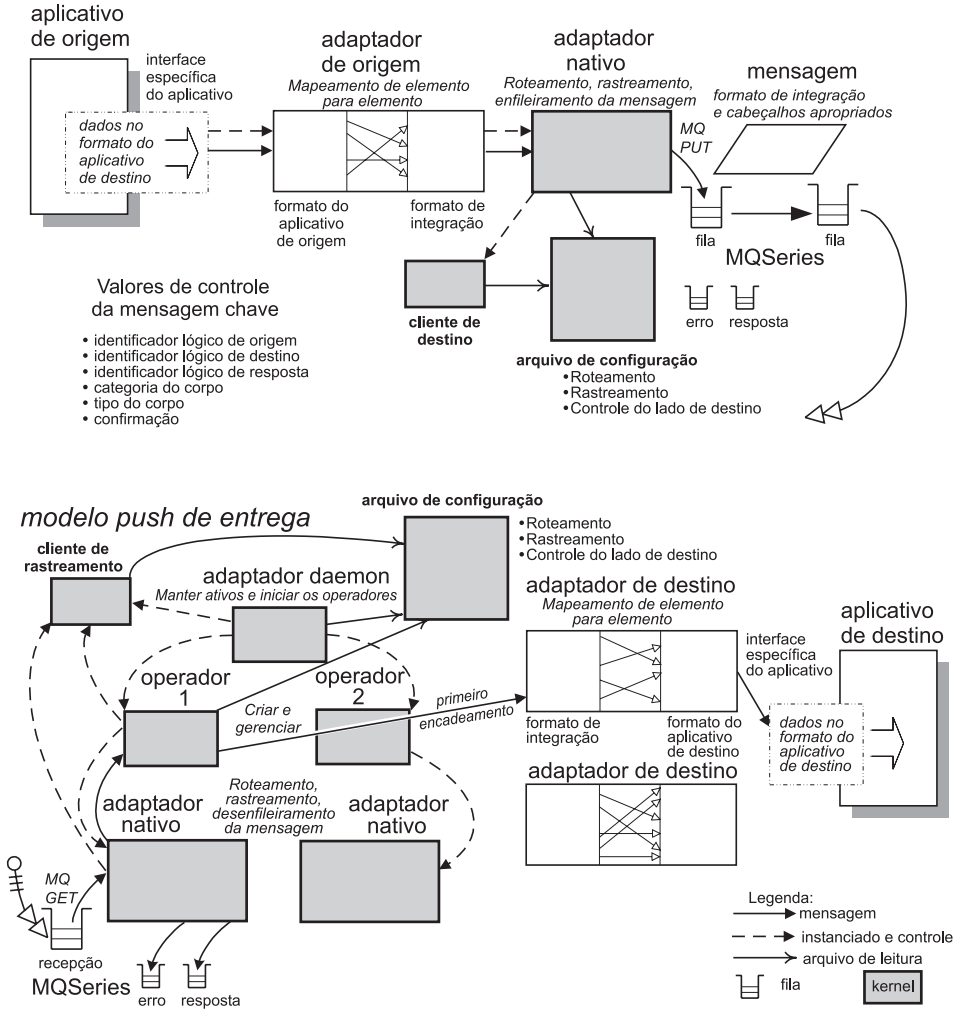


Figura 2. Enfileirar, enviar, rotear e rastrear uma mensagem — visão geral.

Lado de origem do kernel

Esta seção discute o fluxo em tempo de execução no lado de origem do kernel; ou seja, como os dados são movidos a partir do aplicativo de origem através de um adaptador de origem e no transporte de comunicação. O "Lado de destino do kernel" na página 20, discute como os dados são movidos a partir do transporte de comunicação para o destino.

1. Utilizando uma interface específica do aplicativo, o adaptador de origem adquire uma mensagem do aplicativo de origem. Geralmente, o adaptador de origem é chamado pela interface específica do aplicativo.

2. O adaptador de destino executa a funcionalidade que foi criada no MQSeries Adapter Builder. Geralmente, ele transforma os dados no formato do aplicativo de origem em um formato de integração de aplicativo neutro (para o corpo da mensagem).

   Como parte de sua funcionalidade, o adaptador de origem coloca os vários valores de controle de mensagem do cabeçalho do MQSeries Adapter Kernel; ele utiliza esses valores para enviar a mensagem. Os primeiros cinco valores de controle da mensagem determinam o enfileiramento, roteamento e a confirmação.

   valores de controle de mensagem

   identificador lógico de origem
   Identificador lógico do aplicativo de origem. Sempre exigido na mensagem.

   identificador lógico de destino
   Identificador lógico do aplicativo de destino. Se não estiver presente na mensagem, serão utilizados os valores padrão no arquivo de configuração. No arquivo de configuração, diversos identificadores lógicos de destino poderão ser utilizados no lugar de valores que não estão na mensagem.

   identificador lógico de resposta
   O identificador lógico do aplicativo para o qual as respostas devem ser enviadas se uma resposta for solicitada. Assume o padrão do identificador lógico de origem na mensagem.

   categoria do corpo
   Representa o tipo de aplicativo da mensagem; por exemplo, OAG ou RosettaNet. Sempre exigido na mensagem.

   tipo do corpo
   Representa o objetivo específico da mensagem; por exemplo, “incluir pedido de vendas” ou “sincronizar inventário”. Sempre exigido na mensagem.

   confirmação solicitada
   Determina se o aplicativo de origem solicita uma resposta. A resposta pode estar em uma das seguintes formas:

   - Dados de resposta do aplicativo de destino
   - Uma mensagem OAG Confirmar BOD

     Nota: A mensagem Confirmar BOD é predefinida pelo OAG. Sua categoria do corpo é `OAG` e seu tipo de corpo é `CONFIRM_BOD_003`. Também pode conter dados.

   Esta resposta é uma confirmação do nível do aplicativo.

Quando o kernel utiliza o método `sendRequestResponse` para enviar a mensagem, apenas a primeira resposta recebida pelo método `sendRequestResponse` será utilizada. Se a mensagem original for enviada para diversos destinos e solicitar uma resposta (o que não é recomendável), apenas a primeira resposta será retornada ao aplicativo de origem.

O valor padrão é sem confirmação; assim, nenhuma resposta é solicitada ou enviada.

3. O adaptador de origem inicializa o adaptador nativo e passa-o para o seguinte:
   - O identificador lógico do aplicativo no qual, o adaptador de origem está sendo executado.
   - O objeto de retenção de mensagem, que contêm os valores de controle da mensagem e os dados do corpo de mensagem.
4. O adaptador nativo analisa o arquivo de configuração para determinar, se o rastreamento está ativado para esse identificador lógico de origem. Se o rastreamento estiver ativado, o adaptador nativo instanciará um cliente de rastreamento.
5. O cliente de rastreamento analisa o arquivo de configuração para determinar, qual nível de rastreamento deve ser utilizado e para obter outros valores. O cliente de rastreamento utiliza o nível de rastreamento para filtrar as mensagens de saída de rastreamento. Consulte a seção "Rastreamento" na página 29 para obter uma visão geral do rastreamento e o *Manual de Determinação de Problemas* para obter informações detalhadas sobre o rastreamento.
6. O adaptador nativo procura no objeto de retenção de mensagem um identificador lógico de destino. Se estiver presente, será utilizado.
   - Se o identificador lógico de destino não estiver presente, o adaptador nativo procurará o identificador lógico de destino padrão no arquivo de configuração, com base no identificador lógico de origem, na categoria do corpo e no tipo de corpo.
   - Com base no identificador lógico de origem, o adaptador nativo desempenha uma procura com duas etapas, da categoria do corpo e valores de tipo do corpo no arquivo de configuração, na ordem a seguir:
     a. Para a categoria do corpo específico e valores de tipo do corpo.
     b. Para um valor de categoria do corpo específico e um valor de tipo do corpo padrão.
     c. Para um valor de categoria do corpo padrão e um valor de tipo do corpo específico.
     d. Para a categoria do corpo padrão e valores de tipo do corpo.

Nota: O kernel utiliza esta procura em duas etapas sempre que, procurar os valores no arquivo de configuração.

7. Para cada identificador lógico de destino determinado na etapa anterior, o adaptador nativo procura o *modo de comunicação*, com base no identificador lógico de destino, categoria do corpo e tipo do corpo. Os seguintes modos de comunicação são suportados:

   MQPP O kernel transporta as mensagens utilizando os serviços de base do MQSeries.

   MQRFH1 O kernel transporta as mensagens utilizando o MQSeries e serve como intermédio das mensagens utilizando o MQSeries Integrator versão 1.1.

   MQRFH2 O kernel transporta as mensagens utilizando o MQSeries e serve como intermediário das mensagens utilizando o MQSeries Integrator versão 2.

   MQBD O kernel transporta as mensagens utilizando os serviços de base do MQSeries, mas envia e recebe apenas os dados do corpo.

   MQ O kernel transporta as mensagens utilizando o MQSeries.

   JMS O kernel transporta mensagens utilizando o Java Message Service (JMS).

   FILE O kernel coloca as mensagens em um arquivo e pega-as de um arquivo. Este modo é fornecido apenas para fins de diagnóstico.

   Em cada modo de comunicação, a estrutura da mensagem é diferente. Consulte a seção "Mensagem e formato de mensagem" na página 12. Para obter mais informações sobre os modos de comunicação, consulte "Apêndice A. Modos de Comunicação" na página 103.

   Nota: Se o MQSeries Integrator for utilizado, o destino final para o qual, o MQSeries Integrator envia a mensagem deve utilizar o mesmo modo de comunicação como o MQSeries Integrator para receber as mensagens.

8. Com base no modo de comunicação, o adaptador nativo instancia uma subclasse dentro da mesma para manipular a mensagem. A subclasse é chamada *serviço lógico de mensagens*. Cada modo de comunicação possui uma subclasse de serviço lógico de mensagens diferente.

   O adaptador nativo passa os identificadores lógicos de destino, a categoria do corpo e o tipo de corpo para o serviço lógico de mensagens.

9. A subclasse de serviço lógico de mensagem encontra os parâmetros necessários para enviar a mensagem. Por exemplo, se o modo de

comunicação for MQPP, os parâmetros incluirão o formato e os nomes da recepção, resposta e as fila de erros. Com base nos identificadores lógicos de destino, categoria do corpo e o tipo do corpo que são passados a ele, o serviço de mensagem lógica desempenha uma procura em duas etapas no arquivo de configuração:

a. Para a categoria específica do corpo e valores de tipo do corpo.
b. Para um valor de categoria específica do corpo e um valor de tipo do corpo padrão.
c. Para um valor de categoria padrão do corpo e um valor tipo do corpo específico.
d. Para a categoria padrão do corpo e valores de tipo do corpo.

Neste ponto, o serviço lógico das mensagens tem todas as informações necessárias para rotear e enfileirar a mensagem.

10. O serviço lógico de mensagens executa as seguintes tarefas:
    - Enfileira a mensagem conforme apropriado para o modo de comunicação e formato. Cada modo de comunicação utiliza um formato padrão se o formato não foi especificado. Por exemplo, se o modo de comunicação for MQRFH2, o serviço de mensagem lógico cria cabeçalhos apropriados e estruturas de mensagem para o transporte, utilizando o MQSeries e a função de intermediário utilizando o MQSeries Integrator versão 2.
    - Envia a mensagem. Por exemplo, se o modo de comunicação for MQRFH2, ele colocará a mensagem na fila de mensagem do MQSeries apropriado.
11. Existem dois métodos que podem ser utilizados para enviar a mensagem:
    - Se o adaptador nativo utilizar o método `sendMsg` para enviar a mensagem, o adaptador nativo não irá esperar por uma resposta.
    - Se o adaptador nativo utilizar o método `sendRequestResponse` para enviar a mensagem, o serviço lógico de mensagens aguarda a resposta. O adaptador nativo, utilizando o serviço lógico de mensagens, monitora a fila de resposta para o *período de tempo limite de recepção* definido no arquivo de configuração.

      O período de tempo limite de recepção está baseado no identificador do aplicativo de origem, na categoria e no tipo de corpo.

      – Se for recebida uma confirmação, o adaptador nativo retornará a mensagem.
      – Se não for recebida uma confirmação no período de tempo limite de recepção, o adaptador nativo não retornará a mensagem.
12. O MQSeries ou outro software de mensagens transporta a mensagem de acordo com é configurado. Como opção, o MQSeries Integrator executa os serviços da função de intermediário. Consulte a seção "Função do MQSeries Integrator" na página 8.

13. Quando o adaptador de origem termina completamente com adaptador nativo, ele fecha o adaptador nativo para liberar os recursos.

Lado de destino do kernel

Esta seção discute utilizando o MQSeries Adapter Kernel independente para, receber e processar as mensagens no lado de destino, e fornecer uma descrição de alto nível da utilização do kernel com o WebSphere Application Server. Consulte a "Utilização do MQSeries Adapter Kernel com o WebSphere Business Integrator e o WebSphere Application Server" na página 29, para uma discussão utilizando o kernel com JMS, o componente do Interceptador do JMS do WebSphere Business Integrator e o WebSphere Application Server no lado de destino do kernel. Esta seção descreve o modelo pusk de entrega no qual, o kernel é responsável por inicializar e gerenciar as entregas das mensagens para o aplicativo de destino. Consulte a seção "modelos de entrega" na página 134, para obter uma pequena descrição dos modelos.

Visão Geral do operador do adaptador

Esta seção descreve a estrutura e o comportamento dos operadores do adaptador MQSeries Adapter Kernel. Uma das pretensões da arquitetura do MQSeries Adapter Kernel é que os aplicativos de destino não participem ativamente na integração do fluxo de dados com outros aplicativos; ou seja, os aplicativos normalmente, não fazem parte ativa do conjunto de mensagens para efetuar o processo. Neste caso, os dados da mensagem precisam ser ativamente empurrados para o aplicativo de destino. Os operadores de adaptadores empurram os dados de mensagem para um aplicativo ou outros serviços selecionando e chamando uma faixa do tipo de serviço da interface.

O MQSeries Adapter Kernel pode hospedar os operadores de adaptadores que, executam em um daemon independente (daemon do adaptador) ou em um servidor de aplicativo do Enterprise JavaBeans (atualmente, o IBM WebSphere Application Server Advanced Edition). As mensagens chegam no operador do adaptador por diferentes motivos, dependendo do qual tipo de ambiente de destino é utilizado. Se um daemon do adaptador independente for utilizado, ele irá hospedar um ou mais operadores dos adaptadores independentes que, utilizam o adaptador nativo para receber as mensagens. Se um servidor EJB for utilizado, o componente do Interceptador JMS receberá as mensagens e transmitirá as mesmas para um bean de mensagem do operador (mencionado muitas vezes, como um operador de adaptador de bean que guia a mensagem).

Apesar do ambiente de destino utilizado, o operador de adaptador após receber a mensagem, envia então, a mensagem adiante para o adaptador de destino apropriado. O adaptador de destino depois, executa a operação necessária para entregar a mensagem para o aplicativo de destino. Os adaptadores de destino são criados para operar com aplicativos de destino específico. O daemon do adaptador, servidor de aplicativo, operador de

adaptador independente e o bean de mensagem do operador não são especificados para qualquer aplicativo de origem e de destino fornecido.

O operador do adaptador manipula dois tipos de interfaces de adaptador de destino: Os adaptadores de comando do Enterprise Access Builder (EAB) e os beans de sessão do serviço do EJB. Cada tipo de adaptador inclui um manipulador que define o ambiente apropriado, acessa quaisquer informações de configuração adicional requerida para o adaptador e, desempenha outras tarefas de baixo nível requerida da operação dos adaptadores. O manipulador que é utilizado depende do tipo de adaptador listado no arquivo de configuração. Os dois tipos de manipuladores desempenham as tarefas adicionais a seguir:

- O manipulador EAB obtém uma classe de logon, que é utilizado para fornecer as informações de conexão para o adaptador de destino e inicializa o tempo de execução do IBM Common Connector Framework (CCF). A classe de logon é transmitida para o identificador lógico do aplicativo de destino, o qual utiliza para obter as informações de logon específico do aplicativo.
- O manipulador EJB obtém uma conexão do Java Naming and Directory Interface™ (JNDI), em seguida, obtém a interface remota do bean de sessão de serviço e outras informações requeridas para acessar o bean de sessão de serviço.

O fluxo do processo básico de um operador de adaptador no daemon do adaptador independente está como a seguir:

1. Na inicialização, o daemon do adaptador instancia um ou mais operadores de adaptador independente, de acordo com as informações fornecidas no arquivo de configuração do kernel. O identificador lógico do aplicativo, a categoria do corpo opcional e os valores do tipo do corpo são transmitidos para o daemon do adaptador. A categoria do corpo e os valores do tipo do corpo são utilizados para obter os valores de configuração adicionais.
2. Cada operador do adaptador independente desempenha as tarefas a seguir:
   a. O operador de adaptador instancia um adaptador nativo e inicia a recepção das mensagens. Cada mensagem é recebida sob o controle transacional e retornadas para o operador de adaptador como um objeto de retenção de objeto.
   b. Para cada mensagem recebida, o operador de adaptador recupera o tipo de comando do adaptador de destino para o processamento da mensagem a partir do arquivo de configuração e obtém o manipulador apropriado para o tipo de comando.

c. O manipulador obtém quaisquer informações adicionais do arquivo de configuração que precisa para instanciar a instância do adaptador de destino. Ele instancia o adaptador de destino e transmite a mensagem para o mesmo.

d. Se a mensagem é processada com êxito (ou seja, sem exceções, erros ou dados com retorno inadequado), a mensagem é consolidada a partir da fila da mensagem recebida. Se a mensagem não é processada com êxito, a mensagem é colocada em fila de erro. Se a mensagem não for processada com êxito e não puder ser colocada na fila de erro, em seguida, a mensagem é revertida e todos os operadores são encerrados.

O fluxo do processo básico de um operador de adaptador no WebSphere Application Server está como a seguir:

1. O processo do Interceptador JMS opera com o servidor EJB do WebSphere Application Server Advanced Edition recebe uma mensagem JMS. Em seguida, obtenha um bean da mensagem do operador para processar a mensagem. O identificador lógico do aplicativo, a categoria do corpo opcional e os valores do tipo do corpo fazem parte do ambiente do bean de mensagem do operador. A categoria do corpo e os valores do tipo do corpo são utilizados para obter os valores da configuração adicional.
2. Cada bean de mensagem do operador desempenha as tarefas a seguir:
   a. O bean da mensagem do operador instancia um adaptador nativo e utiliza o método `receiveMsg` do adaptador nativo, transmitindo o mesmo para a mensagem JMS. O adaptador nativo converte a mensagem JMS no objeto da mensagem e retorna o objeto de retenção da mensagem.
   b. Para cada objeto de retenção da mensagem recebida, o operador de adaptador recupera o tipo de comando do adaptador de destino para o processamento do objeto de retenção da mensagem a partir do arquivo de confirmação e obtêm um manipulador apropriado para o tipo de comando.
   c. O manipulador obtém quaisquer informações adicionais do arquivo de configuração que precisa para instanciar a instância do adaptador de destino. Ele instancia o adaptador de destino e transmite o objeto de retenção da mensagem para o mesmo.
   d. Se o objeto de retenção da mensagem é processado com êxito (ou seja, sem exceção, erros ou dados com retornos inadequado), a mensagem é consolidada a partir da fila da mensagem recebida. Se o objeto de retenção da mensagem não foi processada com êxito, ele é colocado em uma fila de erro. Se a mensagem não for processada com êxito e não puder ser colocada em uma fila de erro, em seguida, a mensagem é revertida e todos os operadores são encerrados.

O fluxo do tempo de execução do lado de destino com um daemon de adaptador e um operador de adaptador independente está como a seguir:

1. Existe um daemon do adaptador para cada fila de recepção do aplicativo de destino. O daemon do adaptador é iniciado.

   Na inicialização, é fornecido um nome que serve como um identificador de aplicativo. Normalmente, cada nome do daemon de adaptador é baseado no identificador lógico de destino—ou seja, o identificador lógico do aplicativo de destino. Por exemplo, se o daemon de adaptador é atender um aplicativo de destino cujo identificador lógico de destino é `ABC`, o nome do daemon de adaptador é `ABCdaemon`.

   Outros parâmetros que podem ser passados para o daemon do adaptador na inicialização incluem a categoria e o tipo do corpo. O adaptador nativo é utilizado posteriormente, para determinar o modo de comunicação e a fila de recepção das mensagens de recepção.

   Consulte a seção “Início do kernel” na página 93 para obter instruções sobre como iniciar o daemon do adaptador.
2. Quando é inicializado, o daemon do adaptador procura o arquivo de configuração para determinar, se o rastreio é ativado para o nome do daemon do adaptador. Se o rastreamento estiver ativado, o daemon do adaptador instanciará um cliente de rastreamento.

   Consulte a publicação *Manual de Determinação de Problemas* para obter detalhes sobre o rastreamento.
3. Quando é inicializado, o daemon do adaptador instancia a primeira operação e transmite o nome do daemon do adaptador, a categoria do corpo da mensagem e o tipo do corpo.
4. A primeira operação procura o arquivo de configuração para determinar o rastreamento que é ativado para o nome do daemon do adaptador. Se o rastreamento estiver ativo, o primeiro operador instancia um cliente de rastreamento e o cliente de rastreamento examina o arquivo de configuração para determinar o nível do rastreamento. Consulte a publicação *Manual de Determinação de Problemas* para obter uma lista dos níveis de rastreamento válidos.
5. O primeiro operador examina o arquivo de configuração, com base no identificador do aplicativo do daemon do adaptador, em busca de valores que indicam o número mínimo de operadores que devem ser instanciados e iniciados.

   A primeira operação procura também o *identificador do aplicativo de dependência*. O identificador do aplicativo de dependência é o nome do aplicativo atendido pelo operador. Ele é passado posteriormente para o adaptador nativo.
6. O daemon do adaptador coloca na fila o primeiro operador para o número mínimo de operadores.

7. O daemon do adaptador inicia o primeiro operador e, em seguida, instancia e inicia o número mínimo de operadores.

   O objetivo de ter diversos operadores é permitir a entrega de mensagens com diversos encadeamentos para os adaptadores de destino. Cada operador, com seu adaptador nativo, pode cuidar de um encadeamento. Se houver apenas um operador, a entrega de mensagens para o adaptador de destino e, ainda, para o aplicativo de destino, terá um encadeamento simples.

   Nos sistemas AIX, estão disponíveis duas políticas de planejamento para os encadeamentos: planejamento com base no processo e planejamento com base no sistema. No planejamento com base no processo (o padrão), todos os encadeamentos do usuário são mapeados para um conjunto de encadeamento do kernel do sistema operacional (SO) e executados em um conjunto de processadores virtuais. No planejamento com base no sistema, cada encadeamento do usuário é mapeado para um único encadeamento de kernel de sistema operacional e executado em um único processador virtual. Se estiver utilizando adaptadores de origem C chamados a partir dos arquivos executáveis C no AIX, você deverá utilizar o planejamento com base no sistema. Para obter mais informações sobre a definição da política de planejamento de encadeamento no AIX, consulte a Etapa .

   Observe que apenas o planejamento com base no processo é suportado nos sistemas Windows, HP-UX, Solaris e OS/400.

Os outros operadores também executam as seguintes etapas que o primeiro operador executa:

8. Cada operador instancia seu adaptador nativo associado. Existe um adaptador nativo associado a cada operador. O identificador do aplicativo de dependência, a categoria e tipo de corpo são passados para o adaptador nativo. O adaptador nativo utiliza esses três valores para determinar o modo de comunicação, utilizando o serviço lógico de mensagens, o formato e a fila de recepção para as mensagens de entrada. Este processo é semelhante ao processo utilizado para enviar mensagens.
9. O adaptador nativo obtém a mensagem de comunicação a partir da fila de recepção sob o controle de consolidação e converte-o um objeto de retenção de mensagem. Ele remove todos os cabeçalhos específicos para o transporte de comunicação exceto para o cabeçalho do kernel nativo.
10. O adaptador nativo transmite o objeto de retenção da mensagem para a operação, o qual lê a categoria do corpo, tipo do corpo e do valor da confirmação da solicitação a partir do cabeçalho do kernel nativo da mensagem.

    Com base no identificador de aplicativo de dependência, categoria do corpo, tipo do corpo, o operador executa de duas etapas no arquivo de confirmação para o tipo de comando de destino a ser chamado, na seguinte ordem:

    a. Para a categoria do corpo específico e valores de tipo do corpo.

b. Para um valor de categoria do corpo específico e um valor de tipo do corpo padrão.
c. Para um valor da categoria do corpo padrão e um valor de tipo do corpo específico.
d. Para a categoria do corpo padrão e os valores de tipo do corpo.

Com base no tipo de comando de destino, o operador determina o manipulador de tipo de adaptador de destino apropriado, uma classe Java processa o tipo de adaptador particular. Ele instancia um determinado adaptador de destino.

11. Existem dois tipos de manipuladores de tipo adaptador: Manipuladores de adaptador de destino do comando EAB e manipuladores de adaptador de destino bean da sessão do serviço EJB. Diferentes tipos de manipuladores de adaptador operam da seguinte forma:

    Nota: O manipulador de adaptador de destino do bean da sessão do serviço EJB é suportado apenas com o WebSphere Business Integrator executado com o WebSphere Application Server na plataforma do Windows NT.

    - Se um manipulador de destino do comando EAB é chamado, é iniciado o ambiente do Common Connector Framework (CCF), define uma classe do logon com um nome obtido a partir do arquivo de configuração e chama o adaptador de destino EAB com o nome obtido a partir do arquivo de configuração.
    - Um adaptador de destino do bean da sessão do serviço EJB deve interagir com o WebSphere Business Integrator e o WebSphere Application Server para obter as informações de configuração apropriadas e chamar o bean da sessão do serviço EJB. Consulte "Utilização do MQSeries Adapter Kernel com o WebSphere Business Integrator e o WebSphere Application Server" na página 29 para uma discussão da utilização do kernel com JMS, o componente do Interceptador do TMS do WebSphere Business Integrator e o WebSphere Application Server no lado de destino do kernel.

12. Cada tipo de adaptador possui uma interface diferente e as classes de suporte requerida, como a seguir:
    - Um comando de adaptador de destino EAB possui três métodos para chamar; esses métodos execute-os na seguinte ordem:
      a. O método *definir entrada de mensagem*, que define a mensagem a ser processada no adaptador de destino.
      b. O método *executar*, que processa a mensagem que foi colocado no adaptador de destino utilizando o método definir entrada de mensagem e, em seguida, aguarda.

1) O adaptador de destino executa a funcionalidade que foi criada nele utilizando o MQSeries Adapter Builder. Geralmente, transforma os dados a partir da mensagem da integração no formato do aplicativo de destino. Ela faz o mapeamento de elemento para elemento.
2) O adaptador de destino, utilizando uma interface específica do aplicativo, envia a mensagem para o aplicativo de destino.
3) Dependendo do uso do aplicativo de destino, o aplicativo de destino envia ou não envia uma resposta de volta para o adaptador de destino.

c. O método *obter a saída da mensagem*, que obtém a resposta do adaptador de destino. A resposta pode indicar simplesmente que o aplicativo de destino recebeu a mensagem; ela também pode conter dados.

- Um bean da sessão do serviço EJB chama um método, o qual, requer um objeto `TerminalDataContainer`. Os dados retornados pelo método são considerados os dados de resposta e devem ser um tipo de objeto `TerminalDataContainer`.

13. Se o comando do adaptador de destino não lançar uma exceção, ou se não tiver uma resposta Confirmar BOD (que pode indicar um erro), o operador consolida a mensagem recebida da fila de recepção, utilizando o adaptador nativo.
14. Se foi solicitada uma confirmação, o operador chama o método `sendResponse` no adaptador nativo.
    - Se o adaptador de destino criou uma resposta, é colocado o identificador lógico de resposta da mensagem original no campo do identificador lógico de destino da mensagem de resposta.
    - Se o adaptador de destino não criar uma resposta, então o operador cria uma mensagem de resposta Confirmar BOD contendo o status de conclusão.
      – Se não houver erros, o status de conclusão é bem-sucedido.
      – Se houver erros, o status de conclusão é definido como uma condição de erro.
15. A resposta é enviada.
    a. O operador envia a mensagem de resposta, se uma foi criada para o adaptador nativo.
    b. O adaptador nativo coloca a mensagem de resposta na fila de resposta.
    c. O adaptador nativo envia a mensagem de resposta, dependendo da mensagem original recebida:
       - Se foi uma mensagem de pedido do MQSeries, o adaptador nativo obterá as informações da fila para a resposta da mensagem de

pedido do MQSeries. Estas informações da fila substituem o identificador lógico de destino na mensagem.

- Se não foi uma mensagem de pedido do MQSeries, então o adaptador nativo utiliza o método `sendMsg` para enviar a resposta.

16. No caso de mensagem de exceção ou de resposta Confirmar BOD com um status de erro, o operador inicia uma mensagem de exceção em um arquivo de exceção chamado `EpicSystemExceptionFile`*nnnnnnnn*`.log` que reside em um mesmo diretório como daemon do adaptador, em que *nnnnnnnn* é o número do arquivo de log. Além disso, se as classes do WebSphere Business Integrator são instaladas, essas classes enviam uma exceção para o componente do WebSphere Business Integrator Solution Management. Consulte a seção “Mensagens de exceção” na página 97.
17. No caso de uma mensagem de exceção ou de resposta Confirmar BOD com um status de erro, o operador direciona o adaptador nativo para colocar a mensagem original na fila de erros. O nome da fila de erros é obtido no arquivo de configuração com base no identificador lógico de dependência, categoria do corpo e tipo do corpo da mensagem original.

    Com base no identificador de aplicativo de dependência, tipo do corpo, o operador executa uma procura em duas etapas no arquivo de configuração na seguinte ordem:

    a. Para a categoria específica do corpo e valores de tipo do corpo.
    b. Para um valor de categoria específica do corpo e um valor de tipo do corpo padrão.
    c. Para um valor de categoria padrão do corpo e um valor tipo do corpo específico.
    d. Para a categoria padrão do corpo e valores de tipo do corpo.

    - Se o adaptador nativo puder colocar a mensagem de erro na fila de erros, o adaptador nativo será direcionado para consolidar a mensagem da fila de recepção.
    - Se o adaptador nativo não puder colocar a mensagem de erro na fila de erros, ocorrerá o seguinte:

        a. O operador direcionará o adaptador nativo para recuo, ou seja, para não consolidar.
        b. O operador define um sinalizador que direciona todos os operadores nesse daemon do adaptador, para que sejam encerrados. Isso significa que a mensagem tem um problema. Encerrar todos os operadores evita o reprocessamento da mesma mensagem do problema com o mesmo resultado.
        c. Se ocorrer um erro por falta de memória, a exceção será tratada da mesma forma que todas as outras exceções para as quais, o operador define que o sinalizador pare quando, tiver concluído o processamento da mensagem atual. Isso disponibiliza mais memória para outros operadores.

18. Quando o adaptador nativo notifica o operador de que o trabalho foi concluído, o operador verifica dois sinalizadores:
    - Se este operador deve parar. Isso pode ser causado por uma condição sem memória do Java.
    - Se todos os operadores devem parar, pelos motivos descritos na etapa anterior.
19. Se o sinalizador for definido, o operador pára. Se nenhum sinalizador for definido, o operador processa a próxima mensagem. O operador solicita que o adaptador nativo receba uma mensagem.
20. Se uma mensagem for colocada na fila de resposta ou se uma mensagem de erro for colocada na fila de erros, ocorrerá o seguinte:
    a. O MQSeries ou outro software de mensagem a entregará de volta para o lado de origem do kernel.
    b. Se o adaptador de origem chamou seu método `sendRequestResponse` do adaptador nativo, o kernel recupera a mensagem da fila de resposta e devolve-a para o adaptador de origem. Se o adaptador de origem chamou o método `sendMsg`, o kernel coloca a mensagem na fila de recepção do aplicativo de origem.

Recursos transacionais

Uma *transação* é um conjunto de operações que deve ser executada como uma unidade de trabalho indivisível. Se todas as operações que constituem uma transação forem bem-sucedidas, a transação será *consolidada*; ou seja, todas as operações serão executadas. Se uma ou mais operações que constituem uma transação falharem, a transação será *revertida*; ou seja, nenhuma das operações serão executadas. Utilizando os recursos transacionais do MQSeries Adapter Kernel, um adaptador de origem poderá realizar uma série de operações como uma única unidade, com a garantia de que todas as operações serão bem-sucedidas, se a transação for consolidada ou se nenhuma operação ocorrer se a transação for revertida.

Os recursos transacionais podem ser criados nos adaptadores utilizando o MQSeries Adapter Builder ou os métodos `begin`, `rollback` e `commit` na classe `EpicNativeAdapter` da API Java do kernel. Se um método transacional for chamado em um contexto ilegal (por exemplo, chamado o método `commit` sem ter chamado primeiro o método `begin`, chamado o método `begin` no escopo de outra transação), o kernel desconsidera a chamada e emite um aviso para rastreamento. Consulte o “Capítulo 5. Utilização de APIs do MQSeries Adapter Kernel” na página 99, para obter mais informações sobre a utilização da API.

Limitações

As seguintes limitações estão associadas aos recursos transacionais do kernel:

- As transações não são suportadas pelo método `sendRequestResponse`.

- As transações aninhadas (ou seja, as transações chamadas em outras transações) não são suportadas.
- As transações não são suportados por todos os modos de comunicação; consulte "Apêndice A. Modos de Comunicação" na página 103 para obter mais detalhes.

Rastreamento

Uma mensagem de rastreamento contém o estado de processamento de uma mensagem em um determinado ponto no kernel. Você pode utilizar as mensagens de rastreamento para ajudar a diagnosticar problemas com o kernel ou com os adaptadores. O MQSeries Adapter Kernel *Manual de Determinação de Problemas* discute a utilização do rastreamento com o kernel.

Utilização do MQSeries Adapter Kernel com o WebSphere Business Integrator e o WebSphere Application Server

Esta seção discute a utilização dos produtos do MQSeries Adapter Kernel com o WebSphere Business Integrator e o WebSphere Application Server. Para obter informações adicionais, consulte a documentação do WebSphere Business Integrator.

Interceptação do JMS

O WebSphere Business Integrator fornece um componente chamado de Interceptador do JMS que opera com o MQSeries Adapter Kernel e com o WebSphere Application Server Advanced Edition, para fornecer um caminho alternativo com as mensagens de entrega para os aplicativos de destino. O Interceptador do JMS executa no servidor do WebSphere Application Server's Enterprise JavaBeans (EJB). Esta seção fornece uma visão geral da funcionalidade do Interceptador do JMS. Para obter mais informações adicionais, incluindo os detalhes na configuração do WebSphere Business Integrator e do Interceptador do JMS, consulte a documentação do WebSphere Business Integrator. Consulte a "Configuração do kernel" na página 60, para obter mais informações da configuração do MQSeries Adapter Kernel para reconhecer o Interceptador do JMS como um destino. Utilização do Interceptador do JMS como destino é equivalente para enviar uma mensagem para um daemon do adaptador.

Antes do Interceptador do JMS puder ser utilizado, deve-se disponibilizar um operador do adaptador do bean da mensagem do MQSeries Adapter Kernel e os adaptadores do bean da sessão do servidor do Java ou adaptadores EAB para o lado de destino do kernel. Execute essas tarefas utilizando o MQSeries Adapter Builder. Em um ambiente do WebSphere Business Integrator, da operação do kernel no WebSphere Application Server é parecido para essas operações com o daemon do adaptador independente, exceto que o Interceptador do JMS recebe a mensagem como representante do operador e chama o operador do adaptador apropriado. Em um ambiente do MQSeries

Adapter Kernel independente, o daemon do adaptador inicia os operadores do adaptador, em que as mensagens recebidas são retornadas diretamente.

A seqüência dos eventos quando o MQSeries Adapter Kernel opera com o Interceptador do JMS está como a seguir:

1. O Interceptador do JMS, monitoramento de uma fila do JMS, recebe um objeto de mensagem do JMS, a partir de um cliente do EJB ou em um aplicativo não-EJB.
2. O Interceptador do JMS instancia um *bean de mensagem do operador* e transmite o objeto de mensagem para ele. O bean da mensagem do operador é uma instância de um *bean de sessão*, um tipo de bean corporativo que encapsula os dados temporariamente associado com um cliente específico.
3. O bean de mensagem do operador converte o objeto de mensagem do JMS no objeto de retenção de mensagem do MQSeries Adapter Kernel.
4. Com base nos valores do cabeçalho da mensagem, o kernel chama um adaptador EAB ou um adaptador EJB. Se o tipo de adaptador para ser chamado for um adaptador EAB, o fluxo de dados será um caso independente. Se o tipo de adaptador para ser chamado for um adaptador EJB, um manipulador EJB será chamado e desempenhará as tarefas a seguir:
   - Ele determina o bean de sessão de serviço correto (interface de origem) para chamar, o método apropriado para chamar e o tipo de parâmetro de entrada do método para o objeto `TerminalDataContainer`.
   - Ele converte os dados do aplicativo contidos no objeto de retenção da mensagem para a estrutura de dados do `TerminalDataContainer` apropriado, para o bean da sessão do serviço utilizando a classe `Mapper`. O objeto `TerminalDataContainer` contém o meta-dados do objeto de retenção da mensagem, mais os objetos do aplicativo. Em muitos casos, o objeto do aplicativo são os dados do corpo do objeto de retenção da mensagem da cadeia do documento XML.
   - Ele chama o bean da sessão do serviço, transmitindo o objeto `TerminalDataContainer` para o método apropriado em um bean da sessão do serviço. O bean da sessão do serviço, o qual faz parte do adaptador de serviço do Java, é o destino da mensagem.
5. Se uma resposta foi solicitada, o bean da mensagem do operador converte o objeto da resposta `TerminalDataContainer` para um objeto de retenção da mensagem, e envia a resposta utilizando o adaptador nativo.
6. Se um erro ocorrer, o bean da mensagem do operador coloca o objeto de retenção da mensagem em uma fila de erro utilizando o adaptador nativo.

Suporte do Idioma Nacional

O MQSeries Adapter Kernel fornece suporte do idioma nacional quando os adaptadores do Java são utilizados. O suporte do idioma nacional não é fornecido para os adaptadores C.

Capítulo 2. Planejamento da instalação do kernel

Este capítulo lista os pré-requisitos para e os componentes do MQSeries Adapter Kernel.

Para obter os últimos detalhes, consulte o site Web da família do produto do MQSeries em:

www.ibm.com/software/ts/mqseries/

A IBM reserva-se o direito de atualizar as informações mostradas aqui. Para obter as informações mais recentes a respeito de níveis de software suportados, consulte:

www.ibm.com/software/ts/mqseries/platforms/supported.html

Hardware

O MQSeries Adapter Kernel é executado no seguinte hardware:

- Uma máquina IBM PC (ou compatível) executando Windows NT 4.0, Service Pack 5 ou superior ou o Windows 2000, Service Pack 1.
- Uma máquina IBM RS/6000 executando o AIX versão 4.3.2 ou 4.3.3.
- Uma máquina HP Série 9000 executando o HP-UX versão 11.0.
- Uma máquina Sun SPARC ou UltraSPARC executando o Solaris versão 8.
- Uma máquina IBM AS/400 ou iSeries executando o OS/400 versão 4.4 ou 4.5.

  Nota: A instalação do MQSeries Adapter Kernel no OS/400 requer um sistema Windows para interagir com a máquina do AS/400. Consulte a seção "Pré-requisitos para instalação do OS/400" na página 36, para obter detalhes.

O MQSeries Adapter Kernel requer, no mínimo, 25 MB de espaço em disco, aproximadamente, para o código e os dados do produto.

Assegure-se de que existe espaço em disco suficiente disponível para receber os adaptadores. Seu tamanho depende do tamanho das estruturas de dados, da complexidade dos mapeamentos e do código personalizado utilizados. Veja, a seguir, alguns exemplos de tamanhos de adaptadores diferentes nos sistemas Windows. Seus adaptadores de site podem exigir mais ou menos espaço em disco. Cada exemplo representa o adaptador de origem, código do adaptador compilado, origem API e código API compilado no MB ou KB.

- Adaptador de origem para incluir um pedido de vendas: 1,89 MB

- Adaptador de destino para sincronizar um registro do cliente: 389 KB
- Adaptador de destino para sincronizar um registro do inventário: 161 KB
- Adaptador de destino para sincronizar um item: 249 KB
- Adaptador de destino para sincronizar um pedido de vendas: 579 KB

Além disso, reserve, no mínimo, 20 MB para espaço de trabalho para o kernel e os Adaptadores. Os requisitos de espaço de trabalho podem variar dependendo de diversos fatores, como número e tamanho de filas e tamanho dos arquivos de rastreamento.

Software

Esta seção lista o software suportado para ser utilizado com o MQSeries Adapter Kernel. Os níveis suportados são mostrados. Consulte o "Apêndice B. Configurações válidas" na página 109. Observe que os compiladores C são necessários nos sistemas de desenvolvimento, mas não nos sistemas de produção. Os compiladores C listados aqui foram testados com êxito com o MQSeries Adapter Kernel; outros compiladores C poderão operar corretamente com o kernel, mas não são oficialmente suportados.

Para sistemas Windows:

- Microsoft Windows NT versão 4.0, Service Pack 5 ou superior ou Microsoft Windows 2000, Service Pack 1. Para determinar a versão e o pacote de serviço do Microsoft Windows, abra o Windows Explorer, em seguida, dê um clique em **Help > Sobre o Windows**.
- Compilador Microsoft Visual C++ 6.0.
- MQSeries versão 5.2 com SupportPac MA88.
- IBM Java Development Kit (JDK) versão 1.2.2 ou 1.3.

  Nota: O Windows NT e Windows 2000 são atuais e apenas as plataformas em que o MQSeries Adapter Kernel suporta JDK versão 1.3.

Para AIX:

- Sistema operacional AIX versão 4.3.2 ou 4.3.3.
- IBM C Set++ para AIX versão 3.1.3.
- MQSeries versão 5.2 com o SupportPac MA88.
- Java Development Kit versão 1.2.2. JDK 1.3 não foi suportado.
- X Window System (X11R5 ou superior). É necessário para instalação, mas não em tempo de execução.

Para HP-UX:

- Sistema operacional HP-UX versão 11.0.

- Compilador HP-UX C/ANSI C. Consulte o arquivo `readme.txt` para obter mais detalhes.
- MQSeries versão 5.2 com SupportPac MA88.
- Java Development Kit versão 1.2.2. JDK 1.3 não foi suportado.
- X Window System (X11R5 ou superior). É necessário para instalação, mas não em tempo de execução.

Para Solaris:

- Ambiente operacional Solaris versão 8.
- Compiladores Sun Workshop C/C++. Consulte o arquivo `readme.txt` para obter mais detalhes.
- MQSeries versão 5.2 com SupportPac MA88.
- Java Development Kit versão 1.2.2. JDK 1.3 não foi suportado.
- X Window System (X11R5 ou superior). É necessário para instalação, mas não em tempo de execução.

Para OS/400:

- Sistema operacional OS/400 versão 4.4 ou 4.5, incluindo os seguintes programas:
  - Java Toolkit e Java Developer Kit versão 1.2.2. JDK 1.3 não foi suportado. O Java Toolkit e o Java Developer Kit são enviados como programas licenciados número 5769–JV1. Consulte a seção "Pré-requisitos para instalação do OS/400" na página 36 para obter detalhes adicionais sobre a versão do Java Developer Kit, necessária para a instalação do MQSeries Adapter Kernel em um sistema AS/400.
  - A opção Host Servers, enviada como programa licenciado número 5769–SS1, opção 12.
  - Qshell Interpreter, enviado como programa licenciado número 5769–SS1, opção 30.
  - TCP/IP, enviado como programa licenciado número 5769–TC1.
  - Integrated Language Environment C para AS/400, em que é enviado com o número do programa licenciado 5769–CX2.
- MQSeries versão 5.2 com SupportPac MA88.

Consulte a seção "Pré-requisitos para instalação do OS/400" na página 36 para obter os requisitos adicionais para a instalação do MQSeries Adapter Kernel no OS/400.

Os seguintes produtos são suportados com o MQSeries Adapter Kernel:

- MQSeries versão 5.2 com o SupportPac MA88

Nota: Se MQSeries não estiver sendo utilizado, outras mensagens do software como a implementação do Java Message Service (JMS) deve ser utilizado.

- MQSeries Integrator versão 1.1
- MQSeries Integrator versão 2

Consulte o "Apêndice B. Configurações válidas" na página 109 para obter uma lista de configurações válidas do MQSeries Adapter Kernel, MQSeries e MQSeries Integrator.

Pré-requisitos para instalação do OS/400

Esta seção descreve os pré-requisitos para a instalação do MQSeries Adapter Kernel em um sistema AS/400 ou iSeries. Consulte a Etapa 3 na página 44 para obter instruções detalhadas sobre a instalação do MQSeries Adapter Kernel em um sistema AS/400. Uma vez que os terminais AS/400 originalmente não suportam os gráficos Java, uma estação de trabalho com gráficos ativados como um sistema Windows é necessária para executar o programa de instalação da GUI com base em Java do kernel. A estação de trabalho pode interagir com o sistema AS/400 de uma das seguintes maneiras:

- Pela AWT remota, na qual todos os gráficos são processados no sistema AS/400 e exibidos na estação de trabalho. Este procedimento está descrito com mais detalhes na seção "Utilização da AWT remota".
- Como um cliente conectado, no qual a estação de trabalho processa e exibe os gráficos. Este procedimento está descrito com mais detalhes na seção "Utilização de um cliente conectado" na página 37.

Esta seção considera que você esteja utilizando um sistema Windows como a estação de trabalho com gráficos ativados.

Utilização da AWT remota

Quando a AWT remota é utilizada, é feito o processamento de gráficos em Java no sistema AS/400 e os gráficos são exibidos em uma estação de trabalho cliente conectada ao sistema AS/400. Esta seção descreve os requisitos que devem ser atendidos para instalar o MQSeries Adapter Kernel em um sistema AS/400 utilizando a AWT remota.

Os seguintes programas devem ser instalados com o OS/400:

- Java Toolkit e Java Developer Kit versão 1.2.2. O Java Toolkit e o Java Developer Kit são enviados como programas licenciados número 5769–JV1. Os recursos da AWT remota no OS/400 são fornecidos pelo Java Developer Kit.
- TCP/IP, enviado como programa licenciado número 5769–TC1. Para obter mais informações sobre o TCP/IP, consulte os documentos *AS/400 TCP/IP Fastpath Setup Information* e o *AS/400 TCP/IP Configuration*, em que estão

disponíveis a partir da biblioteca do AS/400 em www.ibm.com/servers/eserver/iseries/library/.

Os requisitos de uma estação de trabalho são:

- Uma máquina IBM PC (ou compatível) executando Windows 95, Windows 98, Windows NT ou Windows 2000.
- Uma conexão TCP/IP com o sistema AS/400.
- JDK 1.2.2 ou superior.

Para configurar e iniciar a AWT remota, execute as seguintes etapas:

1. Assegure-se de que o JDK 1.2.2 ou superior esteja instalado na estação de trabalho.
2. Assegure-se de que existe uma conexão TCP/IP entre o sistema AS/400 e a estação de trabalho.
3. Copie o arquivo `RAWTGui.jar` do diretório `/QIBM/ProdData/Java400/jdk12` no sistema AS/400 para um diretório na estação de trabalho.
4. Na estação de trabalho, mude para o diretório em que você copiou o arquivo `RAWTGui.jar` e inicie a AWT remota, digitando o seguinte comando:

   ```
   java -jar RAWTGui.jar
   ```

Nota: Devido ao uso intensivo dos recursos do processamento de gráficos Java em um sistema AS/400, é possível que a utilização AWT remota demore mais do que a utilização de um cliente conectado para instalar o MQSeries Adapter Kernel.

Para obter mais informações sobre a AWT remota, consulte a biblioteca do AS/400 em www.ibm.com/servers/eserver/iseries/library/.

Utilização de um cliente conectado

Quando um cliente conectado é utilizado para instalar o MQSeries Adapter Kernel em um sistema AS/400, é feito o processamento de gráficos em Java na estação de trabalho cliente, não no sistema AS/400. Esta seção descreve os requisitos que devem ser atendidos para instalar o MQSeries Adapter Kernel em um sistema AS/400 utilizando um cliente conectado.

Os seguintes programas devem ser instalados com o OS/400:

- Java Toolkit e o Java Developer Kit versão 1.2.2. O Java Toolkit e o Java Developer Kit são enviados como programas licenciados número 5769–JV1.
- A opção Host Servers, enviada como programa licenciado número 5769–SS1, opção 12.
- TCP/IP, enviado como programa licenciado número 5769–TC1.

Os requisitos de uma estação de trabalho são:

- Uma máquina IBM PC (ou compatível) executando Windows NT 4.0, Service Pack 5 ou Windows 2000, Service Pack 1.
- Uma conexão TCP/IP com o sistema AS/400.
- JDK 1.2.2 ou superior.

Componentes do kernel

Depois da instalação, o MQSeries Adapter Kernel reside em seu diretório raiz. Ele contém subdiretórios que, por sua vez, podem conter outros diretórios. A raiz e seus subdiretórios são listados, junto com um resumo dos arquivos que são mais importantes para a instalação e configuração.

root O nome padrão é `C:\Program Files\MQAK` nos sistemas Windows, `/usr/lpp/mqak` no AIX, `/MQAK` no HP-UX, `/opt/MQAK` no Solaris e `/QIBM/ProdData/mqak` no OS/400. Contém o seguinte:

- Todos os demais diretórios do MQSeries Adapter Kernel.
- O arquivo `aqmsetenv.bat` (sistemas Windows) ou `aqmsetenv.sh` (UNIX), que altera as variáveis de ambientes do sistema depois da instalação, se desejado.
- O arquivo `readme.txt`.
- O arquivo `aqmuninstall.bat` (sistemas Windows) ou `aqmuninstall.sh` (UNIX).

bin Contém o seguinte:

- Bibliotecas de classe e bibliotecas compartilhadas.
- Adaptador fornecidos como parte do kernel, apenas para utilização de verificação.
- O arquivo `aqmversion.bat` (sistemas Windows) ou `aqmversion.sh` (UNIX e OS/400), um script executado para exibir o número da versão do kernel.
- O arquivo `aqmcrtmsg.bat` (sistemas Windows) ou `aqmcrtmsg.sh` (UNIX e OS/400), um script executado para criar um arquivo XML utilizado para validar o arquivo de configuração antes de ser colocado em produção.
- O arquivo `aqmsndmsg.bat` (sistemas Windows) ou `aqmsndmsg.sh` (UNIX e OS/400), um script executado para validar o arquivo de configuração antes de ser colocado em produção.
- O arquivo `aqmstrad.bat` (sistemas Windows) ou `aqmstrad.sh` (UNIX e OS/400), um script executado para iniciar o daemon do adaptador.
- O arquivo `aqmstrtd.bat` (sistemas Windows) ou `aqmstrtd.sh` (UNIX e OS/400), um script executado para iniciar o servidor de rastreamento.

documentation
Contém a documentação do produto, incluindo o Centro de Informações.

runtimefiles
Contém os arquivos em tempo de execução do kernel.

samples

Contém amostras de adaptadores, configuração associada e arquivos utilitários. Você pode experimentá-las e aprender com elas.

Nota: O kernel deve ser utilizado com os adaptadores criados utilizando o MQSeries Adapter Builder. O kernel não deve ser utilizado por chamadas para as APIs de kernel a partir de códigos personalizados independentes. As amostras de adaptadores são fornecidas apenas como um auxílio para entender como o kernel funciona e é também um auxílio para os diagnósticos.

- Amostras de adaptadores.
- O arquivo de configuração do kernel, `aqmsetup`, com valores que suportam as amostras dos adaptadores. Consulte a seção "O arquivo de instalação" na página 66 para obter uma discussão sobre este arquivo.
- O arquivo de configuração do kernel, `aqmconfig.xml`, com valores que suportam as amostras dos adaptadores, incluindo os valores de rastreamento de amostras. Consulte a seção "O arquivo de configuração" na página 66 para obter uma discussão sobre este arquivo.

toolkit
Contém um kit de ferramentas de desenvolvimento de software (SDK) composto por:

- Arquivos de cabeçalhos.
- Arquivos de biblioteca utilizados durante a compilação nos sistemas Windows.

uninstall
Contém os arquivos utilizados para remover a instalação do kernel.

verification
Contém os seguintes arquivos que suportam a verificação da instalação do kernel:

- O arquivo `aqmverifyinstall.bat` (sistemas Windows) ou `aqmverifyinstall.sh` (UNIX e OS/400), um script executado para verificar a instalação do kernel em um computador.
- O arquivo `aqmcreateq.bat` (sistemas Windows) ou `aqmcreateq.sh` (UNIX e OS/400), um script que cria as filas de MQSeries para verificação. Consulte a seção "Criação de filas do MQSeries" na página 98.
- O arquivo `aqmconfig.xml`. Consulte a seção "O arquivo de configuração" na página 66 para obter uma discussão sobre este arquivo.
- O arquivo `aqmsetup`. Consulte a seção "O arquivo de instalação" na página 66 para obter uma discussão sobre este arquivo.
- O arquivo `aqminstalltest.xml`.

Capítulo 3. Instalação do kernel

Este capítulo discute as etapas necessárias para instalar e verificar o MQSeries Adapter Kernel. As instalação é composta pelas seguintes etapas gerais:

Etapa 1. Prepare a instalação. Consulte a seção "Preparação para a Instalação" para obter detalhes.

Etapa 2. Instale o kernel. Consulte a seção "Instalação do kernel" na página 43 para obter detalhes.

Etapa 3. Complete algumas das etapas da pós-instalação. Consulte o "Completando a pós-instalação" na página 46 para obter detalhes.

Etapa 4. Verifique a instalação. Consulte a seção "Verificação da Instalação" na página 48 para obter detalhes.

Este capítulo discute também os tópicos a seguir:

- Utilização da instalação silenciosa para instalar o MQSeries Adapter Kernel. Consulte o "Utilização da instalação silenciosa" na página 53 para obter detalhes.
- Atualziação do MQSeries Adapter Kernel a partir da versão mais recente. Consulte o "Atualização do kernel" na página 55 para obter detalhes.
- Remoção de uma instalação do MQSeries Adapter Kernel. Consulte o "Remoção do kernel" na página 56 para obter detalhes.

Após instalar o kernel, desempenhe as tarefas adicionais a seguir para prepará-las para a utilização:

1. Configure o kernel. Consulte a seção "Configuração do kernel" na página 60 para obter detalhes.
2. Configure o software opcional e o software de mensagens. Consulte a seção "Configuração do MQSeries e do MQSeries Integrator" na página 92 para obter detalhes.
3. Crie seus adaptadores utilizando o MQSeries Adapter Builder, em seguida, teste e implemente-as.
4. Inicie o kernel. Consulte a seção "Início do kernel" na página 93 para obter detalhes.

Preparação para a Instalação

Você deve ter autoridade de administrador ou de raiz para instalar o MQSeries Adapter Kernel. Você deve ter permissão para criar e acessar arquivos no local em que instalou o MQSeries Adapter Kernel e no local em que colocou os dois arquivos de configuração do kernel. Você deve ter o

diretório atual em seu caminho executável. Verifique se todas os IDs que executam o kernel tenham permissões de leitura, gravação e execução.

Você deve ter autoridade para executar as operações do MQSeries, como criar gerenciadores de filas, criar e acessar filas. Essas operações são executadas de maneiras diferentes em plataformas diferentes. Consulte o *MQSeries Administration Guide* de sua plataforma para obter mais informações.

O identificador do usuário que inicia os processos do kernel deve estar no grupo mqm. Existem dois tipos de processos do kernel:

- Daemon do adaptador, um para cada aplicativo de destino atendido pelo computador
- Servidor de rastreamento (opcional)

Observe que o adaptador de origem é executado no processo do aplicativo de origem. Qualquer daemon ou servidor que contenha o adptador de origem precisa ser inicializado para o adaptador de origem para efetuar a execução.

Você deve instalar e configurar o kernel para executar os adaptadores que foram criados. No entanto, você não deve instalar o kernel para instalar o MQSeries Adapter Builder ou para utilizá-lo para criar seus adaptadores.

Execute as etapas a seguir antes de iniciar a instalação:

- Leia o arquivo `readme.txt` no CD-ROM ou na rede local. É possível que neste arquivo contenha todas as informações importantes que foram disponibilizadas após a conclusão deste manual. Ele está no diretório raiz de instalação.
- Visite o site Web do MQSeries Web no endereço www.ibm.com/software/ts/mqseries/. É possível que ele contenha informações importantes que foram disponibilizados após a publicação deste manual, incluindo uma nova edição.
- Se estiver fazendo atualização de uma versão anterior do MQSeries Adapter Kernel, consulte a seção “Atualização do kernel” na página 55 para obter instruções.
- Verifique se os pré-requisitos de hardware e software foram atendidos. Consulte as seções “Hardware” na página 33 e “Software” na página 34 para obter detalhes. O MQSeries deve ser instalado e executado antes da verificação da instalação do MQSeries Adapter Kernel. Verifique se o suporte ao MQSeries Java foi instalado e configurado.

Instalação do kernel

Para instalar o MQSeries Adapter Kernel no sistema Windows (Windows NT ou Windows 2000), em uma plataforma UNIX (AIX, HP-UX ou Solaris) ou na plataforma OS/400 as seguintes etapas específicas do sistema operacional:

Nos sistemas Windows:

Etapa 1. Inicie o programa de instalação da seguinte maneira:

- Se estiver instalando a partir de uma rede local, vá para o diretório que contém os arquivos de instalação do MQSeries Adapter Kernel e execute o arquivo `install.bat`.
- Se estiver instalando a partir do CD-ROM, insira o CD-ROM do MQSeries Adapter Kernel na unidade de CD-ROM. Se a execução automática estiver ativada, o programa de instalação é iniciado automaticamente; se não estiver, execute o arquivo `install.bat` no diretório raiz do CD-ROM para iniciar o programa de instalação.

Nota: Nos sistemas Windows, não é necessário copiar o arquivo `install.bat` para outro local antes de executá-lo. Durante o processo de instalação, você deve escolher o local em que deseja instalar o MQSeries Adapter Kernel.

Etapa 2. Siga os avisos fornecidos pelo programa de instalação. Observe que, se você quiser instalar o MQSeries Adapter Kernel em um local que não seja o padrão (nos sistemas Windows, `C:\Arquivos de programas\MQAK`), deverá especificar o diretório de instalação como um nome de caminho completo, não um nome de caminho relativo.

No UNIX:

Etapa 1. Inicie o programa de instalação da seguinte maneira:

- Se estiver instalando a partir de uma rede local, vá para o diretório que contém os arquivos de instalação do MQSeries Adapter Kernel e execute o script `install.sh`.
- Se estiver instalando a partir do CD-ROM, insira o CD-ROM do MQSeries Adapter Kernel na unidade de CD-ROM e, se necessário, monte a unidade de CD-ROM de acordo com a documentação do sistema operacional. Execute o script `install.sh` no diretório raiz do CD-ROM.

Etapa 2. Siga os avisos fornecidos pelo programa de instalação. Observe que, se você quiser instalar o MQSeries Adapter Kernel em um local que não seja o padrão, deve-se especificar o diretório de instalação como um nome de caminho completamente qualificado, não como um nome de caminho relativo. Os diretórios de instalação padrão no UNIX são as seguintes:

- AIX: `/usr/lpp/mqak`
- HP-UX: `/MQAK`
- Solaris: `/opt/MQAK`

No OS/400:

Etapa 1. Verifique se todos os pré-requisitos listados nas seções "Hardware" na página 33, "pré-requisitos de software para OS/400" na página 35 e "Pré-requisitos para instalação do OS/400" na página 36 foram atendidos. Observe que a instalação do MQSeries Adapter Kernel no OS/400 utiliza um programa com base em InstallShield que exige a utilização de uma estação de trabalho que se conecte com o sistema AS/400; consulte a seção "Pré-requisitos para instalação do OS/400" na página 36 para obter mais detalhes.

Etapa 2. Crie um perfil do usuário denominado MQAKSRV no sistema AS/400 usando o comando **CRTUSRPRF** em um prompt Control Language (CL).

Etapa 3. Se você estiver utilizando uma AWT remota ou uma estação de trabalho cliente conectada para realizar a instalação, execute as seguintes etapas:

- Se estiver utilizando uma AWT remota para realizar a instalação, excute as seguintes etapas:
  a. Verifique se a AWT remota está configurada e em execução. Consulte a seção "Utilização da AWT remota" na página 36 para obter detalhes.
  b. Verifique se o arquivo `installAS400.jar` está acessível ao sistema AS/400. O arquivo deve estar no sistema de arquivos integrados (IFS) ou em um dispositivo conectado ao sistema AS/400. Se o arquivo estiver em um dispositivo conectado, utilize o comando Criar Link (**CRTLINK**) para criar um link simbólico para o arquivo.
  c. Para melhorar o desempenho do processo de instalação, execute o comando Criar Programa Java (**CRTJVAPGM**) no arquivo `installAS400.jar`.
  d. Execute o comando Executar Java (**RUNJVA**) como segue, em que *n.n.n.n* representa o endereço TCP/IP da estação de trabalho que está executando a AWT remota:

     ```
     RUNJVA CLASS(run)
     CLASSPATH('/installAS400.jar')
     PROP((os400.class.path.rawt 1) (RmtAwtServer 'n.n.n.n')
     (java.version 1.2))
     ```

- Se estiver utilizando uma estação de trabalho cliente conectada para executar a instalação, execute as seguintes etapas:

Etapa a. Verifique se os requisitos especificados na seção “Utilização de um cliente conectado” na página 37 foram atendidos.

Etapa b. Verifique se a opção Host Servers está instalada e em execução na máquina com o AS/400. Você pode iniciar o Host Servers utilizando o comando Iniciar Host Servers (**STRHOSTSVR**) no prompt CL.

Etapa c. Verifique se o TCP/IP está instalado e em execução na máquina com o AS/400. Você pode iniciar o TCP/IP utilizando o comando Iniciar TCP/IP (**STRTCP**) em um prompt CL.

Etapa d. Na estação de trabalho, abra um prompt de comandos e mude para o diretório `AS400` da mídia de instalação do MQSeries Adapter Kernel (rede local ou CD-ROM).

Etapa e. Digite o comando a seguir:

```
java -classpath installAS400.jar; run -os400
```

Etapa 4. O programa de instalação é iniciado e exibe o painel **Signon to AS/400**. Digite o endereço TCP/IP da máquina com AS/400 no campo **System:** e seu ID do usuário e senha nos campos correspondentes. Não verifique a caixa de seleção do **Default User**. Clique em **Next**.

Etapa 5. Siga os avisos fornecidos pelo programa de instalação. Dependendo da velocidade da rede e das máquinas, o processo de instalação pode levar até uma hora para ser concluído. Uma barra de progresso exibida na estação de trabalho indica o status da instalação.

Observe que, no OS/400, o MQSeries Adapter Kernel é sempre instalado no diretório `/QIBM/ProdData/mqak` na raiz do sistema de arquivos integrados (IFS).

Etapa 6. Defina as variáveis de ambiente CLASSPATH, PATH, e QIBM_MULTI_THREADED como a seguir:

- Inclua o diretório `/QIBM/ProdData/mqak/bin` na variável de ambiente CLASSPATH.
- Inclua o diretório `/QIBM/ProdData/mqak/bin` na variável de ambiente PATH.
- Defina a variável de ambiente QIBM_MULTI_THREADED como `Y`.

Etapa 7. Inclua a biblioteca MQAK na lista de bibliotecas QSYS.LIB.

A instalação do kernel está concluída. Uma vez instalado, o kernel está configurado para suportar a verificação, não para suportar a produção em seu local determinado. Verifique a instalação executando as etapas listadas na seção “Verificação da Instalação” na página 48. Após verificar a instalação, siga

as etapas em “Completando a pós-instalação” para definir as variáveis de ambiente e mover alguns dos arquivos de configuração para suportar a produção do seu site.

Instale o kernel em outros computadores conforme necessário.

Completando a pós-instalação

Após a instalação do kernel, desempenhe as seguintes etapas:

Etapa 1. Decida em que arquivos colocar o `aqmsetup` e `aqmconfig.xml`, que são utilizados para configurar o kernel. Para obter mais informações sobre esses arquivos, consulte a seção “Configuração do kernel” na página 60.

CUIDADO:
Se você não criar seus próprios arquivos de configuração, mas utilizar os arquivos de configuração fornecidos no diretório `samples` para produção, a instalação de uma nova versão do kernel os substituirá, destruirá sua configuração de produção.

Etapa 2. Crie um diretório para os dois arquivos de configuração. Eles não precisam estar localizados no mesmo diretório, mas é recomendável fazê-lo para simplificar. A localização dos diretórios externos do diretório no lugar que você instalou o MQSeries Adapter Kernel deixa poucos diretórios, se o kernel for remover a instalação mais tarde. O processo de remoção de instalação mantém os diretórios que contêm tudo, exceto os arquivos originais do MQSeries Adapter Kernel.

Etapa 3. Copie os arquivos `aqmsetup` e `aqmconfig.xml` do diretório `samples` para o local desejado. Você pode colocá-los em uma unidade de rede ou em outro local central que seja acessado por muitos computadores para tornar a atualização e backup mais fáceis.

Se você renomear o arquivo `aqmconfig.xml`, o kernel não funcionará corretamente. Você pode renomear o arquivo `aqmsetup`, desde que tenha definido uma variável de ambiente para indicá-lo corretamente em uma Etapa 5 na página 47.

Etapa 4. Utilize um editor de texto, edite o arquivo `aqmsetup` indicando o diretório desejado do arquivo `aqmconfig.xml`. Utilize um nome de caminho completo (não um nome de caminho relativo) como a localização do diretório. Não inclua o nome do arquivo no caminho. A seguir, veja um exemplo:

```
# Location of configuration file aqmconfig.xml.
AQMCONFIG=C:\Program Files\MQAK\Data\
```

Mesmo que o local desejado para o arquivo `aqmconfig.xml` for o mesmo diretório em que o arquivo `aqmsetup` reside, você deverá informar o nome do caminho completo aqui. Salve e feche o arquivo `aqmsetup`.

Etapa 5. Defina a variável de ambiente AQMSETUPFILE para apontar o local do arquivo `aqmsetup` (para a instância `C:\Program Files\MQAK\Data\aqmsetup` nos sistemas Windows, `/MQAK/data/aqmsetup` no UNIX ou `/home/`*`user_name`*`/aqmsetup` no OS/400). Observe que no OS/400, o arquivo `aqmsetup` deve estar sempre localizado no diretório pessoal IFS do usuário atual (que é, `/home/`*`user_name`*`).`

Se o kernel estiver instalado em uma unidade de rede, execute esta etapa para cada computador que acessá-lo.

Etapa 6. Se estiver utilizando o AIX e planeja utilizar os adaptadores de origem de linguagem C que são chamados a partir de um programa C, defina a variável de ambiente AIXTHREAD_SCOPE para o valor `S`. Para definir esta variável de ambiente no shell Bourne ou o shell Korn, digite o seguinte comando:

```
export AIXTHREAD_SCOPE=S
```

Para definir esta variável de ambiente no shell C, digite o seguinte comando:

```
setenv AIXTHREAD_SCOPE S
```

Para que a variável AIXTHREAD_SCOPE seja definida automaticamente quando você inicia o AIX, inclua este comando para o seu arquivo `.profile` (se você utilizar o shell Bourne ou o shell Korn) ou o arquivo `.cshrc` (se utilizar o shell C).

Consulte a Etapa 7 na página 24 para obter informações adicionais sobre as políticas de planejamento.

Etapa 7. Se necessário, defina a variável de ambiente THREADS_FLAG. Você deve definir esta variável apenas se *todas* as seguintes condições forem verdadeiras:

- Solaris é o sistema operacional que está sendo utilizado.
- A versão do Java Development Kit (JDK) utilizada é 1.2.2.
- O MQSeries está sendo utilizado para transportar mensagens.
- Seus adaptadores de origem e de destino estão escritas em C.

Se todas essas condições forem verdadeiras, defina a variável de ambiente THREADS_FLAG como `nativa`. Para definir esta variável de ambiente no shell Bourne ou o shell Korn, digite o seguinte comando:

```
export THREADS_FLAG=native
```

Para definir esta variável de ambiente no shell C, digite o seguinte comando:

```
setenv THREADS_FLAG native
```

Para que a variável THREADS_FLAG seja definida automaticamente quando você inicia no Solaris, inclua este comando para o seu arquivo `.profile` (se você utilizar o shell Bourne ou o shell Korn) ou o arquivo `.cshrc` (se você utilizar o C).

Após completar as etapas pós-instalação, desempenhe as tarefas a seguir para preparar o kernel para a utilização:

1. Prepare a instalação. Consulte a seção “Preparação para produção” na página 59.
2. Edite o arquivo de configuração. Consulte a seção “Configuração do kernel” na página 60 para obter detalhes.
3. Configure o MQSeries e o software opcional. Consulte a seção “Configuração do MQSeries e do MQSeries Integrator” na página 92.
4. Para os sistemas de produção, leve em consideração a seção “Recomendações sobre desempenho” na página 93.
5. Inicie o kernel. Consulte a seção “Início do kernel” na página 93.
6. Configure um plano de manutenção do kernel. Consulte a seção “Manutenção do kernel” na página 95.

Verificação da Instalação

Depois de instalar o kernel, verifique se ele foi instalado corretamente executando um script de verificação. O script envia uma mensagem de teste a partir de um aplicativo de origem utilizando um adaptador de origem, em seguida, utilize o kernel para o MQSeries. Em seguida, ele utiliza o kernel para receber a mensagem do MQSeries e, depois, chama um adaptador de destino. Todos esses processos são executados em um único computador.

Nesta verificação, o aplicativo de origem é uma fila do MQSeries chamada TEST1. O aplicativo de destino é outra fila do MQSeries chamada TEST2.

A verificação executa as seguintes tarefas:

- Verifica se o kernel, com o adaptador de origem e adaptador de destino fornecidos, enfileirou e roteou a mensagem de teste corretamente, utilizando o MQSeries como o software de mensagens, de ponta-a-ponta no computador.
- Verifica os arquivos `aqmconfig.xml` e `aqmsetup` fornecidos na instalação. Eles determinam a configuração do kernel. Consulte a seção “Configuração do kernel” na página 60 para obter mais informações sobre esses arquivos.

Você pode validar o arquivo de configuração antes de colocá-lo em produção. Consulte a seção "Validação do arquivo de configuração" na página 89.

Os scripts de verificação de instalação fornecidos com o MQSeries Adapter Kernel consideram que o MQSeries está instalado e configurado na máquina em que os scripts devem ser executados. Se estiver utilizando um software de mensagens que não seja o MQSeries, você poderá editar os scripts de verificação de instalação para suportar o software de mensagem da seguinte maneira:

1. Mude para o diretório de `verification` da instalação do kernel.
2. Abra o arquivo `aqmconfig.xml` em um editor de texto e altere a linha `<epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>` para `<epicmqppqueuemgr>`*`queue_manager_name`*`</epicmqppqueuemgr>`, em que *queue_manager_name* é o nome do seu gerenciador de filas.
3. Edite o arquivo `aqmverifyinstall` da seguinte maneira:
   - Se estiver executando a verificação de instalação em um sistema Windows, abra o arquivo `aqmverifyinstall.bat` em um editor de texto e altere a linha `aqmcreateq TEST2` para `aqmcreateq TEST2` *`queue_manager_name`*, em que *queue_manager_name* é o nome do seu gerenciador de filas.
   - Se estiver executando a verificação de instalação no UNIX ou OS/400, abra o arquivo `aqmverifyinstall.sh` em um editor de texto e altere a linha `aqmcreateq.sh TEST2` para `aqmcreateq.sh TEST2` *`queue_manager_name`*, em que *queue_manager_name* é o nome do seu gerenciador de filas.

Esta verificação utiliza alguns componentes, como um nome do adaptador de destino `com.ibm.epic.adapters.eak.test.InstallVerificationTest`, que não faz parte do kernel. Eles são fornecidos com o kernel apenas para fins de verificação de instalação.

Quando a verificação termina, o daemon do adaptador de verificação é parado.

O rastreamento não é ativado durante a verificação.

Procedimento de Verificação

Etapa 1. A verificação cria e utiliza três filas do MQSeries. Se essas filas tiverem mensagens nelas antes da verificação ser realizada, a verificação falhará. Elimine as mensagens nas seguintes filas:
- TEST2AIQ
- TEST2AEQ
- TEST2RPL

Etapa 2. Verifique se você está autorizado para instalar e verificar o kernel. Consulte "Preparação para a Instalação" na página 41.

Etapa 3. Inicie a verificação da seguinte maneira:

- Nos sistemas Windows, dê um clique duplo no arquivo `aqmverifyinstall.bat` no diretório `verification`. Como alternativa, abra um prompt de comando, modifique o diretório `verification` e execute `aqmverifyinstall.bat`.
- No UNIX, abra um terminal, vá para o diretório de `verification` e execute o arquivo `aqmverifyinstall.sh`.
- No OS/400, execute as seguintes etapas:
  a. Inicie uma sessão **qsh** digitando o comando **STRQSH** command.
  b. Copie o arquivo `/QIBM/ProdData/mqak/verification/aqmsetup` para o diretório pessoal (`/home/`*`user_name`*).
  c. Mude para o diretório `/QIBM/ProdData/mqak/verification`.
  d. Execute o arquivo `aqmverifyinstall.sh`.

O arquivo `aqmverifyinstall` contém comentários sobre como ele funciona.

Etapa 4. A mensagem `Installation Verification Test completed successfully` indica que a instalação foi bem-sucedida. Feche a janela de verificação, se necessário.

Etapa 5. Em caso de falha, examine a janela de verificação e o arquivo de log, `EpicSystemExceptionFile`*`nnnnnnnn`*`.log`, para determinar o erro.

Etapa 6. Consulte "Problemas comuns de verificação" para obter informações sobre os problemas comuns que podem ser encontrados durante a verificação e sobre as possíveis respostas.

Etapa 7. Se desejar, execute a verificação opcional. Consulte a seção "Verificação opcional" na página 53 para obter detalhes.

Etapa 8. Volte para o procedimento de instalação e configure o kernel para suportar a operação em seu local determinado. Vá para a Etapa 1 na página 46.

Problemas comuns de verificação

Esta seção lista os problemas comuns que podem ser encontrados durante a verificação com as possíveis soluções. As informações importantes nas mensagens de exceção estão destacadas em **negrito**.

Problema: O arquivo `aqmsetup` não foi encontrado.

Resposta: Verifique se a variável de ambiente AQMSETUPFILE está definida para a localização do arquivo `aqmsetup` no diretório `verification`.

Mensagem de exceção:

```
com.ibm.epic.adapters.eak.nativeadapter.EpicNativeAdapter::main: caught
throwable with message <AQM0002: com.ibm.epic.adapters.eak.common.
AdapterDirectory::getProperties():
Received exception <com.ibm.epic.adapters.eak.common.AdapterException>
Message information: <AQM0002: com.ibm.epic.adapters.eak.common.
AdapterCfg::readConfig(String):
Received exception <java.io.FileNotFoundException> Message information:
<C:\aqmsetup> Additional program information <>.>
Additional program information <Error Reading Configuration File
[File or Keys in file may not exist]>.>
```

Problema: O arquivo `aqmconfig.xml` não foi encontrado.

Resposta: Edite o arquivo `aqmsetup` no diretório de `verification` e verifique se a entrada `AQMCONFIG=` indica para o diretório de `verification`. Utilize o nome do caminho completo. Verifique também se o arquivo `aqmconfig.xml` está localizado no diretório `verification`.

Mensagem de exceção:

```
com.ibm.epic.adapters.eak.common.AdapterException: MessageID <AQM0002>
<AQM0002: com.ibm.epic.adapters.eak.common.AdapterDirectory::
getProperties(): Received exception
<java.io.FileNotFoundException> Message information:
<AQMCONFIG.xml> Additional program information <>.>
```

Problema: A fila na qual a mensagem deve ser colocada não existe.

Resposta: Utilize o MQSeries para garantir que a fila especificada na mensagem de exceção (TEST2AIQ quando a instalação está sendo verificada) existe e pode aceitar mensagens. Consulte a seção "Criação de filas do MQSeries" na página 98.

Mensagem de exceção:

```
com.ibm.epic.adapters.eak.nativeadapter.EpicNativeAdapter::main: caught
throwable with message
<AQM0107: com.ibm.epic.adapters.eak.nativeadapter.LMSMQbase::
createMQOutputQueue(String):
Received MQException creating queue, QManager name <DEFAULT>
Queue name <TEST2AIQ>:
completion code <2> reason code <2085>.>
```

Problema: O adaptador de destino não foi encontrado.

Resposta: Verifique se o adaptador de destino especificado na mensagem existe:
`com.ibm.epic.adapters.eak.test.InstallVerificationTest` . Verifique se a variável de ambiente CLASSPATH inclui o diretório do kernel `bin`.

Mensagem de exceção:

```
com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker::sendException
(Throwable, String):Thread-2:
Message <<TEST2> <2000.05.18.09.41.43.781> <<Processing Messages.>
```

```
<com.ibm.epic.adapters.eak.common.AdapterException: MessageID <AQM0002>
<AQM0002: com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker:
:instantiateClass(String, Class[], Object[]): Received exception
<java.lang.ClassNotFoundException> Message information:
<com.ibm.epic.adapters.eak.test.InstallVerificationTest>
Additional program information <[Cannot obtain Class for class name
<com.ibm.epic.adapters.eak.test.InstallVerificationTest>]>.>>>>
```

Problema: Não foi encontrado um adaptador para carregar a entrega da mensagem. O identificador lógico de destino não tem uma entrada no arquivo `aqmconfig.xml` para o tipo do corpo e categoria do corpo especificados na mensagem da fila.

Resposta: Durante a verificação, a causa mais provável desta mensagem de exceção é a existência de mensagens em uma fila denominada TEST2AIQ antes da verificação. Apague todas as mensagens da fila TEST2AIQ e repita a verificação. A única entrada para um nome da classe de comandos para o aplicativo TEST2 no arquivo `aqmconfig.xml` no diretório `verification` é para um tipo do corpo TESTBOD e uma categoria do corpo OAG.

Mensagem de exceção:

```
com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker::sendException
(Throwable, String):Thread-2: Message <<TEST2> <2000.05.18.10.28.43.105>
<<Processing Messages.> <com.ibm.epic.adapters.eak.common.
AdapterException:
MessageID <AQM0401> <AQM0401: com.ibm.epic.adapters.eak.
adapterdaemon.EpicAdapterWorker::processMessage(EpicMessage):
Cannot obtain Command class name to load for a received message.>>>>
```

Problema: O gerenciador de filas de verificação não foi iniciado.

Resposta: Verifique se o gerenciador de filas do MQSeries padrão foi iniciado com êxito.

Mensagem de exceção:

```
com.ibm.epic.adapters.eak.common.AdapterException: Message ID <AQM0104>
<AQM0104: com.ibm.epic.adapters.eak.nativeAdapter.queueCollection::
constructor(String,String,boolean,String,String,int):
Received MQException creating QManager connection for
QManager name <QMGRNAME>
MQ Message information: completion code <2> reason code <2059>.>
```

Problema: Ocorreu um erro geral no MQSeries.

Resposta: Verifique se o MQSeries está instalado e configurado corretamente e se está sendo executado na máquina. Examine o código de razão de MQException e utilize o documento *MQSeries Messages* para determinar a causa do código de razão.

Mensagem de exceção:

```
Received MQException "ACTION ATTEMPTED." Message information:
completion code <completion_code> reason code <reason_code>
```

Verificação opcional

Depois de verificar se o kernel foi instalado corretamente no primeiro computador, você pode, opcionalmente, executar as seguintes etapas:

1. Verificar se o kernel foi instalado corretamente no segundo computador, utilizando a mesma verificação.
2. Verificar se é possível enviar uma mensagem de teste de um adaptador de origem em um computador para um adaptador de destino em outro computador. Configurar e executar manualmente esta verificação. Se você quiser desenvolver esta verificação modificando os arquivos de verificação originais fornecidos com o kernel, retenha uma cópia dos arquivos de verificação originais para fins de backup.

Utilização da instalação silenciosa

O MQSeries Adapter Kernel pode ser instalado em todas as platformas utilizando *instalação silenciosa*. A instalação silenciosa ativa a transferência secundária do programa de instalação do MQSeries Adapter Kernel, em que, você deve selecionar manualmente as opções de instalação que desejar. A instalação silenciosa é utilizada quando você deseja instalar a configuração padrão em máquinas múltiplas.

Para instalar o kernel silenciosamente, desempenhe as etapas específicas do sistema operacional:

Nos sistemas Windows:

Etapa 1. Abra um prompt de comando e mude o diretório que contenha os arquivos de instalação do MQSeries Adapter Kernel.

Etapa 2. Digite o comando a seguir:

```
java -cp install.jar run -P product.installLocation=
 "install_location"
-silent
```

em que *install_location* é o local de instalação desejado (para a instância, `D:\mqak`).

No UNIX:

Etapa 1. Em um terminal, altere para o diretório que contenha os arquivos de instalação do MQSeries Adapter Kernel. Se estiver instalando a partir do CD-ROM, insira o CD-ROM do MQSeries Adapter Kernel na unidade de CD-ROM e, se necessário, monte a unidade de CD-ROM de acordo com a documentação do sistema operacional.

Etapa 2. Digite o comando a seguir:

```
java -cp install.jar run -P product.installLocation=
 "install_location"
-silent
```

em que *install_location* é a localização de instalação desejado (para instância, `/opt/mqak`).

No OS/400:

Se você não estiver utilizando um cliente conectado para acessar a máquina AS/400, desempenhe as etapas a seguir:

Etapa 1. Verifique se o arquivo `installAS400.jar` está acessível ao sistema AS/400. O arquivo deve estar no sistema de arquivos integrados (IFS) ou em um dispositivo conectado ao sistema AS/400. Se o arquivo estiver em um dispositivo conectado, utilize o comando Criar Link (**CRTLINK**) para criar um link simbólico para o arquivo.

Etapa 2. Para melhorar o desempenho do processo de instalação, execute o comando Criar Programa Java (**CRTJVAPGM**) no arquivo `installAS400.jar`.

Etapa 3. Dependendo se você estiver utilizando um prompt CL ou uma sessão **qsh**, digite um dos seguintes comandos:

- Se você estiver utilizando um prompt CL, digite o comando a seguir:

  ```
  RUNJVA CLASS(run)
  CLASSPATH('/installAS400.jar')
  PROP((java.version 1.2)) PARM('-silent')
  ```

- Se você estiver utilizando uma sessão **qsh**, digite o seguinte comando:

  ```
  java -Djava.version=1.2 -classpath installAS400.jar run -silent
  ```

Se você estiver utilizando um cliente conectado para interface com o sistema AS/400, desempenhe as etapas a seguir:

Etapa 1. Verifique se os requisitos especificados na seção “Utilização de um cliente conectado” na página 37 foram atendidos.

Etapa 2. Verifique se a opção Host Servers está instalada e em execução na máquina com o AS/400. Você pode iniciar o Host Servers utilizando o comando Iniciar Host Servers (**STRHOSTSVR**) no prompt Linguagem de Controle (CL).

Etapa 3. Verifique se o TCP/IP está instalado e em execução na máquina com o AS/400. Você pode iniciar o TCP/IP utilizando o comando Iniciar TCP/IP (**STRTCP**) em um prompt CL.

Etapa 4. Na estação de trabalho, abra um prompt de comandos e mude para o diretório `AS400` da mídia de instalação do MQSeries Adapter Kernel (rede local ou CD-ROM).

Etapa 5. Digite o seguinte comando:

```
java -cp installAS400.jar run -silent -os400 machine_name user_ID
password
```

em que *machine_name* é o endereço TCP/IP do sistema AS/400, *user_ID* é o ID do usuário e *password* é a sua senha.

Atualização do kernel

Se você instalou o MQSeries Adapter Kernel versão 1.0, com ou sem o Corrective Service Diskette (CSD) ou MQSeries Adapter Kernel versão 1.1 com um nível de modificação atual, desempenhe as etapas a seguir antes de instalar o MQSeries Adapter Kernel versão 1.1 com o nível de modificação atual:

Etapa 1. Faça o backup dos arquivos `aqmsetup` e `aqmconfig` (`aqmconfig.properties` ou `aqmconfig.xml`) para um local externo do diretório de instalação do MQSeries Adapter Kernel.

Etapa 2. Se um MQSeries Adapter Kernel CSD estiver instalado, remova a instalação como a seguir:

- No Windows NT, utilize um dos seguintes métodos:
  - No menu Iniciar do Windows NT, clique em **Programas > MQSeries Adapter Kernel > Remover CSD**.
  - Utilize o utilitário Adicionar/Remover Programas no Painel de Controle.
  - Execute o arquivo `aqmuninstallCSD.bat` no diretório raiz do kernel.
  - Abra um prompt de comandos, mude para o diretório raiz do kernel e digite o seguinte comando:

    ```
    java uninstallCSD
    ```
- No AIX, mude para o diretório raiz do kernel e digite um dos seguintes comandos:

  ```
  aqmuninstallCSD.sh
  ```

  ```
  java uninstallCSD
  ```

Etapa 3. Remova a instalação do MQSeries Adapter Kernel como a seguir:

- No Windows NT, utilize um dos seguintes métodos:
  - No menu Iniciar do Windows NT, clique em **Programas > MQSeries Adapter Kernel > Desinstalar o MQSeries Adapter Kernel**.
  - Utilize o utilitário Adicionar/Remover Programas no Painel de Controle.
  - Execute o arquivo `aqmuninstall.bat` do diretório raiz do kernel.

– Abra um prompt de comandos, mude para o diretório raiz do kernel e digite o seguinte comando:

```
java uninstall
```

- No AIX, mude para o diretório raiz do kernel e digite um dos seguintes comandos:

```
aqmuninstall.sh
```

```
java uninstall
```

Etapa 4. Instale o MQSeries Adapter Kernel versão 1.1. Consulte a seção "Instalação do kernel" na página 43 para obter detalhes.

Etapa 5. Restaure os arquivos `aqmsetup` e `aqmconfig` para os seus locais anteriores no diretório de instalação do MQSeries Adapter Kernel. Se necessário, converta o arquivo `aqmconfig.properties` para um arquivo no `aqmconfig.xml`. Para obter mais informações sobre o arquivo `aqmconfig.xml`, consulte a seção "O arquivo de configuração" na página 66.

Remoção do kernel

Existem diversas maneiras de remover o kernel. Observe que o processo de remoção da instalação não remove quaisquer arquivos ou diretórios criados após a instalação do kernel. Isso inclui todos os arquivos de log e arquivos de dados copiados pelo usuário.

- Nos sistemas Windows, utilize um dos seguintes métodos:
  – No menu Iniciar, clique em **Programas > IBM MQSeries Adapter Kernel > Remover a Instalação do MQSeries Adapter Kernel**.
  – Utilize o utilitário Adicionar/Remover Programas no Painel de Controle.
  – Execute o arquivo `aqmuninstall.bat` no diretório raiz do kernel.
  – Para remover a instalação silenciosa do Kernel (isto é, sem o prompt do programa de remoção da instalação você pode obter mais informações ou confirmação), abra um prompt de comando, altere o diretório de instalação do kernel e digite o seguinte comando:

```
java -cp uninstall.jar run -silent
```

- No UNIX, mude para o diretório raiz do kernel e digite o seguinte comando:

```
aqmuninstall.sh
```

  Para remover a instalação silenciosa do kernel, (isto é, sem o programa de remoção de instalação para você obter mais informações ou confirmação), altere o diretório raiz do kernel e digite o comando a seguir:

```
java -cp uninstall.jar run -silent
```

- No OS/400, utilize um dos seguintes métodos para remover a instalação do kernel.

– Se você estiver utilizando a AWT remota para remover a instalação do kernel, desempenhe as etapas a seguir:

  Etapa 1. Verifique se a AWT remota está configurada e em execução. Consulte a seção "Utilização da AWT remota" na página 36 para obter detalhes.

  Etapa 2. Para melhorar o desempenho do processo de remoção de instalação, execute o comando Criar Programa Java (**CRTJVAPGM**) no arquivo `/QIBM/ProdData/mqak/uninstall/uninstall.jar`.

  Etapa 3. Execute o comando Executar Java (**RUNJVA**) como segue, em que *n.n.n.n* representa o endereço TCP/IP da estação de trabalho que está executando a AWT remota:

  ```
  RUNJVA CLASS(run)
  CLASSPATH('/QIBM/ProdData/mqak/uninstall/uninstall.jar')
  PROP((os400.class.path.rawt 1) (RmtAwtServer 'n.n.n.n')
  (java.version 1.2))
  ```

– Se você estiver utilizando uma estação de trabalho cliente conectado para remover a instalação do kernel, desempenhe as etapas a seguir:

  Etapa 1. Verifique se os requisitos especificados na seção "Utilização de um cliente conectado" na página 37 foram atendidos.

  Etapa 2. Verifique se a opção Host Servers está instalada e em execução na máquina com o AS/400. Você pode iniciar os Host Servers utilizando o comando Iniciar Host Servers (**STRHOSTSVR**) no prompt Linguagem de Controle (CL).

  Etapa 3. Verifique se o TCP/IP está instalado e em execução na máquina com o AS/400. Você pode iniciar o TCP/IP utilizando o comando Iniciar TCP/IP (**STRTCP**) em um prompt CL.

  Etapa 4. Copie os arquivos `uninstall.jar` e `uninstall.dat` no diretório `/QIBM/ProdData/mqak/uninstall` no sistema AS/400 para um diretório da estação de trabalho cliente.

  Etapa 5. Digite o seguinte comando:

  ```
  java -classpath uninstall.jar; run -os400
  ```

  Para remover a instalação silenciosa do kernel (isto é, sem o prompt de programa de remoção de instalação para você obter mais informações ou confirmação), digit o seguinte comando:

  ```
  java -cp uninstall.jar run -silent -os400 machine_name user_ID
  password
  ```

  em que *machine_name* é o endreço TCP/IP do sistema AS/400, *user_ID* é o ID do usuário e *password* é a sua senha.

– Se você estiver trabalhando diretamente em um sistema AS/400 em um prompt CL ou sessão **qsh** e deseja remover a instalação silenciosa do

kernel (isto é, sem um o prompt de programa de remoção de instalação para você obter mais informações e confirmação), digite um dos seguintes comandos:

- Em um prompt CL:

```
RUNJVA CLASS(run)
CLASSPATH('/uninstall.jar')
PROP((java.version 1.2)) PARM('-silent')
```

- Em uma sessão **qsh**:

```
java -Djava.version=1.2 -classpath uninstall.jar run -silent
```

Capítulo 4. Utilização do kernel

Este capítulo contém as seguintes informações sobre como utilizar o kernel:

Preparação para produção

Antes de colocar o kernel no processo de produção, execute as seguintes tarefas:

1. Crie a arquitetura geral do sistema, incluindo o Pacote do MQSeries Adapter, MQSeries ou outro software de mensagens e, opcionalmente, o MQSeries Integrator, com base nos requisitos e nas condições do local. Normalmente, a arquitetura é exclusiva de cada local.
2. Crie os adaptadores de origem requerido e os adaptadores de destino utilizando o MQSeries Adapter Builder, a seguir, teste e disponibilize-os.
3. Desenvolva interfaces específicas do aplicativo fora do Pacote do MQSeries Adapter para os seguintes objetivos:
   - Permitir que o adaptador de origem obtenha os dados de aplicativo do aplicativo de origem
   - Permitir que o aplicativo de destino obtenha os dados de mensagens do adaptador de destino

   O uso exato da interface específica do aplicativo depende das características do aplicativo de origem e do aplicativo de destino. Alguns exemplos de interfaces específicas do aplicativo incluem:
   - Chamadas da API e saídas do usuário
   - Leituras e gravações de arquivos
   - Disparos de bancos de dados
   - Filas de mensagens
4. Configure o kernel para suportar o fluxo em tempo de execução: envio, roteamento, rastreamento e entrega de mensagens. Consulte a seção "Configuração do kernel" na página 60 para obter informações sobre como configurar o kernel.

5. Configure o MQSeries ou outro software de mensagens e, opcionalmente, o MQSeries Integrator para suportar a arquitetura geral do sistema. Consulte a seção "Configuração do MQSeries e do MQSeries Integrator" na página 92.
6. Se necessário, desenvolva classes de início de sessão Java para suportar a entrega de mensagens. São específicas de cada aplicativo de destino. Serão necessárias apenas se o adaptador de destino exigir informações para início de sessão e conexão com o aplicativo.
7. Teste todo o sistema; ou seja, o MQSeries Adapter Kernel com seus adaptadores de origem e de destino, suas interfaces específicas do aplicativo e seu código personalizado; antes de colocar o sistema em produção.
8. Implemente o sistema no ambiente de produção.
9. Ative o kernel iniciando um ou mais adaptadores daemons e opcionalmente rastreio os servidores. Certifique-se de que o aplicativo de origem tenha sido iniciado. Se o adaptador de origem for executado no processo do aplicativo de origem, ele será automaticamente iniciado com o aplicativo de origem; nenhuma etapa extra será necessária para iniciá-lo. Qualquer daemon ou servidor que contenha o adaptador de origem precisa ser iniciado. Consulte a seção "Início do kernel" na página 93.

Configuração do kernel

Esta seção discute a configuração do kernel para a utilização do seu ambiente. A "Visão geral da configuração" fornece uma visão geral conceptual da configuração kernel. "Arquivos incluídos na inicialização e configuração" na página 65, discute vários arquivos que definem juntos uma configuração do MQSeries Adapter Kernel. "O arquivo de instalação" na página 66 discute o arquivo `aqmsetup`, que define várias configurações iniciais do kernel. "O arquivo de configuração" na página 66 discute o arquivo `aqmconfig.xml`, que fornece o kernel com as informações de configuração específica como os nomes de aplicativos de origem e destino, adaptadores de origem e destino, filas e gerenciadores de fila, modos de comunicação, registro e especificações de rastreamento.

Visão geral da configuração

Esta seção fornece uma visão geral conceptual da configuração kernel. Ela é importante para compreender o fluxo em tempo de execução do kernel antes de configurar o kernel. Esta seção discute o fluxo em tempo de execução em um nível simplificado. Consulte "Fluxo em tempo de execução" na página 14 para obter mais informações em um fluxo em tempo de execução.

Em muitos níveis básicos, a configuração do MQSeries Adapter Kernel é a unidade de dados que fluem entre os aplicativos. A configuração deve obter também os seguintes fatores dentro da conta:

- Os aplicativos que recebem os dados.
- Os adaptadores que são exigidos em um lado de origem e os adaptadores de destino, daemons do adaptador e operadores que são requeridos em um lado de destino.
- Os modos de comunicação, formata de enfileiramento e mecanismos de transporte que são utilizados.

As estruturas de dados e o formato de dados são diferente para cada aplicativo em uma configuração. Por exemplo, se uma configuração inclui dois aplicativos, A e B, que envia cada dados de ordem de compra para o aplicativo C, os dados a partir do aplicativo A é semelhante ao formato diferente, e possui significados de tag diferente a partir dos dados do aplicativo B. Para prevenir o aplicativo C, se necessário, reconheça e analise dois diferentes fluxos de dados a partir dos dois aplicativos diferentes, cada dado do aplicativo é convertido em uma *mensagem de integração* que é um formato de *integração-neutro*. Normalmente, o formato de integração neutro é um padrão da indústria com base no XML. Apenas o formato de dados que o aplicativo C precisa para deixá-lo capaz de reconhecer e analisar é o formato de integração neutral.

A Figura 3 mostra o fluxo de dados a partir dos aplicativos A e B para os aplicativos C e D. A figura a seguir, é uma explicação dos vários fluxos de dados que ela representa.

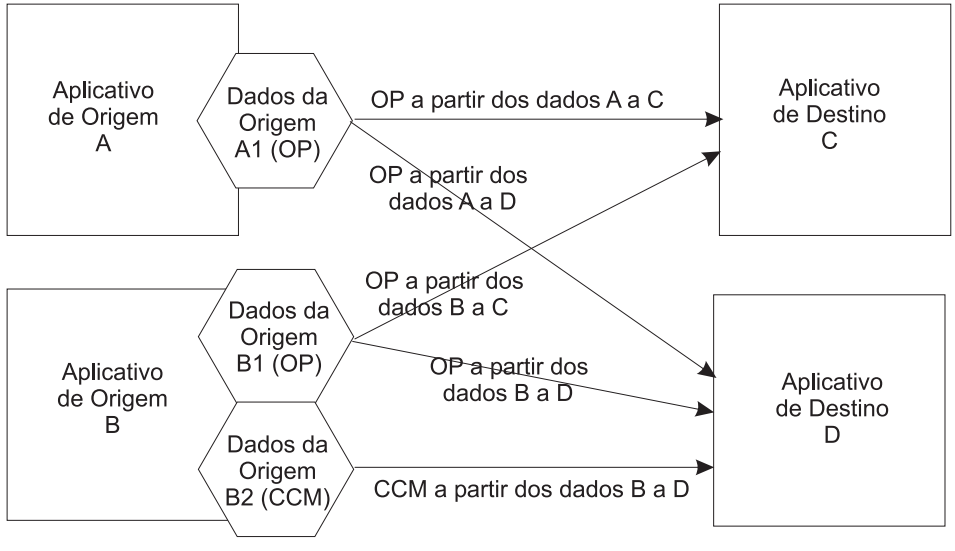


Figura 3. Os aplicativos conectados pelo fluxo de dados em uma configuração simples

Na Figura 3, os dados da ordem de compra do aplicativo A fluxo para os aplicativos C e D, dados da ordem de compra do aplicativo B também o fluxo

para os aplicativos C e D e os dados do comprovante de compra do material do aplicativo B fluxo apenas para o aplicativo D. Em alguns casos, os dados são convertidos para um formato da indústria padrão sendo enviado para os aplicativos de destino. Em alguns casos, os dados da ordem de compra são encaminhados para os aplicativos A e B, os dados são convertidos para um formato XML padrão representando os dados de ordem de compra. Em alguns casos, os dados do comprovante de compra do material são encaminhados para o aplicativo B, os dados são convertidos para um formato XML padrão representando os dados do comprovante de compra do material.

Um transporte de comunicação como um MQSeries ou uma implementação do Java Message Service (JMS) é utilizado para enviar os dados para o aplicativo de destino ou aplicativos. A mensagem de integração é convertida em um *formato de enfileiramento* requerida pelo transporte de comunicação específico, em seguida, entregue o transporte de comunicação (por exemplo, uma fila MQSeries). Cada aplicativo de destino pode utilizar um transporte de comunicação diferente e formato de enfileiramento para as mensagens recebidas. Para a instância, o aplicativo C pode ser MQSeries para as mensagens recebidas e aplicativo D pode utilizar JMS, como mostrado na Figura 4 na página 63. Neste caso, todas as mensagens de integração seguem para o aplicativo C (as quais são, os dados de ordem de compra dos aplicativos A e B) são convertidos para um formato de enfileiramento do MQSeries, e todas as mensagens de integração seguem para o aplicativo D (as quais são, os dados de ordem de compra dos aplicativos A e B assim como, os dados do comprovante de compra do material do aplicativo B) são convertidos para um formato de enfileiramente JMS. O MQSeries Adapter Kernel utiliza os mecanismos a seguir para desempenhar essas conversões:

- Um *adaptador de origem* é utilizado para converter os dados de aplicativo para uma mensagem de integração. Os adaptadores de origem são criados no MQSeries Adapter Builder.
- O *adaptador nativo* é utilizado para converter a mensagem de integração em uma *mensagem de comunicação*. O adaptador nativo utiliza o *serviço da mensagem lógica* (LMS- logical message service) para converter a mensagem para o transporte através do transporte de comunicação, o LMS é específico para o transporte de comunicação sendo utilizado. O LMS utiliza um *formatador* para enfileirar a mensagem no transporte.

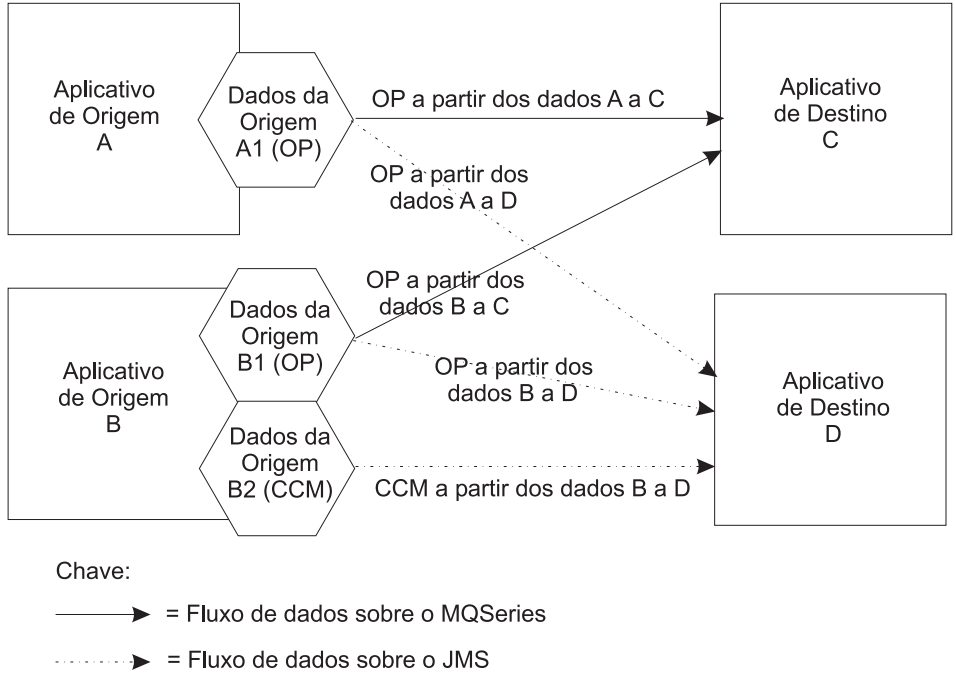


Figura 4. Os aplicativos conectados pelos transportes de comunicação diferente em uma configuração simples

O fluxo de dados a partir do aplicativo da mensagem de integração para a mensagem de comunicação é mostrado na Figura 5 e na Figura 6 na página 64. Quando uma mensagem de comunicação é recebida no destino, as conversões são revertidas: o adaptador nativo converte a mensagem de comunicação de volta para a mensagem de integração; em seguida; se necessário, o adaptador de destino converte a mensagem de integração no formato de dados requerido pelo aplicativo de destino.

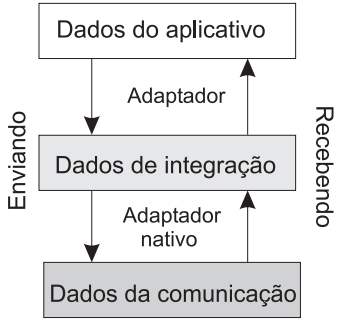


Figura 5. Conversão dos dados

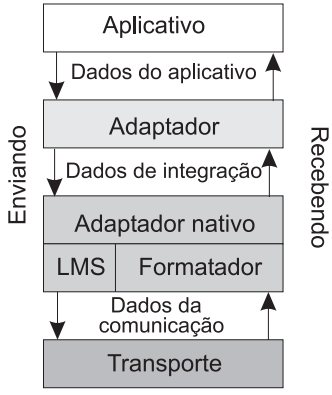


Figura 6. Fluxo de dados

Cada ponto em que os dados são transferidos deve ser representado em um arquivo de configuração do kernel. Existem três requerimentos de configuração lógica que são divididos pelo identificador de aplicativo (origem ou destino). Um identificador de aplicativo pode ser um aplicativo de origem ou um aplicativo de destino, dependendo se as mensagens são transferidos no aplicativo (destino) ou encaminhados para o aplicativo (origem). O arquivo de configuração deve incluir os tipos de informação a seguir:

- Comunicação
  - Para onde os dados se necessário são enviados
  - O transporte de comunicação para a utilização
  - O método de enfileiramento de comunicação (formatador) para a utilização
  - Os requisitos de comunicação sublinhados, como um gerenciador de fila do MCSeries e os nomes da fila do MQSeries, uma produção da conexão de fila do JMS e os nomes de fila do JMS
- Adaptadores (apenas para o lado de destino)
  - Os adaptadores que são requeridos para processar os dados
  - O tipo de adaptadores utilizados (EAB ou EJB)
  - Informações adicionais para o tipo de adaptador específico utilizado
  - Para o MQSeries Adapter Kernel independente, o daemon do adaptador e as informações do operador
- Outros
  - Especificações de rastreamento
  - Especificações de registro

O fluxo de dados como relacionado para as diferentes partes da configuração é mostrado na Figura 7.

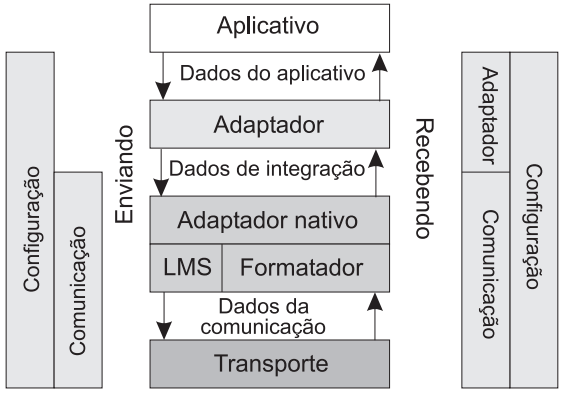


Figura 7. Fluxo de dados relacionados para a configuração

Consulte a "Sintaxe e organização do arquivo de configuração" na página 67, para obter mais informações no mapeamento nesses requisitos de configuração dos elementos XML no arquivo `aqmconfig.xml`, que controla esses aspectos da configuração do kernel. "Configurações comuns" na página 79, lista várias configurações comuns.

Arquivos incluídos na inicialização e configuração

A configuração do kernel é determinada por vários arquivos personalizáveis. Utilizando um editor de texto padrão, edite os arquivos para configurar o kernel para seu local. Os seguintes arquivos estão incluídos na configuração do kernel:

- O arquivo `aqmsetenv.bat` (sistemas Windows) ou `aqmsetenv.sh` (UNIX), que define as variáveis de ambiente. Edite esse arquivo para alterar as variáveis de ambiente do sistema, após a instalação, se desejado. As variáveis de ambiente definidas por esse arquivo incluem PATH, CLASSPATH e LIBPATH. Essas variáveis são definidas automaticamente pelo programa de instalação em sistemas Windows. Para definir as variáveis automaticamente quando você inicia o UNIX, inclua os valores especificados no arquivo `aqmsetenv.sh` para o arquivo `.profile` (se você utilizar o shell Bourne ou o shell Korn) ou o arquivo `.cshrc` (se você estiver utilizando o shell C).

  Para obter mais informações sobre a definição das variáveis de ambientes apropriadas no OS/400, consulte a Etapa 6 na página 45.
- O arquivo `aqmsetup`, que fornece diversos valores iniciais de configuração para o kernel. Consulte a seção "O arquivo de instalação" na página 66 para obter mais informações.

- O arquivo `aqmconfig.xml`, que configura o kernel. Consulte a seção "O arquivo de configuração" para obter informações adicionais. Esse arquivo contém a maior parte dos valores que configuram o kernel.
- Arquivo `aqmcreateq.bat` (sistemas Windows) ou `aqmcreateq.sh` (UNIX e OS/400), que é um script que cria as filas do MQSeries. Consulte a seção "Criação de filas do MQSeries" na página 98.

Todos esses arquivos incluem comentários que podem ajudá-lo a editá-los.

É recomendável que você faça um backup desses arquivos. Para obter informações adicionais, consulte a seção "Manutenção do kernel" na página 95.

O arquivo de instalação

O arquivo de instalação, `aqmsetup`, controla várias das configurações iniciais do kernel, incluindo:

- Localização do arquivo de configuração. Consulte a seção "O arquivo de configuração".
- Localização dos DTDs XML, se não estiverem no diretório atual.
- As variáveis de ambiente do Java Native Interface (JNI) para a interface C, para alterar a quantidade de memória utilizada. Aplica-se quando um módulo executável C inicia um processo e uma máquina virtual Java é instanciada por esse processo. A memória utilizada pode ser controlada, neste caso, pela remoção de comentários e modificações nas seguintes linhas no arquivo `aqmsetup` file:

  ```
  #AQM_JNI_NATIVESTACKSIZE=1048576
  #AQM_JNI_JAVASTACKSIZE=4194304
  #AQM_JNI_MINHEAPSIZE=16777216
  #AQM_JNI_MAXHEAPSIZE=268435426
  ```

  Todos os tamanhos estão em bytes.

Um exemplo do arquivo `aqmsetup` é fornecido no "Apêndice E. Amostra do arquivo de configuração" na página 125 e também está incluído no diretório `samples` da instalação do MQSeries Adapter Kernel.

Se necessário, edite o arquivo de instalação quando o MQSeries Adapter Kernel for instalado pela primeira vez. Após a instalação, edite o arquivo apenas se o kernel encontrar um problema de falta de memória Java, conforme discutido na lista anterior.

O arquivo de configuração

Esta seção trata do arquivo `aqmconfig.xml`, que determina a configuração do kernel. "Sintaxe e organização do arquivo de configuração" na página 67 fornece as informações da estrutura do arquivo de configuração. A seção

"Edição do arquivo de configuração" na página 87 fornece grandes sugestões práticas para a edição do arquivo de configuração.

A configuração do MQSeries Adapter Kernel é determinada por um arquivo XML denominado `aqmconfig.xml`. Um exemplo do arquivo de configuração está incluído no "Apêndice D. Amostra do arquivo de configuração" na página 119 e também no diretório `samples` da instalação do MQSeries Adapter Kernel.

Os valores especificados no arquivo de configuração controlam os seguintes elementos do kernel:

- Identificadores lógicos de origem
- Identificadores lógicos de destino
- Daemons do adaptador e as informações do operador em um lado de destino
- Clientes de rastreamento
- Servidores de rastreamento
- Enfileiramento e roteamento de mensagens, determinados pelas seguintes especificações:
  - Os nomes das filas de recepção, filas de erro e filas de resposta
  - Um ou mais destinos padrão para onde as mensagens são envidas
  - O nome do gerenciador de fila do MQSeries ou a produção de conexão da fila JMS que obtém ou envia a mensagem
  - O tempo limite para as mensagens recebidas ou respostas
  - A classe do adaptador de destino no lado de destino do kernel que processa cada mensagem
  - As informações adicionais específicas para o adaptador de destino
  - O número mínimo de operadores em um lado de destino (se executando em MQSeries Adapter Kernel independente)
  - Ativação e desativação do rastreamento e controle do nível de rastreamento
  - Ativação e desativação do registro de auditoria
- Modo de comunicação

Sintaxe e organização do arquivo de configuração

Como a configuração do MQSeries Adapter Kernel tem como base o Lightweight Directory Access Protocol (LDAP), a estrutura do arquivo de configuração espelha o LDAP. O elemento XML de nível superior, `Epic`, representa o nível superior do diretório LDAP e os objetos do LDAP subordinados são representados por elementos XML aninhados no elemento de nível superior. Alguns dos elementos XML exigem atributos que representam as informações de LDAP. Os valores são incluídos na

configuração como o conteúdo dos elementos ou como atributos dos elementos. Um exemplo de valor de configuração atribuído como o conteúdo de um elemento é o `<epictracelevel>-1</epictracelevel>`, que atribui o valor `-1` (todas as possíveis mensagens) ao elemento `epictracelevel`. Um exemplo de valor de configuração atribuído como um atributo de um elemento é o `<ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">`, que atribui a classe `com.ibm.logging.ConsoleHandler` a ser utilizada como uma rotina de tratamento de rastreamento.

A seguir está uma lista com a descrição dos elementos de alto nível utilizados no arquivo de configuração. A seção "Elementos XML utilizados no arquivo de configuração" na página 70 lista e descreve o conjunto completo de elementos utilizados no arquivo de configuração. Consulte o exemplo do arquivo de configuração para obter exemplos de como os diferentes elementos são utilizados no contexto.

- `Epic`: Elemento de nível superior exigido para o arquivo `aqmconfig.xml`.
- `ePICApplications`: Filho exigido do elemento `Epic`.
- `ePICApplication`: Filho exigido do elemento `ePICApplications`. Lista e define os aplicativos a serem atendidos pelo kernel; um elemento `ePICApplication` completamente definido (incluindo elementos filho) é exigido para cada aplicativo.
- `AdapterRouting`: Um filho opcional do elemento `ePICApplication`. Define o gerenciador de filas e informações relacionadas.
- `ePICBodyCategory`: O filho exigido do elemento `AdapterRouting`. Define a categoria do corpo para mensagens a serem roteadas pelo kernel.
- `ePICBodyType`: O filho exigido do elemento `ePICBodyCategory`. Define o tipo de corpo das mensagens a serem roteadas pelo kernel. Contém as informações para os itens, como destinos de mensagens, modos de comunicação para as mensagens recebidas e formatadores de mensagem.
- `ePICAdapterDaemonExtensions`—Um filho opcional do elemento `ePICApplication` representando um daemon do adaptador. Contém informações relacionadas aos daemons dos adaptadores, incluindo identificadores de aplicativo e número de operadores do adaptador.
- `ePICTraceExtensions`: Um filho opcional do elemento `ePICApplication` representando um aplicativo cliente de rastreamento ou elemento servidor de rastreamento. Define as informações relacionadas ao rastreamento.

A Figura 8 na página 69 mostra a estrutura de alto nível do arquivo de configuração. Esse não é um exemplo prático de um arquivo de configuração; simplesmente demonstra os relacionamentos e dependências entre os elementos de nível superior. Consulte o "Apêndice D. Amostra do arquivo de configuração" na página 119 para obter um exemplo completo.

```
<?xml version="1.0" encoding="UTF-8"?>
   <Epic o="ePIC">
      <ePICApplications o="ePICApplications">
    <!-- The following <ePICApplication> tag configures the kernel to work with
    an application named APP1. -->
    <ePICApplication epicappid="APP1">
      <!-- Tags here specify logging and trace information for the APP1
      application. -->
      <AdapterRouting cn="epicadapterrouting">
        <!-- Tags here specify the queue manager and its attributes. -->
        <ePICBodyCategory epicbodycategory="DEFAULT">
          <ePICBodyType epicbodytype="DEFAULT">
            <!-- Tags here specify the details of transporting and processing
             messages from APP1. -->
          </epicBodyType>
        </ePICBodyCategory>
      </AdapterRouting>
    </ePICApplication>
    <!-- The following <ePICApplication> tag starts an adapter daemon for the
    APP1 application. -->
    <ePICApplication epicappid="APP1Daemon">
      <!-- Specifications for the APP1Daemon adapter daemon, which works with
      the APP1 application. -->
      <ePICAdapterDaemonExtensions cn="epicappextensions">
        <epicdepappid>APP1</epicdepappid>
        <epicminworkers>1</epicminworkers>
      </ePICAdapterDaemonExtensions>
    </ePICApplication>
    <!-- The following <ePICApplication> tag configures the kernel to work with
    an application named APP2. -->
    <ePICApplication epicappid="APP2">
      <!-- Tags here specify logging and trace information for the APP2
      application. -->
      <AdapterRouting cn="epicadapterrouting">
        <!-- Tags here specify the queue manager and its attributes. -->
        <ePICBodyCategory epicbodycategory="DEFAULT">
          <ePICBodyType epicbodytype="DEFAULT">
            <!-- Tags here specify the details of transporting and processing
             messages from APP2. -->
          </epicBodyType>
        </ePICBodyCategory>
      </AdapterRouting>
    </ePICApplication>
    <!-- The following <ePICApplication> tag configures a trace client named
    TraceClient. -->
    <ePICApplication epicappid="TraceClient">
      <ePICTraceExtensions cn="epicappextensions">
        <!-- Tags here specify attributes of the trace client. -->
      </ePICTraceExtensions>
    </ePICApplication>
  </ePICApplications>
</Epic>
```

Figura 8. Estrutura de alto nível do arquivo de configuração

A seguir está uma lista com a descrição do conjunto completo de elementos utilizados no arquivo de configuração. Se um elemento for observado como tendo um valor padrão, o kernel utilizará esse valor se um elemento da configuração exigir um valor não especificado de forma explícita.

Elementos XML utilizados no arquivo de configuração

Epic Elemento de nível superior para a configuração

Elementos-filho:

- `contexto`
- `ePICApplications` (obrigatório)

Atributos: `o="ePIC"` (obrigatório)

contexto

Especifica a raiz do contexto do sistema de arquivo Java Naming and Directory Interface (JNDI) (FSContext) quando os objetos Java Message Service (JMS) são utilizados. O padrão é o diretório atual. Exigido se JMS for utilizado. Consulte a seção "Utilização do armazenamento do objeto JMS" na página 106 para obter mais informações sobre a utilização dos objetos JMS com o MQSeries Adapter Kernel.

Elementos-filho: Nenhum

Atributos: Nenhum

ePICApplications

Contém informações sobre os aplicativos atendidos pelo kernel.

Elementos-filho: `ePICApplication` (obrigatório)

Atributos: `o="ePICApplications"` (obrigatório)

ePICApplication

Especifica as informações sobre um aplicativo atendido pelo kernel.

Elementos-filho:

- `epiclogging`
- `epictrace`
- `epictracelevel`
- `epictraceclientid`
- `epiclogoninfoclassname`
- `AdapterRouting`
- `ePICTraceExtensions`
- `ePICAdapterDaemonExtensions`

Atributos: epicappid="*application_ID*", em que *application_ID* é um identificador de aplicativo válido (obrigatório)

epiclogging
Determina se desempenha o registro de auditoria. O registro de auditoria requer o produto WebSphere Business Integrator. O padrão é false.

Elementos-filho: Nenhum

Atributos: Nenhum

epictrace
Determina se deve ser utilizado o rastreamento. O padrão é false.

Elementos-filho: Nenhum

Atributos: Nenhum

epictracelevel
Define o nível de rastreamento, utilizando as constantes especificadas pela classe com.ibm.logging.IRecordType. O padrão é 0 (sem mensagens). Consulte o *Manual de Determinação de Problemas* para obter mais detalhes sobre o rastreamento para uma lista completa de níveis de rastreamento válido.

Elementos-filho: Nenhum

Atributos: Nenhum

epictraceclientid
Especifica o nome do aplicativo cliente de rastreamento. O padrão é TraceClient.

Elementos-filho: Nenhum

Atributos: Nenhum

epiclogoninfoclassname
Especifica o nome da classe de início de sessão utilizado para o aplicativo ao utilizar um adaptador do EAB. O padrão é com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault.

Elementos-filho: Nenhum

Atributos: Nenhum

AdapterRouting
Contém informações sobre os tipos de mensagens e o roteamento de mensagens.

Elementos-filho:

- epicmqppqueuemgr
- epicuseremotequeuemanagertosend

- epicmqppqueuemgrhostname
- epicmqppqueuemgrportnumber
- epicmqppqueuemgrchannelname
- epicjmsconnectionfactoryname
- ePICBodyCategory (obrigatório)

Atributos: cn="epicadapterrouting" (obrigatório)

epicmqppqueuemgr
Se o MQSeries estiver sendo utilizado como o mecanismo de transporte, ele especificará o nome do gerenciador de filas a ser utilizado. Se não for especificado ou se especificado como DEFAULT, o gerenciador de filas padrão será utilizado.

Elementos-filho: Nenhum

Atributos: Nenhum

epicuseremotequeuemanagertosend
Se o MQSeries estiver sendo utilizado como mecanismo de transporte, especifique se irá utilizar em um gerenciador de fila remoto para enviar as mensagens. O padrão é false.

Elementos-filho: Nenhum

Atributos: Nenhum

epicmqppqueuemgrhostname
Se o MQSeries estiver utilizando como mecanismo de transporte, especifique o nome do host TCP/IP da máquina em que o gerenciador de fila se localiza. Exigido apenas se o Cliente MQSeries estiver sendo utilizado.

Elementos-filho: Nenhum

Atributos: Nenhum

epicmqppqueuemgrportnumber
Se o MQSeries estiver sendo utilizado como mecanismo de transporte, especifique o número da porta do processo do servidor do gerenciador de fila. O padrão é 1414 (padrão do MQSeries). Exigido apenas se o Cliente MQSeries estiver sendo utilizado.

Elementos-filho: Nenhum

Atributos: Nenhum

epicmqppqueuemgrchannelname
Se o MQSeries estiver sendo utilizado como o mecanismo de

transporte, ele especificará o nome do canal do servidor do gerenciador de filas. Exigido apenas se o Cliente MQSeries estiver sendo utilizado.

Elementos-filho: Nenhum

Atributos: Nenhum

epicjmsconnectionfactoryname

Se o JMS estiver sendo utilizado como o mecanismo de transporte, ele especificará o nome de fábrica da Conexão JMS. O valor deve ser especificado como *attribute=object*, em que *attribute* é o atributo do LDAP e *object* é o objeto de Conexão JMS. Espera-se que o objeto seja armazenado no elemento `AdapterRouting`. Por exemplo, para um objeto de conexão JMS denominado `QCFTEST1` com um atributo do LDAP `cn`, o valor especificado por esse elemento será `cn=QCFTEST1`.

Elementos-filho: Nenhum

Atributos: Nenhum

ePICBodyCategory

Especifica a categoria do corpo de mensagens que estão sendo enviadas.

Elementos-filho: `ePICBodyType` (obrigatório)

Atributos: `epicbodycategory=`*`body_category`*, em que *body_category* especifica a categoria do corpo de mensagens que estão sendo envidas (obrigatório)

ePICBodyType

Especifica o tipo de corpo de mensagens que estão sendo enviadas.

Elementos-filho:

- `epiccommandclassname`
- `epiccommandtype`
- `epiccommandejbmapper`
- `epiccommandejbmethod`
- `epiccommandejbmethodparmtype`
- `epiccommandejburl`
- `epiccommandejbinitialcontext`
- `epicdestids`
- `epicreceivemode`
- `epicmessageformatter`
- `epicreceivetimeout`
- `epicreceivemqppqueue`
- `epicerrormqppqueue`

- epicreplymqppqueue
- epicjmsreceivequeuename
- epicjmserrorqueuename
- epicjmsreplyqueuename
- epicreceivefiledir
- epiccommitfiledir
- epicerrorfiledir

Atributos: epicbodytype=*body_type*, em que *body_type* especifica o tipo de corpo das mensagens que estão sendo enviadas (obrigatório)

epiccommandclassname
Especifica o nome de um adaptador de destino EAB ou o comando EJB que é chamado para processar as mensagens. Exigido se um daemon do adaptador ou WebSphere Application Server estiver sendo utilizado para as mensagens recebidas.

Elementos-filho: Nenhum

Atributos: Nenhum

epiccommandtype
Especifica o tipo de adaptador de destino. Os valores possíveis incluem MQAKEAB e MQAKEJB. MQAKEAB especifica um adaptador de destino EAB do MQSeries Adapter Kernel padrão; MQAKEJB especifica se os beans corporativos são utilizados em um lado de destino do kernel no WebSphere Application Server. O padrão é MQAKEAB. Um valor MQAKEJB é requerido quando o adaptador de destino é um bean corporativo.

Elementos-filho: Nenhum

Atributos: Nenhum

epiccommandejbmapper
Especifica o nome da classe TDCMapper utilizado para mapear os dados de entrada. O padrão é TDCGenericMapper. Exigido quando o adaptador de destino é um bean corporativo.

Elementos-filho: Nenhum

Atributos: Nenhum

epiccommandejbmethod
Especifica o nome de um método para chamar em um bean corporativo. O método deve aceitar um objeto TerminalDataContainer como entrada e retorna um objeto TerminalDataContainer. O padrão é execute. Exigido quando o adaptador de destino é um bean corporativo.

Elementos-filho: Nenhum

Atributos: Nenhum

epiccommandejbmethodparmtype
Especifica o nome da classe do objeto que está sendo utilizado como parâmetro para o método sendo chamado em um bean corporativo. O padrão é o nome da classe do objeto retornado pelo `TDCMapper`. Exigido quando o adaptador de destino for um bean corporativo.

Elementos-filho: Nenhum

Atributos: Nenhum

epiccommandejburl
Especifica um uniform resource locator (URL) de um bean corporativo disponibilizado, em um formato `IIOP://`*`host_name`*`:`*`port`*, em que *host_name* é o nome do host do servidor EJB e *port* é a porta em que o servidor de nomes é listado (por padrão, `900`). O padrão é `IIOP:///`. Exigido quando o adaptador de destino é um bean corporativo.

Elementos-filho: Nenhum

Atributos: Nenhum

epiccommandejbinitialcontext
Especifica o nome da produção do contexto inicial para pesquisar o nome do host do bean corporativo. O padrão é `com.ibm.ejs.ns.jndi.CNInitialContextFactory`. Exigido quando o adaptador de destino é um bean corporativo.

Elementos-filho: Nenhum

Atributos: Nenhum

epicdestids
Especifica os identificadores de um ou mais aplicativos a serem utilizados como destinos de mensagens. Exigido se o aplicativo estiver enviando mensagens e o ID lógico do destino estiver definido como `NONE`.

Elementos-filho: Nenhum

Atributos: Nenhum

epicreceivemode
Especifica o modo de comunicação a ser utilizado. Consulte o "Apêndice A. Modos de Comunicação" na página 103 para uma lista e uma explicação dos modos de comunicação válida. Exigido se o aplicativo estiver recebendo mensagens.

Elementos-filho: Nenhum

Atributos: Nenhum

epicmessageformatter
Especifica o formatador de mensagem a ser utilizado, dependente do valor de `epicreceivemode` e do método de transporte utilizado. Consulte a Tabela 10 na página 105 e a Tabela 11 na página 105 para obter detalhes sobre os formatadores de mensagens e métodos de transporte.

Elementos-filho: Nenhum

Atributos: Nenhum

epicreceivetimeout
Especifica, em milissegundos, o período de tempo que o receptor aguarda as mensagens antes do término do tempo limite. O padrão é `0`. Um valor `-1` especifica sem tempo limite (aguarda indefinidamente).

Elementos-filho: Nenhum

Atributos: Nenhum

epicreceivemqppqueue
Especifica o nome da fila da qual as mensagens devem ser recebidas. Exigido quando o elemento `epicreceivemode` especifica um modo de comunicação MQSeries. Consulte "Apêndice A. Modos de Comunicação" na página 103 para uma lista de modos de comunicação MQSeries.

Elementos-filho: Nenhum

Atributos: Nenhum

epicerrormqppqueue
Especifica o nome da fila na qual devem ser colocadas as mensagens de erro. Exigido se a fila de mensagem de erro estiver sendo utilizado e o elemento `epicreceivemode` especifica um modo de comunicação MQSeries. Consulte o "Apêndice A. Modos de Comunicação" na página 103 para uma lista de modos de comunicação MQSeries.

Elementos-filho: Nenhum

Atributos: Nenhum

epicreplymqppqueue
Especifica o nome da fila da qual as mensagens de resposta devem ser recebidas. Exigido se os pedidos de resposta estiverem sendo utilizados e o elemento `epicreceivemode` especifica um modo de comunicação MQSeries. Consulte o "Apêndice A. Modos de Comunicação" na página 103 para uma lista de modos de comunicação MQSeries.

Elementos-filho: Nenhum

Atributos: Nenhum

epicjmsreceivequeuename
Especifica o nome da fila da qual as mensagens devem ser recebidas. Exigido para o modo de comunicação JMS. Espera-se que o objeto seja armazenado no elemento `ePICBodyType`. O valor deve ser especificado como *`attribute=object`*, em que *attribute* é o atributo do LDAP e *object* é o nome do objeto da fila JMS. Por exemplo, para um objeto JMS denominado `TEST1AIQ` com um atributo do LDAP `cn`, o valor especificado por esse elemento será `cn=TEST1AIQ`.

Elementos-filho: Nenhum

Atributos: Nenhum

epicjmserrorqueuename
Especifica o nome da fila na qual devem ser colocadas as mensagens de erro. Exigido se a fila de mensagem de erro estiver sendo utilizanda com o modo de comunicação JMS. Espera-se que o objeto seja armazenado no elemento `ePICBodyType`. O valor deve ser especificado como *`attribute=object`*, em que *attribute* é o atributo do LDAP e *object* é o nome do objeto da fila JMS. Por exemplo, para um objeto JMS denominado `TEST1AEQ` com um atributo do LDAP `cn`, o valor especificado por esse elemento será `cn=TEST1AEQ` .

Elementos-filho: Nenhum

Atributos: Nenhum

epicjmsreplyqueuename
Especifica o nome da fila da qual as mensagens de resposta devem ser recebidas. Exigido se os pedidos de resposta estiverem sendo utilizados com o modo de comunicação JMS. Espera-se que o objeto seja armazenado no elemento `ePICBodyType`. O valor deve ser especificado como *`attribute=object`*, em que *attribute* é o atributo do LDAP e *object* é o nome do objeto da fila JMS. Por exemplo, para um objeto JMS denominado `TEST1RPL` com um atributo do LDAP `cn`, o valor especificado por esse elemento será `cn=TEST1RPL` .

Elementos-filho: Nenhum

Atributos: Nenhum

epicreceivefiledir
Especifica o nome do diretório do qual as mensagens devem ser recebidas. Exigido para o modo de comunicação FILE.

Elementos-filho: Nenhum

Atributos: Nenhum

epiccommitfiledir
Especifica o nome do diretório no qual mantêm as mensagens recebidas até serem consolidadas. Exigido para o modo de comunicação FILE quando as mensagens estão sendo recebidas.

Elementos-filho: Nenhum

Atributos: Nenhum

epicerrorfiledir
Especifica o nome do diretório no qual coloca as mensagens de erro. Exigido se a fila de mensagem de erro está sendo utilizado com o modo de comunicação FILE.

Elementos-filho: Nenhum

Atributos: Nenhum

ePICAdapterDaemonExtensions
Contém informações sobre as extensões do daemon do adaptador.

Elementos-filho:

- `epicdepappid`
- `epicminworkers`

Atributos: `cn="epicappextensions"` (obrigatório)

ePICTraceExtensions
Contém informações sobre as extensões de rastreamento. Consulte a publicação *Manual de Determinação de Problemas*, para obter informações completas sobre esse elemento e seus filhos.

Elementos-filho:

- `epicdepappid`
- `epictracesyncoperation`
- `epictracemessagefile`
- `epictracehandler`
- `ePICTraceHandler`

Atributos: `cn="epicappextensions"` (obrigatório)

epicdepappid
Especifica o identificador do aplicativo que o daemon do adaptador está atendendo. Se o padrão para o ID do aplicativo com o qual o daemon do adaptador foi inicializado.

Elementos-filho: Nenhum

Atributos: Nenhum

epicminworkers

Especifica o número de operadores do adaptador iniciados pelo daemon do adaptador. O padrão é 1.

Elementos-filho: Nenhum

Atributos: Nenhum

Configurações comuns

Esta seção lista os valores da configuração para os vários cenários da configuração comuns, incluindo os valores para enviar e receber as mensagens utilizando os vários transportes de comunicação. Quando uma mensagem é enviada, os valores da configuração são obtidos a partir do lado da origem e do destino; quando uma mensagem é recebida, os valores da configuração são obtidos apenas a partir do lado de destino. A origem e o destino são representados pelos seus identificadores respectivamente lógico. Esses exemplos assumem que a origem e o destino estão em duas máquinas de diferente. Se o identificador do aplicativo de destino não estiver ainda definido, ele será determinado a partir de um valor do elemento `epicdestids` da configuração de origem.

Nota: Os cenários da configuração lista os valores da configuração do elemento que são possíveis de aplicativo e podem ser definidos. Consulte a listagem dos elementos no "Elementos XML utilizados no arquivo de configuração" na página 70 para qualquer padrão que se aplica para o elemento.

Configurações comuns do MQSeries: Esta seção fornece as configurações comuns quando o MQSeries é utilizado como transporte de comunicação. O elemento `epicreceivemode` especifica um modo de comunicação do MQSeries (por exemplo, MQPP ou MQRFH2). Os cenários são listados a seguir:

- Tabela 2 na página 80 mostra os elementos da configuração que precisam ser definidos ao enviar uma mensagem a partir de um servidor MQSeries para um servidor MQSeries.
- Tabela 3 na página 80 mostra os elementos da configuração que precisam ser definidos ao enviar uma mensagem a partir de um servidor MQSeries que está utilizando um gerenciador de fila remoto para um servidor MQSeries.
- Tabela 4 na página 81 mostra os elementos da configuração que precisam ser definidos ao enviar uma mensagem a partir de um cliente MQSeries que está utilizando um servidor host para um servidor MQSeries.
- Tabela 5 na página 82 mostra os elementos de configuração que precisam ser definidos ao receber uma mensagem em um servidor MQSeries.
- Tabela 6 na página 83 mostra os elementos de configuração que precisam ser definidos ao receber uma mensagem em um cliente MQSeries que está utilizando um servidor host.

Tabela 2. Configuração comum: Envia uma mensagem a partir de um servidor MQSeries para outro servidor MQSeries

Configuração da origem	Configuração de destino
	O elemento `epicreceivemode` especifica um modo de configuração MQSeries.
O elemento `epicmqppqueuemgr` especifica o nome do gerenciador de fila. Este gerenciador de fila deve existir em uma máquina do aplicativo.	
	O elemento `epicreceivemqppqueue` especifica o nome da fila recebida. Esta fila deve ser uma fila MQSeries remota da máquina de aplicativo de destino ou parte de um grupo de MQSeries.
O elemento `epicreplymqppqueue` especifica o nome da fila de resposta. Esta fila deve ser uma fila local MQSeries da máquina emissora ou parte de um grupo MQSeries. Utilizado apenas para os pedidos e respostas síncrona.	
	O elemento `epicmessageformatter` especifica o nome do formatador para a utilização.
O elemento `epicreceivetimeout` especifica o tempo em que receptor espera para responder antes do tempo expirar.	

Tabela 3. Configuração comum: Envia uma mensagem a partir de um servidor MQSeries para um servidor MQSeries via um gerenciador de fila remoto

Configuração da origem	Configuração de destino
	O elemento `epicreceivemode` especifica um modo de configuração MQSeries.
O elemento `epicmqppqueuemgr` especifica o nome do gerenciador de fila. Este gerenciador de fila deve existir em uma máquina do aplicativo.	O elemento `epicmqppqueuemgr` especifica o nome do gerenciador de fila. Este gerenciador de fila deve existir em uma máquina do aplicativo de destino. O nome deve ser especificado; um valor padrão não pode ser utilizado.
O elemento `epicremotequeuemanagertosend` especifica que um gerenciador da fila remoto está sendo utilizado para enviar as mensagens.	

Tabela 3. Configuração comum: Envia uma mensagem a partir de um servidor MQSeries para um servidor MQSeries via um gerenciador de fila remoto (continuação)

Configuração da origem	Configuração de destino
	O elemento `epicreceivemqppqueue` especifica o nome da fila recebida. Esta fila deve ser uma fila local MQSeries da máquina do aplicativo de destino ou parte de um grupo MQSeries.
O elemento `epicreplymqppqueue` especifica o nome da fila de resposta. Esta fila deve ser uma fila local MQSeries da máquina emissora ou parte de um grupo MQSeries. Utilizado apenas para os pedidos e respostas síncrona.	
	O elemento `epicmessageformatter` especifica o nome do formatador para a utilização.
O elemento `epicreceivetimeout` especifica o tempo em que receptor espera para responder antes do tempo expirar.	

Tabela 4. Configuração comum: Envia uma mensagem a partir de um cliente MQSeries que está utilizando um servidor host para um servidor MQSeries

Configuração da origem	Configuração de destino
	O elemento `epicreceivemode` especifica um modo de configuração MQSeries.
O elemento `epicmqppqueuemgr` especifica o nome do gerenciador de fila. Este gerenciador da fila deve existir em uma máquina host do cliente.	
O elemento `epicmqppqueuemgrhostname` especifica o nome do host da máquina do servidor MQSeries.	
O elemento `epicmqppqueuemgrportnumber` especifica o número da porta do processo do servidor do gerenciador de fila na máquina do servidor.	
O elemento `epicmqppqueuemgrchannelnumber` especifica o número do canal do servidor de gerenciador da fila.	

Tabela 4. Configuração comum: Envia uma mensagem a partir de um cliente MQSeries que está utilizando um servidor host para um servidor MQSeries (continuação)

Configuração da origem	Configuração de destino
	O elemento `epicreceivemqppqueue` especifica o nome da fila recebida. Esta fila deve ser uma fila MQSeries remota da máquina de aplicativo de destino ou parte de um grupo MQSeries.
O elemento `epicreplymqppqueue` especifica o nome da fila de resposta. Esta fila deve ser uma fila local MQSeries da máquina host do cliente emissor ou parte de um grupo MQSeries. Utilizado apenas para os pedidos e respostas síncronos.	
	O elemento `epicmessageformatter` especifica o nome do formatador para a utilização.
O elemento `epicreceivetimeout` especifica o tempo em que o receptor espera para responder antes do tempo expirar.	

Tabela 5. Configuração comum: O servidor MQSeries recebe uma mensagem

Configuração da origem	Configuração de destino
Não é aplicável.	O elemento `epicreceivemode` especifica um modo de comunicação MQSeries.
	O elemento `epicmqppqueuemgr` especifica o nome do gerenciador de fila. Este gerenciador de fila deve existir da máquina do aplicativo de destino.
	O elemento `epicreceivemqppqueue` especifica o nome da fila recebida. Esta fila deve ser uma fila local MQSeries da máquina de destino.
	O elemento `epicerrormqppqueue` especifica o nome da fila de erro. Esta fila deve ser uma fila local MQSeries da máquina de destino ou parte de um grupo. Exigido apenas se utilizar um operador de adaptador.
	O elemento `epicmessageformatter` especifica o nome do formatador para a utilização.

Tabela 5. Configuração comum: O servidor MQSeries recebe uma mensagem (continuação)

Configuração da origem	Configuração de destino
	O elemento `epicreceivetimeout` especifica o tempo em que o receptor espera para receber uma mensagem antes do tempo expirar.

Tabela 6. Configuração comum: O cliente MQSeries que está utilizando o servidor host recebe uma mensagem

Configuração da origem	Configuração de destino
Não é aplicável.	O elemento `epicreceivemode` especifica um modo de configuração MQSeries.
	O elemento `epicmqppqueuemgr` especifica o nome do gerenciador de fila. Este gerenciador de fila deve existir em uma máquina host do cliente receptor.
	O elemento `epicmqppqueuemgrhostname` especifica o nome do host da máquina do servidor MQSeries.
	O elemento `epicmqppqueuemgrportnumber` especifica o número da porta do processo do servidor do processo da fila em uma máquina do servidor.
	O elemento `epicmqppqueuemgrchannelnumber` especifica o número do canal do servidor do gerenciador de fila.
	O elemento `epicreceivemqppqueue` especifica o nome da fila recebida. Esta fila deve ser uma fila local MQSeries da máquina host do cliente receptor.
	O elemento `epicerrormqppqueue` especifica o nome da fila de erro. Esta fila deve ser um fila local MQSeries da máquina host do cliente receptor ou parte de um grupo. Exigido apenas se for utilizar um operador de adaptador.
	O elemento `epicmessageformatter` especifica o nome do formatador para a utilização.

Tabela 6. Configuração comum: O cliente MQSeries que está utilizando o servidor host recebe uma mensagem (continuação)

Configuração da origem	Configuração de destino
	O elemento `epicreceivetimeout` especifica o tempo em que o receptor espera para receber uma mensagem antes do tempo expirar.

Configurações comuns JMS: Esta seção fornece as configurações comuns quando o JMS é utilizado como transporte de comunicação. O elemento `epicreceivemode` especifica o JMS.

Se a implementação do JMS do MQSeries estiver sendo utilizada, os objetos MQSeries apropriados devem ser criados. Por exemplo, a produção da conexão da fila JMS deve ser relacionada para um gerenciador de fila em um servidor MQSeries e, uma fila JMS deve ser relacionada para uma fila MQSeries. Os objetos MQSeries não precisam ser listados na configuração, mas os objetos MQSeries suportados devem ser criados.

Os cenários a seguir são listados:

- A Tabela 7 mostra os elementos de configuração que precisam ser definidos quando enviam uma mensagem via JMS.
- A Tabela 8 na página 85 mostra os elementos de configuração que precisam ser definidos quando recebem um mensagem via JMS.

Tabela 7. Configuração comum: Envia uma mensagem via JMS

Configuração da origem	Configuração de destino
	O elemento `epicreceivemode` especifica o modo de comunicação JMS.
O elemento `epicjmsconnectionfactoryname` especifica o nome da produção da conexão JMS. O objeto mencionado deve existir na configuração.	
	O elemento `epicjmsreceivequeuename` especifica o nome da fila recebida do JMS. O objeto mencionado deve existir na configuração.
O elemento `epicjmsreplyqueuename` especifica o nome da fila de resposta do JMS. O objeto mencionado deve existir na configuração. Utilizado apenas para os pedidos e respostas síncronos.	

Tabela 7. Configuração comum: Envia uma mensagem via JMS (continuação)

Configuração da origem	Configuração de destino
	O elemento `epicmessageformatter` especifica o nome do formatador para a utilização.
O elemento `epicreceivetimeout` especifica o tempo em que o receptor espera por uma resposta antes do tempo ser expirado.	

Tabela 8. Configuração comum: Recebe uma mensagem via JMS

Configuração da origem	Configuração de destino
Não é aplicável.	O elemento `epicreceivemode` especifica o modo de comunicação JMS.
	O elemento `epicjmsconnectionfactoryname` especifica o nome da produção da conexão JMS. O objeto mencionado deve existir na configuração.
	O elemento `epicjmsreceivequeuename` especifica o nome da fila recebida do JMS. O objeto mencionado deve existir na configuração.
	O elemento `epicjmserrorqueuename` especifica o nome da fila de erro do JMS. O objeto mencionado deve existir na configuração.
	O elemento `epicmessageformatter` especifica o nome do formatador para a utilização.
	O elemento `epicreceivetimeout` especifica o tempo em que o receptor espera para receber uma mensagem antes do tempo ser expirado.

Configuração comum do adaptador: Esta seção fornece as configurações comuns quando um adaptador é chamado em um lado de destino. Os valores da configuração diferente são utilizados dependendo se o destino está em adaptador de destino Enterprise Access Builder (especificado por um valor `epiccommandtype` do `MQAKEAB`) ou um adaptador de destino do bean de sessão do serviço EJB (especifica por um valor `epiccommandtype` do `MQAKEJB`).

Nota: Os adaptadores de destino bean da sessão de serviço EJB suportado apenas no WebSphere Application Server.

Para um valor `epiccommandtype` do `MQAKEAB`, especifica os valores para os elementos adicionais a seguir:

- `epiclogoninfoclassname`
- `epiccommandclassname`

Para um valor `epiccommandtype` do `MQAKEJB`, especifica os valores para os elementos adicionais a seguir:

- `epiccommandclassname`
- `epiccommandejbmethod`
- `epiccommandejbmethodparmtype`
- `epiccommandejburl`
- `epiccommandejbinitialcontext`
- `epiccommandejbmapper`

Inclusão de informações do adaptador na configuração

Quando um novo adaptador for incluído na configuração do kernel, várias especificações, no mínimo, deverão ser incluídas no arquivo de configuração. Para um exemplo de um arquivo de configuração mínima, consulte o arquivo `aqmconfig.minimum.xml`. Este arquivo é mostrado no "Apêndice D. Amostra do arquivo de configuração" na página 119 e é incluído também no diretório `samples` da instalação do MQSeries Adapter Kernel.

As seguintes especificações representam a quantidade mínima de informações que devem ser incluídas na configuração quando um novo adaptador for incluído:

- **Adaptador de origem** (envio de mensagens):
  - O identificador do aplicativo no qual o adaptador de origem está sendo executado.
  - O gerenciador de filas padrão. Se o MQSeries estiver sendo utilizado como o mecanismo de transporte, estiver instalado e executando na mesma máquina que o adaptador de origem, você não precisará especificamente configurar o gerenciador de filas.
  - Os identificadores lógicos de destino para mensagens. Se todas as mensagens forem para o mesmo destino, então utilize uma categoria do corpo DEFAULT e um tipo do corpo DEFAULT.
  - Uma fila de recepção para cada identificador lógico de destino à qual o adaptador de origem está enviando mensagens.
- **Adaptador de destino** (recepção de mensagens):

- O identificador do aplicativo no qual o adaptador de destino está sendo executado.
- O gerenciador de filas padrão. Se o MQSeries estiver sendo utilizado como o mecanismo de transporte e estiver instalado e executando na mesma máquina que o adaptador de origem, você não precisará especificamente configurar o gerenciador de filas.
- O modo de recepção para o MQSeries. Geralmente, é a mesma para todas as mensagens; se esse for o caso, utilize uma categoria do corpo DEFAULT e um tipo do corpo DEFAULT.
- A fila de recepção. Se for a mesma para todas as mensagens, utilize uma categoria do corpo DEFAULT e um tipo do corpo DEFAULT.
- A fila de erros, caso ocorra um erro, quando o adaptador de destino processa a mensagem. Geralmente, é a mesma para todas as mensagens; se esse for o caso, utilize uma categoria do corpo DEFAULT e um tipo do corpo DEFAULT.
- O nome da classe do adaptador de destino a ser solicitado quando uma mensagem for recebida. Específico da categoria do corpo e do tipo do corpo.
- O valor do tempo limite de recepção. É recomendado que um valor apropriado seja definido para evitar o uso excessivo da CPU. Geralmente, é a mesma para todas as mensagens; se esse for o caso, utilize uma categoria do corpo DEFAULT e um tipo do corpo DEFAULT.

Para adaptadores de destino adicionais, as mesmas informações podem ser suficientes se a mesma fila de recepção estiver sendo utilizada. Se esse for o caso, as únicas informações que precisam ser especificadas de forma diferente são o nome da classe do adaptador de destino a ser solicitado para a categoria do corpo e tipo do corpo específicos.

- **Especificações de rastreamento**:
  - Se o rastreamento está ativado ou desativado.
  - O nível de rastreamento.
  - Especificações de rastreamento adicionais, incluindo o destino do rastreamento, para adaptadores de origem e adaptadores de destino. Como padrão, o rastreamento é exibido na janela do prompt de comandos ou terminal onde o kernel foi iniciado.

Edição do arquivo de configuração

Utilize um editor de texto ou um editor XML dedicado para editar o arquivo de configuração. Um arquivo DTD denominado `aqmconfig.dtd` é fornecido no diretório `samples` da instalação do kernel para usuários de editores XML. Um editor XML chamado Xeena pode ser descarregado a partir do site Web do IBM alphaWorks em www.alphaworks.ibm.com. As seguintes recomendações se aplicam à edição do arquivo de configuração:

- Antes de começar a editar o arquivo de configuração, reúna todas as informações pertinentes sobre a sua configuração desejada. Elas incluem os nome dos aplicativos e filas que são incluídas na configuração, os tipos de mensagens sendo trocadas, o modo de comunicação ou modos sendo utilizados e as informações sobre os programas de rastreio e outras extensões.
- Copie o arquivo `aqmconfig.xml` de exemplo do diretório `samples` para seu local desejado. Não renomeie a cópia do arquivo de configuração. Edite a cópia.
- Utilize os comentários para identificar diferentes seções do arquivo de configuração e para documentar os valores específicos utilizados em sua configuração (por exemplo, identificadores do aplicativo, nomes de filas de mensagens e valores de tempo limite). Em XML, os comentários começam com os caracteres `<!--` e terminam com os caracteres `-->`. Os comentários podem ocupar múltiplas linhas, como no seguinte exemplo:

  ```
  <!--
       Texto de comentário
  -->
  ```

  Observe que XML não permite comentários dentro de outros comentários.
- Organize o arquivo de configuração de acordo com os identificadores do aplicativo. Mantenha juntas as entradas para cada identificador do aplicativo.
- Se você estiver utilizando um editor XML dedicado, utilize um editor de texto que preserva os finais de linhas e não divide as linhas quando o arquivo é salvo. Exemplos desse tipo de editor de texto são o Bloco de Notas em sistemas Windows e vi ou Emacs em UNIX.
- Lembre-se de que o XML faz distinção entre maiúsculas e minúsculas; tenha muito cuidado ao utilizar o tipo de letra correto para todos os nomes de tags (elemento) e atributos. Utilizar um tipo de letra incorreto na tag poderá invalidar o arquivo de configuração. Utilizar um editor XML dedicado poderá ajudar a evitar erros de tipos de letras.
- Se você deseja utilizar os valores padrão para a categoria do corpo e o tipo de corpo e os valores não são padronizados, deve-se configurar o valor DEFAULT para cada valor do arquivo de configuração. Caso contrário, o kernel não utilizará os valores padrão.
- Valide o arquivo de configuração antes de colocá-lo em produção. Consulte a seção “Validação do arquivo de configuração” na página 89.
- As alterações para o arquivo de configuração têm efeito na próxima vez que, o processo kernel é inicializado. Se um processo estiver sendo executado quando o arquivo de configuração for alterado, o processo deverá ser parado e, em seguida, reiniciado para que as alterações tenham efeito. Tenha muito cuidado se for editar o arquivo de configuração que esteja em produção.

- Faça um backup do arquivo de configuração sempre que editá-lo.

Validação do arquivo de configuração

Depois que o arquivo de configuração for editado e antes de ser colocado em produção, é recomendável validá-lo. Para validar o arquivo de configuração, execute as seguintes etapas gerais:

1. Crie um diretório de validação para o arquivo de configuração dentro do qual será feita a validação e configure o teste.
2. Crie uma mensagem XML de validação.
3. Configure as filas de mensagens para suportar o teste de validação.
4. Configure e, em seguida, execute um teste de validação do arquivo de configuração que envia uma mensagem e recebe outra.
5. Examine os resultados do teste para determinar se o arquivo de configuração está correto.

O utilitário que ajuda a criar uma mensagem XML de validação e o teste de validação do arquivo de configuração são fornecidos como parte do kernel.

O teste de validação do arquivo de configuração solicita o método `sendMsg` e envia uma mensagem XML de validação de um adaptador nativo no lado de origem do kernel para um daemon do adaptador no lado de destino do kernel. Um adaptador de origem e um adaptador de destino não são exigidos. No entanto, se um adaptador de destino estiver instalado, você poderá também testar o envio da mensagem ao aplicativo de destino.

O procedimento é o seguinte.

Nota: Vários scripts são fornecidos como conveniência para utilização no procedimento. Se desejar, copie os scripts e, em seguida, edite as cópias para criar suas próprias versões. Se estiver utilizando o OS/400, observe que as versões UNIX dos scripts podem ser executadas em uma sessão **qsh**. Você pode iniciar uma sessão **qsh** ao inserir o comando Iniciar QSH (**STRQSH**) em um prompt Linguagem de Controle (CL).

Etapa 1. Abra uma janela de prompt de comandos.

Etapa 2. Crie um diretório de validação para o arquivo de configuração. Copie o arquivo de configuração e o arquivo de instalação nele.

Etapa 3. Mude para o diretório de validação.

Etapa 4. Digite o seguinte comando para criar a mensagem XML de validação:

- `aqmcrtmsg.bat` (sistemas Windows)
- `aqmcrtmsg.sh` (UNIX e OS/400)

Etapa 5. Uma lista de opções é exibida. Selecione uma opção e pressione Enter. Digite um valor para cada. A ordem em que os valores são inseridos não é importante. Exemplos de opções são `set sourcelogicalid`, `set msgtype` e `set bodycategory`. Você deve inserir valores para as opções 20, 21, 22 e 23. Você pode utilizar as opções 24 ou 241 para fornecer os dados do corpo da mensagem. Outros valores não são exigidos.

Etapa 6. Insira a opção 1 para criar o arquivo XML de validação. O arquivo XML de validação é criado no diretório atual e é denominado `EpicMessage`*nn*`.xml`, em que *nn* é o número do arquivo XML.

Etapa 7. Insira a opção 0 para sair do utilitário de validação.

Etapa 8. Configure as filas de mensagens apropriadas para suportar a validação.

Etapa 9. Defina a variável de ambiente AQMSETUPFILE para indicar temporariamente o arquivo de configuração no diretório de validação:

- Em um prompt de comandos em sistemas Windows, insira o seguinte:

  ```
  set AQMSETUPFILE=E:\run_time_files\aqmsetup
  ```

  em que `E:\` representa a unidade correta e *run_time_files* está no diretório de validação.
- No UNIX e OS/400, digite o seguinte comando. O comando de exemplo assume que você está utilizando o shell Korn; se você estiver utilizando um shell diferente, altere o comando de acordo.

  ```
  export AQMSETUPFILE=root_directory/run_time_files/aqmsetup
  ```

  em que *root_directory* é o diretório de instalação do kernel e o *run_time_files* está no diretório de validação. No OS/400, o arquivo `aqmsetup` deve sempre no local do seu diretório pessoal IFS (`/home/`*user_name*).

Se necessário, edite o arquivo de instalação no diretório de validação para indicar o arquivo de configuração que está sendo validado.

Etapa 10. Escolha qual dos seguintes deve ser testado:

- Apenas o lado de origem do kernel.
- Se a mensagem puder ser roteada ao aplicativo de destino. Esse teste requer que um adaptador de destino já esteja instalado.
- Rastreamento.

Primeiro teste o lado de origem, em seguida, teste o lado de destino. Desative o daemon do adaptador para testar apenas o

lado de origem. Ative o daemon do adaptador para testar o lado de destino também. Se um adaptador de destino ainda não estiver instalado, você poderá testar se o daemon do adaptador processa a mensagem até o ponto quando tenta solicitar o comando para o adaptador de destino apropriado. É recomendável que você ative o rastreamento, principalmente se um adaptador de destino ainda não estiver instalado.

Etapa 11. Execute o teste de validação. Em qualquer diretório, digite o seguinte comando:

- Em sistemas Windows:

  ```
  aqmsndmsg.bat -a source_logical_identifier -f XML_message_file
  ```

- No UNIX e OS/400:

  ```
  aqmsndmsg.sh -a source_logical_identifier -f XML_message_file
  ```

  em que:

  source_logical_identifier
  indica o identificador lógico de origem. Esse valor deve corresponder ao valor do identificador lógico da origem inserido para opção 20 na Etapa 5 na página 90.

  XML_message_file
  indica o arquivo de mensagens XML.

Nota: Uma lista de todas as opções para esse teste pode ser exibida ao digitar o seguinte comando:

Em sistemas Windows:

```
aqmsndmsg.bat -?
```

No UNIX e OS/400:

```
aqmsndmsg.sh -?
```

Observe que `-?` funciona apenas no shell Korn; se você utilizar outro shell UNIX (como o shell Bourne ou o shell C), utilize o caractere de barra invertida antes do ponto de interrogação (ou seja, `-\?`).

Etapa 12. Analise os resultados. A mensagem de validação contém a categoria do corpo, o tipo do corpo e os dados corretos.

- Se você estiver testando apenas o lado de origem do kernel (ou seja, se o daemon do adaptador não tiver sido iniciado), examine a fila para a qual a mensagem devia ter sido ser roteada.
  - Se o sistema exibir sua mensagem de validação na fila, as entradas no arquivo de configuração serão validadas.

- Se o sistema não exibir sua mensagem de validação na fila, verifique o arquivo de exceções. Se o rastreamento estiver ativado, verifique as mensagens de rastreamento.

- Se você estiver testando o lado de destino do kernel e um adaptador de destino estiver instalado, verifique o aplicativo de destino.
  - Se sua mensagem de validação for recebida pelo aplicativo de destino, as entradas no arquivo de configuração serão validadas.
  - Se sua mensagem de validação não for recebida pelo aplicativo de destino, verifique o arquivo de exceções. Se o rastreamento estiver ativado, verifique as mensagens de rastreamento.
- Se estiver testando o lado de destino do kernel e nenhum adaptador de destino estiver instalado, procure na fila de erros a mensagem de validação e no arquivo de exceções uma mensagem de exceção. Se o rastreamento estiver ativado, verifique as mensagens de rastreamento.
  - Se o sistema exibir sua mensagem de validação na fila de erros e uma mensagem de exceção, as entradas no arquivo de configuração serão validadas.
  - Se o sistema não exibir sua mensagem de validação na fila de erros, verifique o arquivo de exceções. Se o rastreamento estiver ativado, verifique as mensagens de rastreamento.

Etapa 13. Se necessário, modifique o arquivo de configuração e o valide novamente.

Configuração do MQSeries e do MQSeries Integrator

Configure o MQSeries e o software opcional, como o MQSeries Integrator para suporte do kernel como a seguir:

No MQSeries:

- Várias filas são utilizadas para a verificação da instalação. Se você estiver utilizando essas filas para seus ambiente de teste e de produção, limpe-os para verificar a instalação. Consulte a seção “Procedimento de Verificação” na página 49 quanto às filas utilizadas para a verificação da instalação.
- Configure as filas para suportar o transporte de mensagens de acordo com o esquema de roteamento que você designou.
- Ao criar filas, defina a variável de ambiente MAX_QUEUE_DEPTH com o tamanho máximo de fila permitido.

No MQSeries Integrator, configure a entrada e saída das filas em regras (versão 1.1) ou em fluxo de mensagens (versão 2) que correspondam às filas que foram configuradas no arquivo de configuração.

Recomendações sobre desempenho

As seguintes recomendações de desempenho se aplicam especificamente ao MQSeries Adapter Kernel:

- Quando DTDs XML forem analisadas, certifique-se de que os arquivos de DTD residam no mesmo diretório que o processo que as analisa. Isso reduz o esforço exigido pelo processo para localizar as DTDs.
- Quando mensagens extensas estiverem sendo enviadas e recebidas, a utilização do tipo de mensagem RFH2 resultará em um desempenho melhor do que utilizar o tipo mensagem XML.

Consulte a documentação do MQSeries para obter recomendações gerais para melhorar o desempenho.

Início do kernel

Para iniciar o kernel, inicie os seguintes itens:

- Daemon do adaptador para cada aplicativo de destino
- Servidor de rastreamento (opcional)

Observe que se o adaptador de origem estiver sendo executado no processo do aplicativo de origem, o adaptador de origem será automaticamente iniciado com o aplicativo de origem; nenhuma etapa extra será necessária para iniciar o adaptador de origem. Qualquer daemon ou servidor que contenha os adaptadores de origem precisará ser iniciado. Você não inicia os adaptadores de origem diretamente.

Inicie cada daemon do adaptador e servidor de rastreamento executando as seguintes etapas:

Nota: Vários scripts são fornecidos como conveniência para utilização no procedimento. Se desejar, copie os scripts e, em seguida, edite as cópias para criar suas próprias versões. Se estiver utilizando o OS/400, observe que as versões UNIX dos scripts podem ser executadas em uma sessão **qsh**. Você pode iniciar uma sessão **qsh** ao inserir o comando Iniciar QSH (**STRQSH**) em um prompt Linguagem de Controle (CL).

Etapa 1. Inicie o MQSeries ou o software de mensagens e software opcional, como o MQSeries Integrator.

Etapa 2. Inicie qualquer software associado que o seu site local exige—por exemplo, aplicativos (fora do kernel) para ler as mensagens de rastreamento a partir das filas.

Etapa 3. Abra um prompt de comandos. Para cada daemon do adaptador, digite o seguinte comando:

- Em sistemas Windows:

  ```
  aqmstrad.bat -a application_identifier [-bc body_category
  -bt body_type] [-noretry]
  ```

- No UNIX e OS/400:

  ```
  aqmstrad.sh -a application_identifier [-bc body_category
  -bt body_type] [-noretry]
  ```

  em que

  -a *application_identifier*
  Reconhece o identificador lógico de destino que o daemon do adaptador atende.

  -bc *body_category*
  Especifica a categoria do corpo que o operador do daemon do adaptador utiliza para determinar o modo de comunicação, e as informações relacionados para as mensagens recebidas. Se nenhum valor for fornecido, o daemon do adaptador utiliza o valor DEFAULT.

  -bt *body_type*
  Especifica o tipo de corpo que o operador do daemon do adaptador utiliza para determinar o modo de comunicação, e as informações relacionados para as mensagens recebidas. Se nenhum valor for fornecido, o daemon do adaptador utiliza o valor DEFAULT.

  -noretry
  Especifica que o operador pára automaticamente quando não há mais mensagens. Se `-noretry` não for especificado, o operador continuamente verificará se existem mensagens na fila e o daemon do adaptador deverá ser parado manualmente.

  Nota: Se precisar modificar os parâmetros de inicialização Java, edite o arquivo `aqmstrad.bat` (sistemas Windows) ou `aqmstrad.sh` (UNIX e OS/400). Consulte os comentários dentro do arquivo para obter detalhes.

Etapa 4. Para cada servidor de rastreamento, digite o seguinte comando:

- Em sistemas Windows:

  ```
  aqmstrtd.bat -how -a source_application_identifier
  ```

- No UNIX e OS/400:

```
aqmstrtd.sh -how -a source_application_identifier
```

em que:

-how
: Indica como as mensagens de rastreamento devem ser recebidas. Os valores possíveis incluem:
  - `socket`
  - `ena`, ou seja, adaptador nativo

-a *source_application_identifier*
: Identificador do aplicativo de origem. Se nenhum valor for fornecido, o valor padrão `TraceServer` no arquivo de configuração será utilizado.

Consulte o *Manual de Determinação de Problemas* para obter mais informações sobre servidores de rastreamento.

Etapa 5. Depois que um daemon do adaptador ou servidor de rastreamento for iniciado, uma janela de processo permanecerá aberta até você parar o daemon do adaptador. A janela de processo pode exibir exceções. Consulte a seção "Mensagens de exceção" na página 97.

Parada do kernel

Para parar o kernel, pare cada um dos daemons dos adaptadores e servidores de rastreamento. Há vários modos de pará-los:

- Quando você iniciar o daemon do adaptador, defina o parâmetro `-noretry`. Consulte a seção "Início do kernel" na página 93.
- Vá para o prompt de comando (sistemas Windows) ou terminal (UNIX) a partir do qual, o daemon do adaptador ou o servidor de rastreio foi inicializado, e pressione **Ctrl-C**. Execute esta etapa para cada daemon do adaptador ou servidor de rastreamento.
- Em sistemas Windows, você pode utilizar o Gerenciador de tarefas para encerrar o processo.
- No UNIX, você pode utilizar o comando **ps** para determinar o número do processo e, em seguida, utilize o comando **kill** para encerrar o processo.

Manutenção do kernel

Configure um plano de manutenção do kernel. É recomendado que você faça periodicamente o backup dos itens a seguir:

- Configuração, conforme especificada nos seguintes arquivos:
  - `aqmconfig.xml`
  - `aqmsetup`
- Adaptadores criados e seus arquivos associados

O backup ou exclusão periódica do conteúdo do rastreamento e outros arquivos utilizados pelo kernel para suportar seu próprio processamento não é exigido. Faça o backup desses arquivos, se desejar. Se as mensagens de rastreamento estiverem sendo roteadas para um único arquivo em vez de vários arquivos, o arquivo de rastreamento único poderá tornar-se muito grande. Se o nível de rastreamento for definido para capturar um alto nível de detalhes (por exemplo, todas as mensagens de rastreamento ou mensagens informativas), considere a exclusão dos arquivos de rastreamento periodicamente.

Diagnóstico de problemas

Você pode utilizar mensagens de exceção, mensagens de rastreamento e a fila de erros do MQSeries para ajudar a diagnosticar problemas. O MQSeries Adapter Kernel cria mensagens de exceção e, se o rastreamento estiver ativado, rastreia mensagens. Consulte a publicação *Manual de Determinação de Problemas* para obter informações sobre como diagnosticar problemas em um ambiente do MQSeries Adapter Kernel.

Para compreender as mensagens de exceção e as mensagens de rastreamento, você deve compreender como o kernel funciona. O kernel utiliza uma fila de erros para tratar alguns erros. Consulte a seção "Como o kernel funciona" na página 9.

Você pode identificar a mensagem que gerou as mensagens de exceção e as mensagens de rastreamento pela combinação do identificador de mensagem exclusivo e identificador de transação exclusivo.

Não há identificador que permita identificar definitivamente a mesma mensagem na fila de erros e no kernel. No entanto, você pode correlacionar manualmente uma mensagem na fila de erros com a mensagem de exceção, mensagem de rastreamento correspondente ou ambas. Você pode comparar um ou mais dos seguintes:

- Marca de hora aproximada
- Fila para o identificador lógico de origem
- Fila para o identificador lógico de destino
- Categoria do corpo
- Tipo de corpo
- Identificador de mensagem exclusivo
- Identificador de transação exclusivo

Se corresponderem, você provavelmente terá correlacionado a mensagem na fila de erros com a mensagem de exceção ou mensagem de rastreamento correspondente.

Número da versão

Execute `aqmversion.bat` (sistemas Windows) ou `aqmversion.sh` (UNIX e OS/400) no diretório `bin` para exibir o número da versão do kernel.

Mensagens de exceção

O kernel cria os seguintes tipos de mensagens de exceção:

- O adaptador nativo no lado de origem do kernel cria exceções no adaptador de origem. Consulte a documentação do MQSeries Adapter Builder para saber como o adaptador de origem trata essas exceções.
- O adaptador nativo no lado de destino do kernel cria exceções no operador que gerencia o adaptador nativo.
- O operador grava exceções no arquivo `EpicSystemExceptionFile` *`nnnnnnnn`*`.log`, que reside no mesmo diretório que o operador.
- O daemon do adaptador grava mensagens de exceção em um arquivo de exceção denominado `EpicSystemExceptionFile`*`nnnnnnnn`*`.log` que reside no mesmo diretório que o daemon do adaptador. Como o daemon do adaptador e seus operadores residem no mesmo diretório, eles serão gravados no mesmo arquivo de exceção. O daemon do adaptador grava também as mensagens de exceção para o console (ou seja, o prompt do comando ou do terminal que foi utilizado para inicializá-lo, se ele foi inicializado a partir da janela).

As mensagens de exceção do kernel são diferentes das mensagens de exceção do MQSeries. A seguir está um exemplo de uma mensagem de exceção do kernel:

```
2000.10.26 19:38:20.929 com.ibm.epic.adapters.eak.nativeadapter.LMSMQ
Thread Name=main receiveRequest(ENAService)  ePIC TEST2
TYPE_ERROR_EXC AQM5004: Received exception <com.ibm.epic.adapters.eak.common.
AdapterException> Message information: <AQM0114: com.ibm.epic.adapters.eak.
nativeadapter.MQNMRFH2Formatter::convertMessage(MQMessage): Expecting a message
with an MQHRF2 format and received a message with format <MQSTR   >.>
for <unmarshall Message()> having invalid data <(null)>
```

Os valores em uma mensagem de exceção dependem do uso da mensagem, é possível incluir os seguintes itens:

- Marca de hora
- Identificador lógico de origem
- Identificador lógico de destino
- Categoria do corpo
- Tipo de corpo
- Identificador de mensagem exclusivo
- Identificador de transação exclusivo
- Informações de exceção

Consulte a seção “Problemas comuns de verificação” na página 50 para problemas comuns que podem ocorrer durante a verificação da instalação e para possíveis respostas.

Mensagens de rastreamento

O kernel pode ser configurado para gerar mensagens de rastreamento. Para obter informações sobre rastreamento, consulte o *Manual de Determinação de Problemas*.

Utilitários

Criação de filas do MQSeries

Você pode utilizar arquivos batch ou scripts de shell para automatizar a criação de filas do MQSeries. Execute `aqmcreateq.bat` (sistemas Windows) ou `aqmcreateq.sh` (UNIX e OS/400), utilizando o nome do aplicativo como um parâmetro. Esses arquivos criam as seguintes filas para cada aplicativo:

- Fila de recepção denominada *application_name*AIQ.
- Fila de erros, denominada *application_name*AEQ.
- Fila de resposta denominada *application_ name*RPL.

Capítulo 5. Utilização de APIs do MQSeries Adapter Kernel

O kernel inclui APIs utilizadas para funções como enviar e receber mensagens, criar e analisar XML e gerenciar a configuração do kernel. Essas APIs são utilizadas pelos adaptadores criados utilizando o MQSeries Adapter Builder. O Centro de Informações do MQSeries Adapter Kernel inclui a documentação online relacionada à API no formato Javadoc HTML.

O kernel deverá ser utilizado com os adaptadores criados pelo usuário utilizando o MQSeries Adapter Builder. O kernel não deve ser utilizado por chamadas para as APIs de kernel a partir de códigos personalizados independentes. A documentação online da API é fornecida apenas como um auxílio para entender como o kernel funciona e também como um auxílio para os diagnósticos.

A documentação online da API do kernel está localizada no diretório `documentation`.

Capítulo 6. Obtenção de informações adicionais

Existem diversas fontes de informações que podem ser úteis ao utilizar o Pacote do MQSeries Adapter. Para obter informações adicionais sobre o MQSeries Adapter Kernel, consulte o documento *Manual de Determinação de Problemas* disponível no Centro de Informações do MQSeries Adapter Kernel, instalado com o produto. O *Manual de Determinação de Problemas* fornece informações para a solução de problemas específicos que podem surgir com a utilização do kernel. Para obter mais informações sobre o MQSeries Adapter Builder, consulte o Centro de Informações do produto e o sistema de ajuda online.

Disponível na Internet

O site Web da família do produto do MQSeries no endereço www.ibm.com/software/ts/mqseries/. Seguindo os links nesse site Web, você pode:

- Obter as últimas informações sobre a família de produtos MQSeries, incluindo o Pacote do MQSeries Adapter.
- Acessar os manuais do MQSeries nos formatos HTML e PDF, incluindo possivelmente uma edição mais recente deste manual. O link diferente para a página da biblioteca MQSeries no endereço www.ibm.com/software/ts/mqseries/library/manualsa/.
- Fazer download do MQSeries SupportPacs.

Para obter mais informações sobre a utilização do MQSeries no OS/400, consulte a biblioteca do OS/400 no endereço www.ibm.com/servers/eserver/iseries/library/. Consulte também os manuais específicos do OS/400– disponíveis a partir do site Web da biblioteca do MQSeries no endereço www.ibm.com/software/ts/mqseries/library/manualsa/.

Referências

O seguinte material de referência discute os tópicos abordados neste documento:

- O site Web do Open Applications Group no endereço www.openapplications.org/
- Recomendações do Extensible Markup Language (XML) 1.0 W3C no endereço www.w3.org/TR/1998/Rec-xml-19980210

Esses não são sites Web da IBM.

Apêndice A. Modos de Comunicação

Este apêndice fornece informações sobre os modos de comunicação suportados pelo MQSeries Adapter Kernel e as classes Java utilizadas para suportá-los. Alguns dos modos de comunicação são fornecidos como modos de conveniência com os formatadores padrão. Consulte a Tabela 10 na página 105 para obter os formatadores padrão utilizados com os modos de conveniência.

Os modos de comunicação a seguir são suportados:

MQPP O kernel transporta as mensagens utilizando os serviços de base do MQSeries. Este é um modo de conveniência.

MQRFH1 O kernel transporta as mensagens utilizando o MQSeries e serve como intermediária das mensagens utilizando o MQSeries Integrator versão 1.1. Este é um modo de conveniência.

MQRFH2 O kernel transporta as mensagens utilizando o MQSeries e serve como intermediária das mensagens utilizando o MQSeries Integrator versão 2. Este é um modo de conveniência.

MQBD O kernel transporta as mensagens utilizando os serviços de base do MQSeries, mas envia e recebe apenas os dados do corpo. Este é um modo de conveniência. As seguintes características são exclusivas neste modo:

- Ele pode enviar apenas os dados do corpo, não os valores do cabeçalho da mensagem.
- Ele pode receber mensagens que contêm apenas os dados do corpo. Ele utiliza os seguintes valores de cabeçalho de mensagem padrão para as mensagens recebidas:
  - `SourceLogicalApplicationID`—O valor no objeto `ENAService` utilizado na chamada do método de recepção.
  - `BodyCategory`: O valor no objeto `ENAService` utilizado na chamada do método de recepção.
  - `BodyType`: O valor no objeto `ENAService` utilizado na chamada do método de recepção.
  - `Acknowledgment`: Se a MQMessage recebida for um REQUEST do MQSeries, então `Acknowledgment` será definido como 1.

– `BodyData`—Os dados da mensagem recebidas do MQSeries.

Todos os outros valores de cabeçalho utilizam os padrões normais.

MQ O kernel transporta as mensagens utilizando os serviços de base do MQSeries.

JMS O kernel transporta as mensagens utilizando o Java Message Service (JMS). Consulte a seção "Utilização do armazenamento do objeto JMS" na página 106 para obter informações sobre a utilização dos objetos JMS com o MQSeries Adapter Kernel.

FILE O kernel coloca as mensagens em um arquivo e pega-as de um arquivo. Este modo é fornecido apenas para fins de diagnóstico.

A Tabela 9 lista os modos de comunicação e as classes Java que os suportam. Todas as classes Java são do pacote Java `com.ibm.epic.adapters.eak.nativeadapter`. Observe que qualquer classe Java que suporte o serviço lógico de mensagem (LMS - logical message service) pode ser especificado como um modo de comunicação; neste caso, a própria classe é utilizada para suportar a comunicação.

Tabela 9. Modos de comunicação e classes Java suportadas

Modo de comunicação	Classe Java	Notas
MQPP	`LMSMQBindingMQPP`	Requer a instalação do MQSeries
MQRFH1	`LMSMQBindingMQRFH1`	Requer a instalação do MQSeries
MQRFH2	`LMSMQBindingMQRFH2`	Requer a instalação do MQSeries
MQBD	`LMSMQMQBD`	Requer a instalação do MQSeries
MQ	`LMSMQBinding`	Requer a instalação do MQSeries
JMS	`LMSJMS`	Requer a instalação do JMS
FILE	`LMSFile`	Nenhuma

A Tabela 10 na página 105 lista dos modos de comunicação e suas interfaces do formatador associado. A Tabela 11 na página 105 estabelece uma referência cruzada das interfaces do formatador, nomes de classes do formatador e sua utilização. Todos os formatadores são do pacote Java `com.ibm.epic.adapters.eak.nativeadapter`. Observe que cada classe de

formatador pode ser especificada para o modo de comunicação; neste caso, a classe do formatador especificada é utilizada como o formatador.

Tabela 10. Modos de comunicação e interfaces do formatador

Modo de comunicação	Interface do formatador	Formatador padrão
MQPP	MQFormatterInterface	MQNMXMLFormatter
MQRFH1	MQFormatterInterface	MQNMRFH1Formatter
MQRFH2	MQFormatterInterface	MQNMRFH2Formatter
MQBD	MQFormatterInterface	MQNMBDFormatter
MQ	MQFormatterInterface	MQNMXMLFormatter
JMS	JMSFormatterInterface	JMSNMRFH2Formatter
FILE	StringFormatterInterface	NMXMLFormatter

Tabela 11. Interfaces do formatador, nomes de classes do formatador e objetivos

Interface do formatador	Nome da classe do formatador	Objetivo
MQFormatterInterface	MQNMXMLFormatter	EpicMessage como XML
	MQNMRFH1Formatter	EpicMessage como RFH1
	MQNMRFH2Formatter	EpicMessage como RFH2
	MQNMDBFormatter	Somente dados do corpo
JMSFormatterInterface	JMSNMXMLFormatter	EpicMessage como XML
	JMSNMRFH2Formatter	EpicMessage como RFH2
	JMSNMBDFormatter	Somente dados do corpo
StringFormatterInterface	NMXMLFormatter	EpicMessage como XML

A Tabela 12 lista as classes de LMS suportadas e seu grau de suporte transacional. Consulte a seção "Recursos transacionais" na página 28 para obter mais informações sobre a utilização das transações com o MQSeries Adapter Kernel.

Tabela 12. Classes do LMS e suporte transacional

Classe LMS	Suporte transacional
LMSMQBindingMQPP	Fase única
LMSMQBindingMQRFH1	Fase única
LMSMQBindingMQRFH2	Fase única
LMSMQMQBD	Fase única
LMSMQBinding	Fase única

Tabela 12. Classes do LMS e suporte transacional (continuação)

Classe LMS	Suporte transacional
LMSJMS	Fase única
LMSFILE	Sem suporte

Utilização do armazenamento do objeto JMS

Os nomes dos objetos JMS são armazenados utilizando a implementação de arquivo FSContext de JNDI, que vem como parte do MQSeries JMS SupportPac. O contexto (estrutura de diretórios) que o kernel utiliza para FSContext segue a hierarquia LDAP, utilizando a distinção de atributo com o valor associado para o nome do diretório. Por exemplo, para a hierarquia LDAP `o=ePIC, o=ePICApplications, epicappid=TEST1`, a estrutura de diretórios é `o-ePIC/o-ePICApplications/epicappid-TEST1`.

Para criar o contexto e os objetos, utilize a ferramenta JMS Admin fornecida com a instalação do JMS. As etapas básicas são: definição de contexto e, em seguida, alteração de contexto. A alteração de contexto o leva para o contexto. Crie os objetos JMS nos locais adequados. A seguir, veja os comandos de exemplo para criar a estrutura de contexto e os objetos JMS. Neste exemplo, o ID do aplicativo é `TEST1`.

```
#
# aqmjmscreatesample.scp 1.00 09Mar01
# Used for MQSeries Adapter Kernel
# Sample AQM JMS Configuration.
#
# Copyright (c) 2001 International Business Machines. All Rights Reserved.
#
# This configuration file is as an example only.
#
# IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS
# SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
# PURPOSE, OR NON-INFRINGEMENT.
#
# CopyrightVersion 1.0
#
#
# This is a script to use with the JMS administration (JMSAdmin) tool
# which comes with MQSeries Support pac MA88.
# This tool requires the JMSAdmin.config to be set to use either
# FSCONTEXT (file) or LDAP. This script is setup to work with
# FSCONTEXT, but will work with LDAP with the following changes:
# - Change the "-" signs to "=".  Example: define ctx(o-ePIC)
#   becomes define ctx(o=ePIC)
# - In LDAP the contexts have to already be defined using the
#   LDAP administration tool.  For example you do not need
#   to "define ctx(o=ePIC) but only change into it with the
```

```
#   "change ctx(o-ePIC)" command.
# - There are some notes in the following script which highlight
#   differences when using LDAP.
#
#
# Example usage: MQSeries root\java\bin\jmsadmin.bat < aqmjmscreatesample.scp
#
# Some helpful commands:
# "display ctx" will display the context's of the context you are
# currently in.
# "=UP" means return to the parent context.  Example: change ctx(=UP)
# "=INIT" means return to root context. In this example one directory level
# above o-ePIC.  Example: change ctx(=INIT)
# "define xxx" is for creating either a context or object.
# "change xxx" is for changing/moving into the context.
#
# Always required.
define ctx(o-ePIC)
change ctx(o-ePIC)
# Always required.
define ctx(o-ePICApplications)
change ctx(o-ePICApplications)
# Application id is TEST1, requires a context.
define ctx(epicappid-TEST1)
change ctx(epicappid-TEST1)
# Always required.
define ctx(cn-epicadapterrouting)
change ctx(cn-epicadapterrouting)
# This will hold the JMS QueueConnectionFactory object.
# Note: These two steps are not required for LDAP.
define ctx(cn-QCFTEST1)
change ctx(cn-QCFTEST1)
# Create the JMS QueueConnectionFactory object whose name is QCFTEST1
# Using MQSeries in server (bindings) mode.
define qcf(QCFTEST1) qmgr(yourQManagerName) tran(BIND)
change ctx(=UP)
# BodyCategory is DEFAULT
define ctx(epicbodycategory-DEFAULT)
change ctx(epicbodycategory-DEFAULT)
# BodyType is DEFAULT
define ctx(epicbodytype-DEFAULT)
change ctx(epicbodytype-DEFAULT)
# This will hold the JMS Queue object whose name is TEST1AIQ.
# Note: These two steps are not required for LDAP.
define ctx(cn-TEST1AIQ)
change ctx(cn-TEST1AIQ)
# Create the JMS Queue object whose name is TEST1AIQ
# q(JMS Q Object Name) queue(MQSeries Queue name)
define q(TEST1AIQ) queue(TEST1AIQ)
# Can move up and define other contexts and JMS objects.
# Quit the administration tool.
end
```

Apêndice B. Configurações válidas

Existem muitas configurações e combinações potenciais do MQSeries, Pacote do MQSeries Adapter e do MQSeries Integrator. Cada um desses membros da família de produtos MQSeries possui uma grande quantidade de recursos e configurações. Além disso, você pode combinar as funcionalidades no MQSeries, Pacote do MQSeries Adapter e MQSeries Integrator. Alguma funcionalidade em um membro da família de produtos MQSeries pode sobrepor parcialmente a funcionalidade fornecida por outros membros da família. Você deve determinar como utilizar e combinar as diversas funcionalidades de entrega e roteamento de mensagens no MQSeries, Pacote do MQSeries Adapter e MQSeries Integrator.

As configurações a seguir do MQSeries, Pacote do MQSeries Adapter e MQSeries Integrator foram validadas na época da publicação. Consulte o site do MQSeries na Web para obter as configurações válidas mais recentes.

MQSeries Adapter Kernel:

- Envio de mensagem com pedido de confirmação e sem pedido de confirmação.
- Uso dos modos de comunicação do MQSeries ou do JMS. Para os modos de comunicação vlaid, consulte "Apêndice A. Modos de Comunicação" na página 103.
- Entrega e roteamento de mensagens:
  - Envio de uma mensagem de um adaptador de origem para um adaptador de destino
  - Envio de uma mensagem de um adaptador de origem para diversos adaptadores de destino
  - Entrega de mensagens com diversos encadeamentos, ou seja, diversos operadores
  - Com o identificador lógico de destino definido como NONE na mensagem, para que o arquivo de configuração do kernel seja utilizado para determinar o identificador lógico de destino com base na categoria do corpo, tipo do corpo e identificador lógico de origem
  - Modelo push de entrega
  - Com o rastreamento ativado

  Nota: Consulte o "Apêndice C. Cabeçalhos de mensagens" na página 111. Contém os campos dos cabeçalhos de mensagens do MQSeries Adapter Kernel que o kernel ocupa e processa.

- Com os pré-requisitos mostrados nas seções “Hardware” na página 33 e “Software” na página 34.
- Utilização do arquivo de configuração, não LDAP, definir a configuração.

MQSeries:

- Não utilizando os clusters do MQSeries.

  Nota: Consulte o “Apêndice C. Cabeçalhos de mensagens” na página 111. Contém os campos dos cabeçalhos de mensagens do MQSeries Adapter Kernel que o kernel ocupa e processa.

MQSeries Integrator:

- O MQSeries Adapter Kernel e o MQSeries podem rotear e entregar a mensagem ao MQSeries Integrator. Consulte as informações sobre o MQSeries Integrator para determinar seus recursos para ser o intermediário dessas mensagens.
- Enviando mensagens do lado de origem do kernel, pelo MQSeries e MQSeries Integrator versão 2 e roteando diretamente para o lado de destino do kernel. No MQSeries Integrator, o fluxo de mensagens é configurado para rotear estaticamente. Todas as mensagens que chegam no nó do MQInput do fluxo são roteadas diretamente para a fila MQOutput específica.

  Nota: Consulte o “Apêndice C. Cabeçalhos de mensagens” na página 111 . Contém os campos dos cabeçalhos de mensagens do MQSeries Adapter Kernel que o kernel ocupa e processa.

Apêndice C. Cabeçalhos de mensagens

O Pacote do MQSeries Adapter utiliza diversos cabeçalhos de mensagens. Consulte a seção “Mensagem e formato de mensagem” na página 12 para saber quais cabeçalhos são utilizados sob quais circunstâncias.

Este apêndice lista e descreve os campos dos cabeçalhos das mensagens.

Cabeçalho descritor de mensagem do MQSeries Adapter Kernel

Valores do cabeçalho utilizado pelo MQSeries Adapter Kernel. Esses valores são posicionados nos objetos de retenção da mensagem. A coluna **Propagados em respostas?** lista se valor será ou não propagado de volta ao aplicativo de origem em uma mensagem de resposta quando o aplicativo de origem solicitar uma resposta. Alguns valores são utilizados apenas com o WebSphere Business Integrator.

Tabela 13. Cabeçalho do MQSeries Adapter Kernel

Nome do cabeçalho	Foram propagados em respostas?	Significado ou utilização
UniqueID	Não	Identificador único para cada mensagem.
TransactionID	Sim	Se for necessário, compartilhe o identificador de transação para cada mensagem e para suas respostas. Equivalente para um Extricity PublicProcessID ou um DataInterchange (DI) ApplicationID.
MessageType	Não	Utilizado para o gateway e as mensagens log/trace/exception.
SourceLogicalID	Não	Identificador lógico do aplicativo de origem. Equivalente para os nomes reservados no DI, Partner Agreement Manager (PAM) e o Business Flow Manager (BFM).

Tabela 13. Cabeçalho do MQSeries Adapter Kernel (continuação)

Nome do cabeçalho	Foram propagados em respostas?	Significado ou utilização
DestinationLogicalID	Não	Identificador lógico do aplicativo de destino. Para o DI e PAM, o valor padrão é `none`, mas pode ser substituído.
RespondToLogicalID	Sim	Identificador lógico para o qual a mensagem de resposta deve ser enviada. Copiado para o DestinationLogicalID para DI e para o SourceLogicalID para PAM.
CorrelationID	Não	Utilização reservada.
GroupStatus	Não	Utilização reservada.
ProcessingCategory	Não	Equivalente para um PAM Public Process Identifier ou um DI Command Process Identifier.
QosPolicy	Não	Utilização reservada.
DeliveryCategory	Não	Equivalente para um DI RequestorProfileID.
AckRequested	Não	Determina se aplicativo de origem ou não solicita uma mensagem de resposta.
PublicationTopic	Não	Utilização reservada.
SessionID	Não	Equivalente para um DI BatchID.
EncryptionStatus	Não	Determina o tipo de criptografia do corpo e assinatura.
TimeStampCreated	Não	Hora e data de quando a mensagem foi criada.
TimeStampExpired	Não	Data e hora depois das quais a mensagem não é mais significativa. Um valor `-1` significa sem expiração.
Size	Não	Utilização reservada.

Tabela 13. Cabeçalho do MQSeries Adapter Kernel (continuação)

Nome do cabeçalho	Foram propagados em respostas?	Significado ou utilização
BodyType	Não	Representa a finalidade específica da mensagem.
BodyCategory	Não	Representa o tipo de aplicativo da mensagem.
BodySecondaryType	Não	Utilização reservada.
UserArea	Não	Área geral para os dados do usuário.
RelatedSubjectID	Não	Utilizado para a correlação de interprocessos.
ExternalID	Não	Identificador de proprietário atual (por exemplo, um usuário ou parceiro de negócio) externo de um ambiente de aplicativo.
InternalID	Não	Identificador de proprietário atual (por exemplo, um usuário ou parceiro de negócio) no ambiente de aplicativo.
BodySignature	Não	Utilização reservada.
TransportCorrelationID	Sim	Utilização reservada.

Cabeçalho do descritor de mensagens do MQSeries

O conteúdo dos campos é determinado pelo MQSeries. O Pacote do MQSeries Adapter coloca as mensagem em filas conforme determinado pelos valores de controle da mensagem. Consulte a seção "valores de controle de mensagem" na página 16 para obter detalhes.

Tabela 14. Cabeçalhos do MQSeries

Seção ou campo	Significado ou utilização
Revision	Correção.
UniqueID	Cada mensagem tem um identificador único.
TransactionID	Uma mensagem e sua resposta compartilham o mesmo identificador de transação.
MessageType	Utilização reservada.
SourceLogicalID	Identificador lógico do aplicativo de origem.

Tabela 14. Cabeçalhos do MQSeries (continuação)

DestinationLogicalID	Identificador lógico do aplicativo de destino.
RespondToLogicalID	Um identificador lógico para o qual a mensagem de resposta deve ser enviada.
CorrelationID	Utilização reservada.
GroupStatus	Utilização reservada.
ProcessingCategory	Utilização reservada.
QosPolicy	Utilização reservada.
DeliveryCategory	Utilização reservada.
AckRequested	Determina se o aplicativo de origem solicita uma resposta ou não.
PublicationTopic	Utilização reservada.
SessionID	Utilização reservada.
EncryptionStatus	Utilização reservada.
TimeStampCreated	Hora e data de quando a mensagem foi criada.
TimeStampExpired	Data e hora depois das quais a mensagem não é mais significativa.
Size	Utilização reservada.
BodyCategory	Representa o tipo de aplicativo da mensagem, por exemplo, OAG ou RosettaNet.
BodyType	Representa o objetivo específico da mensagem, por exemplo, incluir pedidos ou sincronizar o inventário.
BodySecondaryType	Reservado.
UserArea	Área geral para os dados do usuário.
BodyData	Dados do corpo da mensagem.

MQSeries sem o MQSeries Integrator

Os valores do cabeçalho do kernel e os dados do corpo são colocados em um documento XML. A seguir, um exemplo do DTD que descreve o documento XML:

```
<!ELEMENT EPICHEADER (HEADER, EPICBODY,USERAREA*)>
<!ELEMENT HEADER (#PCDATA)>
<!ATTLIST HEADER Revision CDATA #FIXED "001">
<!ATTLIST HEADER UniqueID CDATA #REQUIRED>
<!ATTLIST HEADER TransactionID CDATA #REQUIRED>
<!ATTLIST HEADER MessageType CDATA #REQUIRED>
<!ATTLIST HEADER SourceLogicalID CDATA #REQUIRED>
<!ATTLIST HEADER DestinationLogicalID CDATA #REQUIRED>
```

```
<!ATTLIST HEADER RespondToLogicalID CDATA #IMPLIED>
<!ATTLIST HEADER CorrelationID CDATA #IMPLIED>
<!ATTLIST HEADER GroupStatus CDATA #IMPLIED>
<!ATTLIST HEADER ProcessingCategory CDATA #IMPLIED>
<!ATTLIST HEADER QosPolicy CDATA #IMPLIED>
<!ATTLIST HEADER DeliveryCategory CDATA #IMPLIED>
<!ATTLIST HEADER AckRequested CDATA #IMPLIED>
<!ATTLIST HEADER PublicationTopic CDATA #IMPLIED>
<!ATTLIST HEADER SessionID CDATA #IMPLIED>
<!ATTLIST HEADER EncryptionStatus CDATA #IMPLIED>
<!ATTLIST HEADER TimeStampCreated CDATA #REQUIRED>
<!ATTLIST HEADER TimeStampExpired CDATA #REQUIRED>
<!ATTLIST HEADER Size CDATA #IMPLIED>
<!ELEMENT EPICBODY (#PCDATA)> <!-- The data will be escaped -->
<!ATTLIST EPICBODY Size CDATA #IMPLIED>
<!ATTLIST EPICBODY BodyType CDATA #REQUIRED>
<!ATTLIST EPICBODY BodyCategory CDATA #REQUIRED>
<!ATTLIST EPICBODY BodySecondaryType CDATA #IMPLIED>
<!ELEMENT USERAREA (#PCDATA) >
```

Cabeçalho do MQSeries Integrator versão 1

O cabeçalho do MQSeries Integrator versão 1, RFH1, é composto pelos seguintes itens:

1. Parte corrigida
2. Cabeçalho Neon
3. Seção de dados, que contém o cabeçalho do kernel e os dados do corpo da mensagem

Tabela 15. Cabeçalho do MQSeries Integrator versão 1 — RFH1

Seção ou campo	**Significado ou utilização**
Parte corrigida	Utilizado conforme especificação no MQSeries Integrator versão 1.1.
Cabeçalho Neon	Segue o formato do cabeçalho Neon.
OPT_APP_GRP	Valor SourceLogicalId. Tirado do cabeçalho do kernel.
OPT_MSG_TYPE	BodyCategory+BodyType. Derivado do cabeçalho do kernel. Exemplo: Se BodyCategory for OAG e BodyType for SyncItem, o valor será OAG+SyncItem.
Seção de dados	Composto pelos valores de cabeçalho do kernel seguidos pelos dados do corpo da mensagem.

Tabela 15. Cabeçalho do MQSeries Integrator versão 1 — RFH1 (continuação)

Cabeçalho do kernel	O cabeçalho do Kernel está entre as tags <EPICHEADER>*cabeçalho*</EPICHEADER>. Os valores do cabeçalho do kernel estão na sintaxe XML. Apenas os atributos com valores estão presentes. Os dados reais não estão em linhas separadas. Exemplo do formato de um valor: <MessageType>*valor*</MessageType>.
MessageType	Utilização reservada.
SourceLogicalID	Identificador lógico do aplicativo de origem.
DestinationLogicalID	Identificador lógico do aplicativo de destino.
RespondToLogicalID	Identificador lógico para o qual a mensagem de resposta deve ser enviada.
TimeStampCreated	Hora e data de quando a mensagem foi criada.
TimeStampExpired	Data e hora depois das quais a mensagem não é mais significativa.
TransactionID	Uma mensagem e sua resposta compartilham o mesmo identificador de transação.
UniqueID	Cada mensagem tem um identificador único.
AckRequested	Determina se o aplicativo de origem solicita uma resposta.
ProcessingCategory	Reservado.
BodyCategory	Representa o tipo de aplicativo da mensagem, por exemplo, OAG ou RosettaNet.
BodyType	Representa o objetivo específico da mensagem, por exemplo, incluir pedidos ou sincronizar o inventário.
BodySecondaryType	Reservado.
UserArea	Dados do aplicativo específico para a integração do usuário.
MsgHeaderVersion	Versão do cabeçalho do kernel (reservado).
CorrelationID	Integração específica do usuário.
GroupStatus	Integração específica do usuário.
QosPolicy	Reservado.
DeliveryCategory	Reservado.
PublicationTopic	Reservado.
SessionID	Reservado.
EncryptionStatus	Reservado.

Tabela 15. Cabeçalho do MQSeries Integrator versão 1 — RFH1 (continuação)

Dados do corpo da mensagem	Dados do corpo da mensagem.

Cabeçalho do MQSeries Integrator versão 2

O cabeçalho do MQSeries Integrator versão 2, RFH2, é composto pelos seguintes itens:

1. Parte corrigida
2. Pasta <mcd> — descritor do conteúdo da mensagem
3. Pasta <usr> — propriedades definidas pelo aplicativo (usuário)
4. Seção de dados, que contém o cabeçalho do kernel e os dados do corpo da mensagem

Tabela 16. Cabeçalho do MQSeries Integrator versão 2 — RFH2

Seção ou campo	**Significado ou utilização**
Parte corrigida	Utilizado conforme especificação no MQSeries Integrator versão 2.
<mcd>	XML se a mensagem for XML. Siga as regras do MQSeries Integrator versão 2.
conjunto	Não utilizado pelo kernel.
tipo	Não utilizado pelo kernel.
formato	XML se a mensagem for XML. Siga as regras do MQSeries Integrator versão 2.
pasta <usr> — propriedades definidas pelo aplicativo (usuário)	Composto pelos valores do cabeçalho do kernel.
Cabeçalho do kernel	Apenas os atributos com valores estão presentes. Os dados reais não estão em linhas separadas.
SourceLogicalID	Identificador lógico do aplicativo de origem.
DestinationLogicalID	Identificador lógico do aplicativo de destino.
MessageType	Utilização reservada.
RespondToLogicalID	Um identificador lógico para o qual a mensagem de resposta deve ser enviada.
TimeStampCreated	Hora e data de quando a mensagem foi criada.
TimeStampExpired	Data e hora depois das quais a mensagem não é mais significativa.
TransactionID	Uma mensagem e sua resposta compartilham o mesmo identificador de transação.
UniqueID	Cada mensagem tem um identificador único.

Tabela 16. Cabeçalho do MQSeries Integrator versão 2 — RFH2 (continuação)

ProcessingCategory	Reservado.
BodyCategory	Representa o tipo de aplicativo da mensagem, por exemplo, OAG ou RosettaNet.
BodyType	Representa o objetivo específico da mensagem, por exemplo, incluir pedidos ou sincronizar o inventário.
BodySecondaryType	Reservado.
AckRequested	Determina se o aplicativo de origem solicita uma resposta.
UserArea	Dados do aplicativo específico para a integração do usuário.
MsgHeaderVersion	Versão do cabeçalho do kernel (reservado).
CorrelationID	Integração específica do usuário.
GroupStatus	Integração específica do usuário.
QosPolicy	Reservado.
DeliveryCategory	Reservado.
PublicationTopic	Reservado.
SessionID	Reservado.
EncryptionStatus	Reservado.
Seção de dados	Dados do corpo da mensagem.

Apêndice D. Amostra do arquivo de configuração

Esta seção lista a versão do arquivo `aqmconfig.xml` que era atual no momento desta publicação. A seção "Amostra de um arquivo de configuração mínimo" na página 123 lista a versão do arquivo `aqmconfig.minimum.xml` que era atual no momento desta publicação. Consulte os arquivos `aqmconfig.xml` e `aqmconfig.minimum.xml` no diretório de instalação do kernel `samples` para obter informações da edição mais recente; os exemplos listados aqui estão possivelmente desatualizados.

Consulte a seção "O arquivo de configuração" na página 66 para obter informações sobre a interpretação e edição do arquivo de configuração.

Diversos identificadores do aplicativo estão incluídos neste exemplo de arquivo de configuração. Um conjunto de entradas é listado em cada identificador do aplicativo. A amostra do arquivo de configuração contém os seguintes identificadores do aplicativo:

- TEST1
- TEST1Daemon
- TEST2
- TEST3
- TraceClient
- TraceServer

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- aqmconfig.xml 1.01 09Mar01                                          -->
<!-- Used for MQSeries Adapter Kernel -->
<!-- Sample AQM Configuration.            -->
<!--  -->
<!-- Copyright (c) 2001 International Business Machines. All Rights Reserved. -->
<!--  -->
<!-- This configuration file is as an example only. -->
<!--  -->
<!-- IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS -->
<!-- SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE -->
<!-- IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR -->
<!-- PURPOSE, OR NON-INFRINGEMENT. -->
<!--  -->
<!-- CopyrightVersion 1.0 -->
<!--  -->

   <Epic o="ePIC">
      <!-- If getObject is called this indicates the top level directory -->
      <!-- where the JNDI file system context will retrieve objects from. -->
      <!-- This defaults to the current directory if this key is not present. -->
      <!-- All applications share this context root. -->
      <context>file:///epic/configContext</context>
      <!-- Example using a drive letter 'c' -->
      <!--
      <context>file://c:/E/runtimefiles</context>
      -->
      <ePICApplications o="ePICApplications">
         <!-- The following is for sample Test Application ID: TEST1 with a  -->
         <!-- sample AdapterDaemon named TEST1Daemon  -->
         <ePICApplication epicappid="TEST1">
     <!--  Audit Logging on/off.  Requires WebSphere Business Integrator product. -->
     <!--  If no entry defaults to false. -->
```

```
      <epiclogging>false</epiclogging>
<!--  Tracing on/off.  If no entry defaults to false. -->
      <epictrace>false</epictrace>
<!-- Trace levels - Uses the jlog com.ibm.logging.IRecordType constants, -->
<!-- common constants: -->
<!-- 0=TYPE_NONE (No messages), 1=TYPE_INFO, 512=TYPE_ERROR_EXC (Exceptions), -->
<!-- 513=TYPE_INFO | TYPE_ERROR_EXC, -1=TYPE_ALL (All possible messages). -->
<!-- No entry defaults to TYPE_NONE -->
      <epictracelevel>-1</epictracelevel>
<!-- Name of the Trace application id.  Will be used for -->
<!-- trace configuration information.  Defaults to TraceClient -->
      <epictraceclientid>TraceClient</epictraceclientid>
<!-- When processing messages into the application. -->
<!-- LogonInfo class name used for connecting to an application. -->
  <!-- Will be used by the AdapterDaemon.  If no entry will default -->
<!-- to com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault. -->
<epiclogoninfoclassname>com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault
</epiclogoninfoclassname>
      <AdapterRouting cn="epicadapterrouting">
         <!-- MQSeries Q Manager for this application use, no entry -->
         <!-- uses the default Q Manager.  A value of DEFAULT means -->
         <!-- use the default Q Manager. -->
         <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
         <!-- Use the remote Q Manager for sending messages.  Remote queue -->
         <!-- definitions are not required.  true - use remote Q Manager, -->
         <!-- false - do not use remote Q Manager.  No entry defaults to false -->
         <epicuseremotequeuemanagertosend>false</epicuseremotequeuemanagertosend>
         <!-- MQSeries Client hostname for where the MQSeries server -->
         <!-- resides for TEST1.  Required if using MQSeries Client -->
         <!--
         <epicmqppqueuemgrhostname>localhost</epicmqppqueuemgrhostname>
   -->
         <!-- MQSeries Client port to use for where the MQSeries server -->
         <!-- resides for TEST1.  No entry defaults to MQSeries default -->
         <!--
         <epicmqppqueuemgrportnumber>1414</epicmqppqueuemgrportnumber>
   -->
         <!-- MQSeries Client channel name to use for the MQSeries server, required -->
         <!--
         <epicmqppqueuemgrchannelname>xyz</epicmqppqueuemgrchannelname>
   -->
         <!-- JMS example for TEST1.  Refers to the JMS Connection factory name. -->
               <!-- Requires the attribute describing the object plus the attributes value. -->
               <!-- For JMS the attribute is 'cn'. -->
         <!--
         <epicjmsconnectionfactoryname>cn=QCFTEST1</epicjmsconnectionfactoryname>
   -->
         <ePICBodyCategory epicbodycategory="DEFAULT">
            <ePICBodyType epicbodytype="DEFAULT">
<!-- Contains the Command selection criteria when processing -->
<!-- a message into an application.  Will be used by the -->
<!-- AdapterDaemon - Command to invoke. -->
               <epiccommandclassname>com.ibm.epic.adapters.eak.samples.SampleCAdapterWrapper
               </epiccommandclassname>
<!-- Represents the type of command the "epiccommandclassname" -->
<!-- represents.  MQAKEAB is the EAB style interface.  MQAKEJB -->
<!-- is an EJB Service Session Bean.  No entry defaults to MQAKEAB -->
               <epiccommandtype>MQAKEAB</epiccommandtype>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- method name to invoke.  No entry defaults to "execute". -->
               <epiccommandejbmethod>execute</epiccommandejbmethod>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- parameter type for the method specified by "epiccommandejbmethod". -->
<!-- This will be the same datatype returned by the -->
<!-- TerminalDataContainerMapper.  No entry defaults -->
<!-- to "com.ibm.mqao.mqak.ejbclient.TDCGenericMapper". -->
               <epiccommandejbmethodparmtype>com.ibm.mqao.mqak.ejbclient.TDCGenericMapper
               </epiccommandejbmethodparmtype>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- URL where the EJB specified in "epiccommandclassname" -->
<!-- has been deployed in the form "IIOP://hostname:900/". -->
<!-- No entry defaults to "IIOP:///". -->
               <epiccommandejburl>IIOP:///</epiccommandejburl>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- name of the Initial Context Factory used to to lookup the -->
<!-- home name for the EJB specified in "epiccommandclassname". -->
<!-- No entry defaults to "com.ibm.ejs.ns.jndi.CNInitialContextFactory". -->
               <epiccommandejbinitialcontext>com.ibm.ejs.ns.jndi.CNInitialContextFactory
               </epiccommandejbinitialcontext>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- name of the Mapper the Worker uses for creating the  -->
<!-- "epiccommandejbmethodparmtype" object passed in to the  -->
<!-- "epiccommandejbmethod" in to the "epiccommandclassname". -->
<!-- No entry defaults to "com.ibm.mqao.mqak.ejbclient.TDCGenericMapper". -->
               <epiccommandejbmapper>com.ibm.mqao.mqak.ejbclient.TDCGenericMapper
```

```
                </epiccommandejbmapper>
<!-- Default destinations to send messages to. -->
        <!-- Single destination. -->
                <epicdestids>TEST2</epicdestids>
        <!-- Multiple destinations. -->
                <!--
                <epicdestids>
                     <Value>TEST2</Value>
                     <Value>TEST3</Value>
                </epicdestids>
                -->
                <!-- Receive transport communications mode this application -->
                <!-- wants for receiving messages. -->
                <!-- For MQSeries normal mode use MQPP. -->
                <!-- For MQSeries using an RFH1 header format use MQRFH1, -->
                <!-- when using MQSeries Integrator V1 -->
                <!-- For MQSeries using an RFH2 header format use MQRFH2, -->
                <!-- when using MQSeries Integrator V2 -->
                <!-- For file normal mode use FILE. -->
                <epicreceivemode>MQPP</epicreceivemode>
                <!-- How to format the message for the receive mode. -->
                <!-- Entry is the class name of the formatter which -->
                <!-- must be for the receive mode -->
                <!-- Receive modes MQPP, MQRFH1, MQRFH2, FILE have -->
                <!-- default receive modes -->
                <epicmessageformatter>com.ibm.epic.adapters.eak.nativeadapter.MQNMBDFormatter
                </epicmessageformatter>
 <!-- JMS formatter for mode for MQSeries provider implementation -->
 <!--
                <epicmessageformatter>com.ibm.epic.adapters.eak.nativeadapter.JMSNMRFH2Formatter
                </epicmessageformatter>
 -->
 <!-- Receive Time out in milliseconds ie. 1000 = 1 second, -->
 <!-- -1 means never ending.  No entry defaults to 0 -->
 <!-- milliseconds.  Used when receiving messages. -->
                <epicreceivetimeout>30000</epicreceivetimeout>
 <!-- MQSeries queue for this application to receive messages -->
 <!-- from for receive modes MQPP, MQRFH1, MQRFH2 -->
                <epicreceivemqppqueue>TEST1AIQ</epicreceivemqppqueue>
 <!-- MQSeries queue required by the AdapterWorker when -->
 <!-- errors encountered processing a message -->
 <!-- for receive modes MQPP, MQRFH1, MQRFH2 -->
                <epicerrormqppqueue>TEST1AEQ</epicerrormqppqueue>
 <!-- MQSeries reply queue required for synchronous request/replies -->
 <!-- for receive modes MQPP, MQRFH1, MQRFH2 -->
                <epicreplymqppqueue>TEST1RPL</epicreplymqppqueue>
 <!-- JMS receive mode, refers to the JMS queue object name for -->
 <!-- this application to receive messages from. -->
                <!-- Requires the attribute describing the object plus the attribute's value. -->
                <!-- For JMS the attribute is 'cn'. -->
                <epicjmsreceivequeuename>cn=TEST1AIQ</epicjmsreceivequeuename>
 <!-- JMS receive mode, refers to the JMS queue object name for -->
 <!-- errors required by the AdapterWorker when errors -->
 <!-- encountered processing a message. -->
                <!-- Requires the attribute describing the object plus the attribute's value. -->
                <!-- For JMS the attribute is 'cn'. -->
                <epicjmserrorqueuename>cn=TEST1AEQ</epicjmserrorqueuename>
 <!-- JMS receive mode, refers to the JMS queue object name for -->
 <!-- the reply queue, required for synchronous request/replies -->
                <!-- Requires the attribute describing the object plus the attribute's value. -->
                <!-- For JMS the attribute is 'cn'. -->
                <epicjmsreplyqueuename>cn=TEST1RPL</epicjmsreplyqueuename>
 <!-- In FILE receive mode, directory for this application to receive messages from -->
                <epicreceivefiledir>./TEST1AID</epicreceivefiledir>
 <!-- In FILE receive mode, interim directory for this application to -->
 <!-- hold received messages until committed. -->
                <epiccommitfiledir>./TEST1ACD</epiccommitfiledir>
 <!-- In FILE receive mode, directory for this application to put error messages -->
 <!-- File receive mode, directory required by the AdapterWorker when -->
 <!-- errors encountered processing a message -->
                <epicerrorfiledir>./TEST1AED</epicerrorfiledir>
            </ePIBodyType>
         </ePICBodyCategory>
      </AdapterRouting>
   </ePICApplication>
   <!-- The following is for sample AdapterDaemon 'TEST1Daemon' -->
   <!-- for the 'TEST1' application  -->
   <ePICApplication epicappid="TEST1Daemon">
      <epictrace>false</epictrace>
      <epictracelevel>-1</epictracelevel>
      <ePICAdapterDaemonExtensions cn="epicappextensions">
  <!-- Dependency appid, if no entry then will default -->
  <!-- to the application id of the daemon.     -->
         <epicdepappid>TEST1</epicdepappid>
  <!-- Minimum number of workers the AdapterDaemon will start. -->
```

```
      <!-- No entry defaults to 1. -->
            <epicminworkers>1</epicminworkers>
         </ePICAdapterDaemonExtensions>
       </ePICApplication>
       <!-- The following is for Test Application ID: TEST2 -->
       <!-- Refer to TEST1 for explanations and possible additional entries. -->
       <ePICApplication epicappid="TEST2">
          <epictrace>true</epictrace>
          <epictracelevel>512</epictracelevel>
          <AdapterRouting cn="epicadapterrouting">
             <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
             <ePICBodyCategory epicbodycategory="DEFAULT">
                <ePICBodyType epicbodytype="DEFAULT">
                   <epiccommandclassname>com.ibm.epic.adapters.eak.test.InstallVerificationTest
                   </epiccommandclassname>
                   <epicreceivemode>MQPP</epicreceivemode>
                   <epicreceivemqpppqueue>TEST2AIQ</epicreceivemqpppqueue>
                   <epicerrormqpppqueue>TEST2AEQ</epicerrormqpppqueue>
                   <epicreplymqpppqueue>TEST2RPL</epicreplymqpppqueue>
                </ePICBodyType>
             </ePICBodyCategory>
          </AdapterRouting>
       </ePICApplication>
       <!-- The following is for Test Application ID: TEST3 -->
       <!-- Refer to TEST1 for explanations and possible additional entries. -->
       <ePICApplication epicappid="TEST3">
          <AdapterRouting cn="epicadapterrouting">
             <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
             <ePICBodyCategory epicbodycategory="DEFAULT">
                <ePICBodyType epicbodytype="DEFAULT">
                   <epicdestids>TEST1</epicdestids>
                   <epicreceivemode>MQPP</epicreceivemode>
                   <epicreceivemqpppqueue>TEST3AIQ</epicreceivemqpppqueue>
                </ePICBodyType>
             </ePICBodyCategory>
          </AdapterRouting>
       </ePICApplication>
<!-- The following is for sample Trace Client Application ID: TraceClient -->
<!-- Contains the TraceClient configuration information for doing tracing. -->
<!-- This is the application id value in the 'epictraceclientid' element -->
<!-- configured for the application wanting to do tracing -->
       <ePICApplication epicappid="TraceClient">
          <ePICTraceExtensions cn="epicappextensions">
       <!-- Dependency Trace Server application id used for SocketHandler -->
       <!-- and ENAHandler (uses MQSeries), defaults to TraceServer -->
             <epicdepappid>TraceServer</epicdepappid>
             <!-- Write messages synchronously (true) or asynchronously (false), -->
             <!-- defaults to false (write messages asynchronously).  This is    -->
             <!-- used when giving the messages to the handlers. -->
             <epictracesyncoperation>false</epictracesyncoperation>
       <!-- Default Trace message file to use if none passed in to the -->
          <!-- writeTrace method call. Defaults to this file if not indicated -->
             <epictracemessagefile>com.ibm.epic.trace.client.TraceMessage</epictracemessagefile>
       <!-- Handlers to load.  Handlers do the actual processing of the -->
       <!-- Trace message.  If the default trace client id 'TraceClient' -->
       <!-- is used then the handler defaults to the -->
       <!-- com.ibm.logging.ConsoleHandler.  If the default trace client -->
       <!-- id 'TraceClient' is not used, the handler has to be specified. -->
       <!-- A Single Trace Handler -->
             <epictracehandler>com.ibm.logging.ConsoleHandler</epictracehandler>
       <!-- Multiple Trace Handlers -->
       <!--
             <epictracehandler>
                <Value>com.ibm.logging.ConsoleHandler</Value>
                <Value>com.ibm.logging.SocketHandler</Value>
             </epictracehandler>
       -->
             <!-- Handler definitions.  Available definitions depend on the -->
             <!-- handler.  Formatters are used for formatting the trace message.-->
             <ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">
                <!-- ConsoleHandler formatter to use, defaults to this formatter if none provided. -->
                <epictraceformatter>com.ibm.epic.trace.client.EpicTraceFormatter</epictraceformatter>
             </ePICTraceHandler>
             <ePICTraceHandler epictracehandler="com.ibm.logging.FileHandler">
                <!-- FileHandler formatter to use, defaults to this formatter if none provided. -->
                <epictraceformatter>com.ibm.epic.trace.client.EpicTraceFormatter</epictraceformatter>
                <!-- Trace filename to use, defaults to trc.log in the current directory. -->
                <epictracefilename>trc.log</epictracefilename>
             </ePICTraceHandler>
             <ePICTraceHandler epictracehandler="com.ibm.epic.trace.client.ENAHandler">
                <!-- ENAHandler formatter to use, defaults to this formatter if none provided. -->
                <epictraceformatter>com.ibm.epic.trace.client.EpicXMLFormatter</epictraceformatter>
             </ePICTraceHandler>
             <ePICTraceHandler epictracehandler="com.ibm.logging.SocketHandler">
                <!-- SocketHandler formatter to use, defaults to this formatter if none provided. -->
```

```
                <epictraceformatter>com.ibm.epic.trace.client.EpicXMLFormatter</epictraceformatter>
             </ePICTraceHandler>
          </ePICTraceExtensions>
       </ePICApplication>
<!-- The following is for sample Trace Server Application ID: TraceServer -->
<!-- Contains the TraceServer configuration information. -->
<!-- This is the application id pointed to by the trace client -->
<!-- epicdepappid value.  Definitions are similar to TraceClient example. -->
       <ePICApplication epicappid="TraceServer">
          <AdapterRouting cn="epicadapterrouting">
             <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
             <ePICBodyCategory epicbodycategory="DEFAULT">
                <ePICBodyType epicbodytype="DEFAULT">
                   <epicreceivemode>MQPP</epicreceivemode>
                   <epicreceivemqppqueue>TraceServerAIQ</epicreceivemqppqueue>
                </ePICBodyType>
             </ePICBodyCategory>
          </AdapterRouting>
          <ePICTraceExtensions cn="epicappextensions">
      <!-- Write messages synchronously/asynchronously (true/false (default)). -->
             <epictracesyncoperation>false</epictracesyncoperation>
      <!-- Trace message file.  Defaults to this file if not indicated -->
             <epictracemessagefile>com.ibm.epic.trace.server.TraceServerMessage</epictracemessagefile>
             <!-- Handlers to load, for multiple handlers see TraceClient example. -->
             <!-- If the default trace server id 'TraceServer' is used then the handler -->
             <!-- defaults to the com.ibm.logging.MultiFileHandler. -->
             <!-- Note: Do not use SocketHandler or ENAHandler for the trace server. -->
             <epictracehandler>com.ibm.logging.MultiFileHandler</epictracehandler>
      <!-- Handler definitions for com.ibm.logging.SocketHandler -->
      <!-- Formatter to use, defaults to this formatter if none provided.-->
             <ePICTraceHandler epictracehandler="com.ibm.logging.SocketHandler">
 <!-- Entries when using socket handler from the TraceClient and -->
 <!-- starting the Trace Server in socket receive mode. -->
 <!-- SocketHandler host machine, defaults to localhost -->
                <epictracesocketserverhost>localhost</epictracesocketserverhost>
         <!-- SocketHandler port number, defaults to 8181 -->
                <epictraceportnumber>8181</epictraceportnumber>
             </ePICTraceHandler>
      <!-- Formatter to use, defaults to this formatter if none provided.-->
             <ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">
                <!-- ConsoleHandler formatter to use, defaults to this formatter if none provided. -->
                <epictraceformatter>com.ibm.epic.trace.client.ReFormatter</epictraceformatter>
             </ePICTraceHandler>
             <ePICTraceHandler epictracehandler="com.ibm.logging.MultiFileHandler">
 <!-- MultiFileHandler formatter to use, defaults to this formatter if none provided. -->
                <epictraceformatter>com.ibm.epic.trace.client.ReFormatter</epictraceformatter>
 <!-- MultiFileHandler trace base filename to use, defaults to trc.log in the -->
 <!-- current directory.  The actual filename will be for this -->
 <!-- example trcx.log, where x is a numeric number starting at -->
 <!-- 0 and going up to the number of trace files specified. -->
                <epictracefilename>trc.log</epictracefilename>
 <!-- MultiFileHandler number of trace files, defaults to 3 -->
                <epictracefilenumber>3</epictracefilenumber>
 <!-- MultiFileHandler file size in number of bytes, defaults to -->
                <epictracefilesize>1000000</epictracefilesize>
             </ePICTraceHandler>
          </ePICTraceExtensions>
       </ePICApplication>
    </ePICApplications>
```

Amostra de um arquivo de configuração mínimo

Esta seção fornece um exemplo de um arquivo de configuração mínimo para ser utilizado com o MQSeries Adapter Kernel. Consulte a seção "Inclusão de informações do adaptador na configuração" na página 86 para obter informações sobre o arquivo de configuração mínimo.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- aqmconfig.minimum.xml 1.00 00/11/07                                    -->
<!-- Used for MQSeries Adapter Kernel -->
<!-- Sample AQM Configuration showing a minimum configuration for the       -->
<!-- following conditions:                                                   -->
<!-- 1) Going from applicationid TEST1 to TEST2.  TEST1 is not receiving  -->
<!--    messages. -->
<!-- 2) TEST2 has no special application requirements.  -->
<!-- 3) TEST2 is using 1 worker.                          -->
<!-- 4) Using MQSeries with the default QManager installed on each machine.    -->
<!--    and using default format.                                           -->
<!-- 5) No specific body category and body type being used.                 -->
<!-- 6) Using default tracing to the console.   -->
```

```
<!--  -->
<!--  -->
<!--  -->
<!-- Copyright (c) 2000 International Business Machines. All Rights Reserved. -->
<!--  -->
<!-- This configuration file is as an example only. -->
<!--  -->
<!-- IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS -->
<!-- SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE -->
<!-- IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR -->
<!-- PURPOSE, OR NON-INFRINGEMENT. -->
<!--  -->
<!-- CopyrightVersion 1.0 -->
<!--     -->
   <Epic o="ePIC">
      <ePICApplications o="ePICApplications">
         <!-- The following is for sample Test Application ID: TEST1  -->
         <ePICApplication epicappid="TEST1">
    <!--  Tracing on/off.  If no entry defaults to false. -->
            <epictrace>false</epictrace>
    <!-- Trace levels - 512=TYPE_ERROR_EXC (Exceptions),-1=TYPE_ALL (All possible messages). -->
            <epictracelevel>0</epictracelevel>
            <AdapterRouting cn="epicadapterrouting">
               <epicmqpqueuemgr>DEFAULT</epicmqpqueuemgr>
               <ePICBodyCategory epicbodycategory="DEFAULT">
                  <ePICBodyType epicbodytype="DEFAULT">
    <!-- Default destinations to send messages to. -->
                     <epicdestids>TEST2</epicdestids>
                  </ePICBodyType>
               </ePICBodyCategory>
            </AdapterRouting>
         </ePICApplication>
         <!-- The following is for Test Application ID: TEST2 -->
         <ePICApplication epicappid="TEST2">
            <epictrace>false</epictrace>
            <epictracelevel>512</epictracelevel>
            <AdapterRouting cn="epicadapterrouting">
               <epicmqpqueuemgr>DEFAULT</epicmqpqueuemgr>
               <ePICBodyCategory epicbodycategory="DEFAULT">
                  <ePICBodyType epicbodytype="DEFAULT">
    <!-- AdapterDaemon - Command to invoke. -->
                     <epiccommandclassname>com.ibm.epic.adapters.eak.samples.SampleCAdapterWrapper
                     </epiccommandclassname>
                     <epicreceivemode>MQ</epicreceivemode>
     <!-- Receive Time out in milliseconds ie. 1000 = 1 second, -->
     <!-- -1 means never ending.  No entry defaults to 0. -->
     <!-- milliseconds.  Used when receiving messages. -->
                     <epicreceivetimeout>30000</epicreceivetimeout>
                     <epicreceivemqppqueue>TEST2AIQ</epicreceivemqppqueue>
                     <epicerrormqppqueue>TEST2AEQ</epicerrormqppqueue>
                     <epicreplymqppqueue>TEST2RPL</epicreplymqppqueue>
                  </ePICBodyType>
               </ePICBodyCategory>
            </AdapterRouting>
         </ePICApplication>
      </ePICApplications>
   </Epic>
```

Apêndice E. Amostra do arquivo de configuração

A seguir, um exemplo do arquivo aqmsetup, que define muitos dos valores iniciais de configuração do kernel, incluindo diversas variáveis de ambiente. Consulte a seção "O arquivo de instalação" na página 66 para obter informações adicionais sobre este arquivo. O arquivo aqmsetup está localizado no diretório de samples do diretório raiz de instalação do kernel.

```
#
# aqmsetup 1.01 01/03/27
# Sample AQM Adapter runtime parameter configuration file entries.
#
# Copyright (c) 2001 International Business Machines. All Rights Reserved.
#
# This configuration file is as an example only.
#
# IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS
# SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
# PURPOSE, OR NON-INFRINGEMENT.
#
# CopyrightVersion 1.0
#
#
# Pound (#) signs are comments.
#
##################################################################
#
# Use Websphere Business Integrator(WSI) product 5724-A78 LDAP
# Directory Services or configuration file. No entry defaults to
# true (use configuration file). To use the WSI directory service
# set the value to false.  Refer to the WSI documentation for
# specifics on using the directory service.
#AdapterDirectoryUseFileFlag=true
# When using Websphere Business Integrator(WSI) product 5724-A78 LDAP
# Directory Services this additional entry is required. Refer to the
# WSI documentation for specifics on using the directory service.
#DirectoryServices=ChangeToDestDir/samples/DirectoryServices.properties
# Location of configuration file aqmconfig.xml when not using
# the Websphere Business Integrator(WSI) product 5724-A78 LDAP
# Directory Services.
# No entry defaults to current directory.
#AQMConfig=ChangeToDestDir/samples
#
##################################################################
##################################################################
# XML DTD Catalogs and Directories - where to locate DTD's if not
# in the current directory.
# Format: XML_DTD_DIRECTORY_x=ddd where x is a numeric suffix to
# be incremented for each key and ddd is the directory.
# The numeric suffix's must start with 1 and be contiguous.
```

```
#####################################################################
XML_DTD_DIRECTORY_1=ChangeToDestDir/runtimefiles/oag
#XML_DTD_DIRECTORY_2=ChangeToDestDir/runtimefiles
#
#####################################################################
# Java JNI Environment Variables for C Interface for increasing
# the amount of memory used.  This applies to when a C module
# is instantiating a JVM.  When a C Interface is being called
# from within JAVA the JVM is already established.
#####################################################################
# The stack memory is used for holding local function, function
# parameters, local variable references.
# Native stack is used for non-Java calls from within Java such
# as to C code.  Stack size in bytes to use.
# Default is 128 kilobytes on NT.
#AQM_JNI_NATIVESTACKSIZE=1048576
# Java stack is for Java method calls and local variables.
# Stack size in bytes to use.
# Default is 400 kilobytes on NT.
#AQM_JNI_JAVASTACKSIZE=4194304
# The heap memory is used for storing instantiated Java objects
# Minimum heap size in bytes to start with.
# Default is 1 megabyte on NT.
#AQM_JNI_MINHEAPSIZE=16777216
# Maximum heap size in bytes which can be used.
# Default is 16 megabytes on NT.
#AQM_JNI_MAXHEAPSIZE=268435426
#
#####################################################################
#  Designate end of configuration file
#####################################################################
*ENDCFG
```

Avisos

Estas informações foram desenvolvidas para produtos e serviços oferecidos nos Estados Unidos. A IBM pode não oferecer os produtos, serviços ou recursos discutidos nessas informações em outros países. Consulte o representante IBM de sua localidade para obter informações sobre os produtos e serviços disponíveis atualmente em sua região. Referências a produtos, programas ou serviços IBM não significam que apenas os produtos, programas ou serviços IBM possam ser utilizados. Qualquer produto, programa ou serviço funcionalmente equivalente, que não infrinja nenhum direito de propriedade intelectual da IBM, poderá ser utilizado. Entretanto, o usuário é responsável pela avaliação e verificação da operação de qualquer produto, programa ou serviço.

A IBM pode ter patentes, ou solicitações de patentes relativas a assuntos tratados nesta publicação. O fornecimento desta publicação não garante ao Cliente nenhum direito sobre tais patentes. Pedidos de licença devem ser enviados, por escrito, à:

Gerência de Relações Comerciais e Industriais
Av. Pasteur 138-146 / Botafogo
CEP: 22290-240
Rio de Janeiro - RJ
Brasil

O parágrafo a seguir não se aplica a nenhum país em que tais disposições estejam inconsistentes com a legislação local: A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO "COMO ESTÁ" SEM GARANTIA DE ESPÉCIE ALGUMA, EXPLÍCITA OU IMPLÍCITA, INCLUINDO, MAS NÃO SE LIMITANDO ÀS GARANTIAS IMPLÍCITAS DE NÃO-VIOLAÇÃO, DE COMERCIALIZAÇÃO OU DE ADEQUAÇÃO A UM FIM ESPECÍFICO. Alguns países não permitem a exclusão de garantias expressas ou implícitas em certas transações; portanto, esta disposição pode não se aplicar a você.

Estas informações podem incluir imprecisões técnicas ou erros tipográficos. Periodicamente, são feitas alterações nas informações aqui contidas; tais alterações serão incorporadas em futuras edições desta publicação. A IBM pode, a qualquer momento, aperfeiçoar e/ou alterar o(s) produto(s) e/ou programa(s) descrito(s) nestas informações sem aviso prévio.

Referências nesta publicação a sites não-IBM são fornecidas apenas por conveniência e não representam de forma alguma um endosso a esses sites

Web. Os materiais contidos nesses sites Web não fazem parte dos materiais deste produto IBM e a utilização destes sites é sua responsabilidade.

A IBM pode utilizar ou distribuir as informações fornecidas da forma que julgar apropriada sem incorrer em qualquer obrigação para com você.

Licenciados deste programa que desejam obter informações sobre este assunto com objetivo de: (i) trocar informações entre programas criados independentemente e outros programas (incluindo este) e (ii) utilizar mutuamente as informações que foram trocadas, devem entrar em contato com:

Centro de Atendimento a Clientes IBM
Av. Pasteur 138/146
CEP: 22290-240
Botafogo,
Rio de Janeiro - RJ,
Brasil.

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 são fornecidos pela IBM sob os termos do Contrato com o Cliente IBM, Contrato de Licença de Programa Internacional IBM ou qualquer contrato equivalente.

Quaisquer dados de desempenho contidos aqui foram determinados em um ambiente controlado. No entanto, os resultados obtidos em outros ambientes operacionais podem variar significativamente. Algumas medições podem ter sido feitas em sistemas em desenvolvimento e não existe garantia de que essas medições sejam as mesmas nos sistemas geralmente disponíveis. Além disso, algumas medições podem ter sido estimadas por extrapolação. Os resultados reais podem variar. Os usuários deste documento devem verificar os dados aplicáveis a seus ambientes específicos.

As informações sobre produtos de outros fabricantes foram obtidas junto aos fornecedores dos respectivos produtos, seus anúncios públicos e outras fontes disponíveis publicamente. A IBM não efetuou nenhum teste nesses produtos e não pode afirmar a precisão de seu desempenho, compatibilidade ou qualquer outro requisito. Perguntas sobre as capacidades de produtos de outros fabricantes devem ser endereçadas aos respectivos fornecedores desses produtos.

Marcas

Os termos a seguir são marcas da International Business Machines Corporation nos Estados Unidos e/ou em outros países:

AIX	OS/400
AS/400	RISC System/6000
IBM	RS/6000
MQSeries	WebSphere

Lotus e LotusScript são marcas da Lotus Development Corporation nos Estados Unidos e/ou em outros países.

Java e todas as marcas e logotipos baseados em Java são marcas ou marcas de serviço da Sun Microsystems, Inc. nos Estados Unidos e/ou em outros países.

Windows, Windows NT e o logotipo do Windows são marcas de serviço da Microsoft Corporation nos Estados Unidos e/ou em outros países.

Outros nomes de empresas, produtos e serviços podem ser marcas ou marcas de serviços de terceiros.

Glossário

O glossário contém os termos-*chave* e seus significados como são utilizados na documentação do MQSeries Adapter Kernel.

Se um determinado conceito ou termo aparecer em apenas uma seção, é possível que ele não tenha sido incluído no glossário. Ele poderá, no entanto, ser encontrado no "Índice Remissivo" na página 137.

O glossário não inclui os termos para outros produtos da IBM como o MQSeries.

adaptador. A saída do MQSeries Adapter Builder. Normalmente, o usuário cria cada adaptador para ser específico de *um tipo de mensagem* enviada de ou para um aplicativo. Assim, os adaptadores não fazem parte do Pacote do MQSeries Adapter. Um adaptador é composto de um código fonte C compilado para uma biblioteca compartilhada. Quando os adaptadores e o MQSeries Adapter Kernel são executados juntos, eles realizam a funcionalidade em tempo de execução do Pacote do MQSeries Adapter. Dependendo de como foi modelado pelo usuário no MQSeries Adapter Builder, o adaptador pode conter uma ampla variedade de funcionalidades, como fluxo de controle, fluxo de dados; navegação seqüencial; ramificação condicional incluindo a decisão e iteração; digitação de dados, armazenamento do contexto de dados, transformação dos elemento de dados, controle transacional, operações lógicas e códigos personalizados. Você pode reutilizar os adaptadores criados.

Consulte "tipo de mensagem" na página 135, "aplicativo de origem" na página 132 e "aplicativo de destino" na página 132.

adaptador de destino. Um adaptador que executa as seguintes tarefas:

- Recebe uma mensagem (do kernel e do MQSeries ou de outro software de mensagens) que foi enviada por um adaptador de origem.
- Processa a mensagem de integração de acordo como o adaptador foi modelado.
- Transforma a mensagem de integração em uma mensagem formatada específica do aplicativo que o aplicativo de destino pode receber.
- Envia a mensagem ao aplicativo de destino utilizando uma interface específica de aplicativo.
- Permite que o operador saiba quando terminou de enviar a mensagem ao aplicativo de destino, para permitir que ele envie uma confirmação.

Se o aplicativo de destino puder receber a mensagem de integração, o adaptador de destino possivelmente não será requerido.

Para cada tipo de mensagem, há um adaptador de destino. Normalmente, um aplicativo de destino pode aceitar múltiplos tipos de mensagens; na maioria dos casos, portanto, um aplicativo de destino é suportado por diversos adaptadores de destino. Consulte "adaptador".

adaptador de origem. Um adaptador que executa as seguintes tarefas:

- Aceita ou, ao contrário, obtém dados estruturados de um aplicativo de origem (normalmente utilizando uma interface específica de aplicativo desenvolvida fora do adaptador).
- Processa os dados estruturados de acordo com o modo como o adaptador foi modelado.
- Transforma os dados estruturados em um formato de mensagem de integração.

- Utilizando o kernel, coloca se a mensagem em uma fila de mensagens, para entrega a um ou mais adaptadores de destino e, então, ao aplicativo de destino.

Para cada tipo de mensagem, há um adaptador de origem. Normalmente, um aplicativo de origem pode enviar múltiplos tipos de mensagens; portanto, na maioria dos casos, um aplicativo de origem é suportado por diversos adaptadores de origem.

Consulte “adaptador” na página 131.

adaptador de serviço Java. Um tipo de adaptador de linguagem Java que em um ambiente do Interceptador JMS, fornece as funções de um daemon do adaptador, operador e adaptador de destino.

adaptador nativo. Software utilizado para enviar e receber os objetos de retenção de mensagens

adaptador nativo do MQSeries Adapter Kernel. Sinônimo de adaptador nativo.

aplicativo de destino. Um programa requerido para receber dados através de uma rede de computadores de um programa (conhecido como aplicativo de origem) que, normalmente, reside em outro computador.

aplicativo de origem. Um programa que é exigido para enviar os dados através de uma rede de computadores para um programa (conhecido como aplicativo de destino) que, normalmente, reside em outro computador.

arquivo aqmconfig.xml. Consulte “arquivo de configuração”.

arquivo aqmsetup. Consulte “arquivo de configuração”.

arquivo de configuração. O arquivo `aqmconfig.xml`, que contém a maioria dos valores de configuração do kernel. Consulte “O arquivo de configuração” na página 66 para obter detalhes.

arquivo de configuração. Um arquivo que contém diversas configurações iniciais do kernel. O nome do arquivo padrão é `aqmsetup`.

bean operador de mensagem. Um bean corporativo que desempenha a função de um operador do WebSphere Application Server é utilizado ao lado de destino do kernel.

BOD. Documentos de Objeto Comercial. Uma representação de um processo comercial padrão que flui dentro de uma organização ou entre organizações. Entre os exemplos estão incluir pedido de compra, mostrar disponibilidade do produto e incluir pedido de vendas. OS BODs são definidos pelo OAG utilizando XML. Consulte “OAG” na página 134 e “XML” na página 135.

Os BODs podem ser utilizados pelo Pacote do MQSeries Adapter para definir corpos de mensagem e suas mensagens de integração.

categoria do corpo. Dados contidos em uma mensagem que representam o tipo de aplicativo da mensagem, por exemplo, OAG ou RosettaNet. Pertencem ao conjunto de valores de controle da mensagem. Consulte “valores do controle de mensagem” na página 135.

A categoria do corpo também ajuda a especificar o tipo de mensagem. Consulte “tipo de mensagem” na página 135.

classe de início de sessão. Uma classe Java específica de cada aplicativo de destino e que pode ser utilizada para ajudar a entregar a mensagem ao aplicativo de destino. A classe de início de sessão é requerida apenas quando o adaptador de destino tiver que iniciar sessão no aplicativo de destino antes de entregar a mensagem. Cada classe de início de sessão é gravada pelo usuário. O operador instancia a classe de início de sessão. A classe de início de sessão procura no arquivo de configuração os valores que o adaptador de destino precisa para suportar a interface específica do aplicativo para o aplicativo de destino. Normalmente, esses valores são parâmetros de início de sessão. Assim, os valores estarão disponíveis para o adaptador de destino.

Uma classe de início de sessão fictícia sem atuação é fornecida com o kernel.

cliente de rastreamento. Um componente do kernel que grava mensagens de rastreamento.

daemon do adaptador. Software executável que faz parte do kernel. O daemon do adaptador é utilizado apenas no modelo push de entrega. Seu objetivo é instanciar os operadores. Depois de iniciado, o daemon do adaptador permanece ativo. Para cada aplicativo de destino, pode existir um ou mais daemons de adaptador.

Em alguns casos, o daemon do adaptador realiza a função de um aplicativo de destino. Ele realiza a funcionalidade requerida, por exemplo, utiliza um adaptador de destino para enviar uma mensagem de e-mail ou para gravar um registro em um arquivo.

DTD. Definição de Tipo de Documento. Em XML, geralmente, um arquivo (ou vários arquivos utilizados juntos) que contém uma definição formal de um tipo de documento em particular. Ele especifica os nomes que podem ser utilizados para os elementos dentro da DTD, onde os elementos podem ocorrer dentro da DTD e como os elementos se adequam em conjunto. No Pacote do MQSeries Adapter, você pode utilizar DTDs para definir corpos de mensagens. Consulte "XML" na página 135 e "mensagem de integração" na página 134.

fila de erros. Na terminologia do Pacote do MQSeries Adapter, uma fila de mensagens que é utilizada quando uma mensagem obtida de uma fila de recepção não pode ser processada.

fila de recepção. Na terminologia do Pacote do MQSeries Adapter, uma fila de mensagens utilizada como a principal fila de entrada para receber mensagens. Podem existir múltiplas filas de recepção por aplicativo de destino, mas apenas uma fila de recepção para cada combinação de identificador de aplicativo, categoria do corpo e tipo do corpo.

fila de resposta. Uma fila de mensagens utilizada para receber respostas. É utilizada com o método `sendRequestResponse` do kernel.

formato de aplicativo neutro. Consulte "mensagem de integração" na página 134.

identificador do aplicativo de dependência. Nome do aplicativo que o operador atende. O operador obtém o identificador do aplicativo de dependência do arquivo de configuração, com base no nome do daemon do adaptador.

identificador lógico de destino. Um valor que representa o aplicativo de destino associado a um adaptador de destino. Consulte "identificador lógico de destino" e "identificador lógico do aplicativo".

Identificador lógico de destino. Um valor que representa o aplicativo de destino. É utilizado, com outros valores de controle de mensagem, através do kernel encaminha e organiza as mensagens. Consulte "valores do controle de mensagem" na página 135.

identificador lógico de origem. Um valor que representa o aplicativo de origem. É utilizado, com outros valores de controle de mensagem, através do kernel encaminha e organiza as mensagens. Consulte "valores do controle de mensagem" na página 135, "identificador lógico do aplicativo" e "identificador lógico de destino".

identificador lógico de resposta. Identificador lógico do aplicativo ao qual respostas devem ser enviadas quando solicitadas. Assume o padrão do identificador lógico de origem na mensagem.

identificador lógico do aplicativo. Um identificador que representa o aplicativo ao qual um adaptador (um adaptador de origem ou um adaptador de destino) está associada. Consulte "identificador lógico de origem" e "identificador lógico de destino".

Interceptador JMS. Um componente fornecido pelo produto WebSphere Business Integrator que ativa a integração compacta entre o MQSeries Adapter Kernel e o WebSphere Application Server Advanced Edition.

interface específica do aplicativo. Uma interface desenvolvida fora do Pacote do MQSeries Adapter com um dos seguintes objetivos:

- Permitir que o adaptador de origem adquira uma mensagem do aplicativo de origem.
- Permitir que o aplicativo de destino adquira uma mensagem do adaptador de destino.

kernel. Sinônimo de MQSeries Adapter Kernel.

lado de destino do kernel. Parte da funcionalidade do kernel que começa quando a mensagem é obtida de uma fila de mensagens e que termina quando a mensagem é enviada para o adaptador de destino.

lado de origem do kernel. Parte da funcionalidade do kernel que começa quando a mensagem é recebida do adaptador de origem e que termina quando a mensagem é colocada em uma fila de mensagens.

mensagem. No MQSeries, incluindo o Pacote do MQSeries Adapter, um conjunto de dados é enviado por um programa e destina-se a outro programa.

mensagem de comunicação. Quaisquer informações específicas do transporte de comunicação, mais o objeto de retenção de mensagens, convertidos em um formato de mensagem específico do transporte de comunicação utilizado

mensagem de integração. Uma mensagem consistindo em dados do aplicativo em um formato neutro do aplicativo para integração. Um exemplo é um documento XML que o adaptador de origem transforma a partir do formato do aplicativo de origem para XML.

mensagens de rastreamento. Mensagens que contêm o estado de processamento de uma mensagem em um determinado ponto no kernel. Você pode utilizar as mensagens de rastreamento para ajudar a diagnosticar problemas com o kernel ou com os adaptadores.

Consulte “rastreamento” na página 135.

modelo de entrega pull. Consulte “modelos de entrega”.

modelo push de entrega. Consulte “modelos de entrega”.

modelos de entrega. Existem dois modelos pelos quais o kernel interage com o aplicativo de destino. Esses modelos são:

push O kernel é responsável pelo início e gerenciamento da entrega de mensagens ao aplicativo de destino. Normalmente, esse modelo não requer a alteração do aplicativo de destino para suportar o Pacote do MQSeries Adapter.

pull O aplicativo de destino é responsável pelo gerenciamento da entrega de mensagens. Esse modelo requer a alteração do aplicativo de destino para suportar o Pacote do MQSeries Adapter. O aplicativo de destino deve gerenciar a interface do kernel para o aplicativo de destino.

modo de comunicação. Modo utilizado pelo kernel para transportar a mensagem e executar serviços intermediários.

MQSeries Adapter Builder. Software que permite a um usuário criar de forma virtual um adaptador para qualquer aplicativo utilizando uma interface gráfica com o usuário (GUI).

MQSeries Adapter Kernel. Um conjunto de APIs e vários programas executáveis, em C e Java, e vários arquivos de configuração. O kernel funciona com os adaptadores e as suporta. Consulte “adaptador” na página 131. Além de suportar diretamente os adaptadores, o kernel realiza funções relacionadas, entre as mais importantes: roteamento de mensagens e serviços de infra-estrutura, como criação de mensagens, rastreamento e interface com o MQSeries ou outros softwares de mensagens.

OAG. Grupo de Aplicativos Abertos (Open Applications Group). Um consórcio industrial sem fins lucrativos que consiste em acionistas importantes na arena de interoperabilidade de componentes de softwares comerciais. O OAG

define os Documentos de Objeto Comercial (BODs - Business Object Documents).

objeto de retenção de mensagens. Um contêiner para meta-dados utilizado pelo kernel para encapsular uma mensagem de integração e outros dados de controle.

operador . Software que faz parte do kernel. O operador é utilizado apenas no modo de entrega push. O daemon do adaptador inicia e cria os operadores. Cada operador gerencia um adaptador nativo. O operador entrega cada mensagem para o adaptador de destino adequado.

Pacote do MQSeries Adapter. Um conjunto de produtos de integração de aplicativo que consiste no MQSeries Adapter Builder e no MQSeries Adapter Kernel.

rastreamento. Um conjunto de processos que o kernel utiliza para gravar mensagens de rastreamento. Consulte "mensagens de rastreamento" na página 134.

serviço de mensagem lógica. Um componente utilizado pelo adaptador nativo para converter as mensagens de transporte através do transporte de comunicação.

tipo de corpo. Dados contidos em uma mensagem que representam o objetivo específico da mensagem, por exemplo, incluir um pedido de vendas ou sincronizar o inventário. Pertencem ao conjunto de valores de controle da mensagem. Consulte "valores do controle de mensagem".

O tipo de corpo também ajuda a especificar o tipo de mensagem. Consulte "tipo de mensagem".

tipo de mensagem. Uma mensagem que é especificada por uma combinação exclusiva de categoria do corpo e tipo do corpo. Consulte "categoria do corpo" na página 132 e "tipo de corpo".

transação. Um conjunto de operações que devem ser executadas como uma unidade de trabalho indivisível. Se todas as operações que compõem uma transação forem bem-sucedidas, a transação será consolidada; ou seja, todas as operações serão executadas. Se uma ou mais operações que compõem uma transação falharem, a transação será retornada; ou seja, nenhuma das operações será executada.

valores do controle de mensagem. Um termo coletivo para um conjunto de valores nas mensagens (corpo e cabeçalhos) e no arquivo de configuração que o kernel utiliza para controlar o enfileiramento e o roteamento de mensagens, e que cada adaptador utiliza para controlar, parcialmente, como executa sua funcionalidade.

WebSphere Application Server Advanced Edition. Um software da IBM que ativa o uso da especificação da Sun Microsystems Enterprise JavaBeans (EJB). O WebSphere Application Server Advanced Edition inclui o servidor EJB, no qual o beans pode ser executado. Encapsular os beans corporativos do negócio lógico e os dados utilizados e compartilhados pelos clientes EJB. Existem dois tipos de beans corporativos: beans de sessão, que encapsulam a operação breve, tarefas específicas do cliente e objetos; beans de entidade que encapsulam os dados persistentes. Um tipo de bean de mensagem do operador de uma chamada do bean de sessão pode ser utilizado no lado de destino do MQSeries Adapter Kernel.

XML. Extensible Markup Language. Um padrão W3C para a representação de dados.

Índice Remissivo

W

X

Impresso em Brazil

G517-7169-05