WebSphere MQ Everyplace V2.0.2

Índice

Desenvolvendo um Aplicativo Básico

Este tópico contém as informações de que você precisa para criar um aplicativo simples do MQe. Ele apresenta o toolkit de desenvolvimento do MQe e explica o que você precisa fazer para configurar seu ambiente de desenvolvimento. O tutorial fornece instruções passo a passo sobre como criar um aplicativo simples do MQe e confirmar se está funcionando.

Um aplicativo simples de exemplo chamado HelloWorld também é descrito. Esse aplicativo simples demonstra como utilizar alguns dos recursos do MQe. Por fim, o tópico apresenta algumas das ferramentas que você pode utilizar para desenvolver e administrar os aplicativos do MQe.

Introdução ao Kit de Desenvolvimento do WebSphere MQe

Este tópico apresenta o Kit de Desenvolvimento do WebSphere MQe, que é um ambiente de desenvolvimento para gravação de aplicativos de sistemas de mensagens e de enfileiramento baseados em Java e C. Para obter informações sobre a disponibilidade de kits de desenvolvimento para ambientes diferentes de Java e C, consulte o Web site do WebSphere MQ no endereço:

`http://www.ibm.com/software/ts/mqseries`

A parte de código do kit de desenvolvimento em Java é dividida em duas seções:

Classes Básicas do WebSphere MQ Everyplace
: Um conjunto de classes Java que fornece todas as funções necessárias para a construção de aplicativos de sistemas de mensagens e de enfileiramento.

Exemplos
: Código-fonte Java e classes que demonstram como utilizar os diversos recursos do MQe.

A parte de código do kit de desenvolvimento em C também se divide em duas seções:

Funções Básicas do WebSphere MQ Everyplace
: Código C que fornece todas as funções necessárias para a construção de aplicativos de sistemas de mensagens e de enfileiramento.

Exemplos
: Código-fonte C que demonstra como utilizar os diversos recursos do MQe.

Configurando Seu Ambiente de Desenvolvimento

Este tópico fornece informações sobre a configuração do seu ambiente de desenvolvimento em Java e C.

Desenvolvimento em Java

Para desenvolver programas em Java utilizando o kit de desenvolvimento do MQe, configure o ambiente Java como se segue:

- Configure o *CLASSPATH* de modo que o JDK (Java Development Kit) possa localizar as classes do MQe.

  Windows

  Em ambientes Windows, utilizando um JDK padrão, você pode usar o seguinte:

  ```
  Set CLASSPATH=<Dir_de_Instalação_do_MQe>\Java;%CLASSPATH%
  ```

  UNIX Em ambientes UNIX você pode usar o seguinte:

```
CLASSPATH=<Dir_de_Instalação_do_MQe>/Java:$CLASSPATH
export CLASSPATH
```

Você pode utilizar diversos ambientes de desenvolvimento Java e ambientes de tempo de execução Java diferentes com o MQe. A configuração do sistema para ambos, desenvolvimento e tempo de execução, é dependente do ambiente utilizado. O MQe inclui um arquivo que mostra como configurar um ambiente de desenvolvimento para diferentes kits de desenvolvimento Java. Em sistemas Windows, é um arquivo em batch denominado JavaEnv.bat, para sistemas UNIX, é um script shell denominado JavaEnv. Para utilizar esse arquivo, copie-o e modifique a cópia para corresponder ao ambiente da máquina na qual pretende utilizá-lo.

Um conjunto de arquivos em batch e scripts shell que executam alguns dos exemplos do MQe utilizam o arquivo de ambiente descrito acima e, se você pretende utilizar os arquivos em batch de exemplo, deve modificar o arquivo de ambiente da seguinte maneira:

- Configure a variável de ambiente *JDK* como o diretório básico do JDK.
- Configure a variável de ambiente *JavaCmd* como o comando utilizado para executar aplicativos Java.
- Se o MQ Classes para Java estiver instalado, configure a variável de ambiente *MQDIR* como o diretório básico do MQ Classes para Java.

Nota: Versões customizadas do JavaEnv.bat ou do JavaEnv podem ser sobrescritas se você reinstalar o MQe.

Ao chamar JavaEnv.bat no Windows, você precisa transmitir um parâmetro que determina o tipo de kit de desenvolvimento Java a ser utilizado.

Os valores possíveis são:

Sun - Sun

JB - Borland JBuilder

MS - Microsoft

IBM - IBM

Nota: Esses parâmetros são sensíveis a maiúsculas e minúsculas e devem ser digitados exatamente como mostrado.

Se você não transmitir um parâmetro, o padrão será IBM.

O script shell JavaEnv no UNIX não utiliza um parâmetro correspondente.

No Windows, por padrão, você precisa executar o JavaEnv.bat a partir do diretório <Dir_de_Instalação_do_MQe>\java\demo\Windows. No UNIX, por padrão, você precisa executar o JavaEnv a partir do diretório <Dir_de_Instalação_do_MQe>/Java/demo/UNIX. Ambos os arquivos podem ser modificados para permitir que sejam executados a partir de outros diretórios ou para utilizar outros kits de desenvolvimento Java.

Ambiente J2ME

Há dois ambientes J2ME distintos:

CDC (Connected Device Configuration) e Profile
: Um exemplo é Foundation + Applications no ambiente CDC, que pode ser efetivamente desenvolvido como um aplicativo Java 2 J2SE (Platform Standard Edition) normal. A única alteração necessária é modificar a opção bootclasspath para apontar para o jar CDC pertinente ou para o arquivo de classe zip.

Nota: A opção 'bootclasspath' pode não estar disponível em todas as JVMs.

CLDC (Connected Limited Device Configuration) e MIDP (Mobile Information Device Profile)
Os aplicativos desenvolvidos para MIDP também podem ser compilados utilizando uma JVM J2SE normal (mais uma vez, utilizando o bootclasspath para apontar para a biblioteca de classe Midp necessária), mas em geral têm que ser executados dentro de um emulador de Midp. Por isso, recomendamos desenvolver o aplicativo utilizando um dos Toolkits MIDP disponíveis na Web. O MQe fornece um jar MIDP que deve ser utilizado nesse ambiente. O `MQeMidp.jar` está localizado no diretório <Dir_de_Instalação_do_MQe>\Java\Jars.

Desenvolvimento em C

Para desenvolver programas em C utilizando o MQe Development Kit, serão necessárias as seguintes ferramentas:

Microsoft eMbedded Visual C++ (EVC) Versão 3.0.
Esta ferramenta está incluída no Microsoft eMbedded Visual Tools 3.0, disponível como download gratuito na página da Web da Microsoft:

`http://msdn.microsoft.com/mobile/`

Você precisa utilizar a versão 3.0, uma vez que a versão 4.0 não suporta PocketPC.

Um SDK para a plataforma da sua preferência
O Microsoft eMbedded Visual Tools 3.0 inclui um SDK para o PocketPC 2000. Também é possível fazer o download de um SDK para o PocketPC 2002 na página da Web da Microsoft:

`http://msdn.microsoft.com/mobile/`

Ligações C

Sobre o código-base Ligações C, consulte a Referência de Programação das Ligações C.

C Nativo

Para obter informações gerais, consulte a Referência de Programação em C, especialmente a página *Informações de Compilação*. Entretanto, essa página está ligeiramente desatualizada e este tópico fornece uma atualização.

Para o código-base C Nativo, há suporte disponível para quatro plataformas:

- PocketPC2000
- PocketPC2002
- PocketPC2003
- Windows 32 bits.

Para PocketPC, são fornecidos arquivos binários para o dispositivo e para o emulador disponível no Integrated Development Environment Microsoft eMbedded Visual C++. Os arquivos binários fornecidos para os dispositivos são compilados para processadores ARM.

Arquivos Binários

A raiz dos arquivos binários, assim como a documentação e exemplos, é o diretório C abaixo do diretório em que você instalou o MQe.

Assim, dentro do diretório C, os arquivos estão localizados conforme se segue:

PocketPC2000

ARM

DLLs C\PocketPc2000\arm\bin

LIBs C\PocketPc2000\arm\lib

Emulador

DLLs C\PocketPc2000\x86emulator\bin

LIBs C\PocketPc2000\x86emulator\lib

PocketPC2002

ARM

DLLs C\PocketPc2002\arm\bin

LIBs C\PocketPc2002\arm\lib

Emulador

DLLs C\PocketPc2002\x86emulator\bin

LIBs C\PocketPc2002\x86emulator\lib

PocketPC2003

ARM

DLLs C\PocketPc2003\arm\bin

LIBs C\PocketPc2003\arm\lib

Emulador

DLLs C\PocketPc2003\x86emulator\bin

LIBs C\PocketPc2003\x86emulator\lib

Windows 32 bits

DLLs C\Win32\Native\bin

LIBs C\Win32\Native\lib

Arquivos de Cabeçalho

Os arquivos de cabeçalho são comuns a todas as plataformas Nativas e podem ser localizados no diretório incluir abaixo do diretório de instalação.

MQe_API.h

Este é o arquivo de cabeçalho "raiz". Se ele estiver incluso, todos os arquivos de cabeçalho relevantes estarão incluídos para você.

A fim de garantir que os arquivos e definições estejam incluídos, indique que você está executando o código-base Nativo conforme segue:

```
#definir   NATIVE                  // ou especificá-lo como opção para o compilador
#includir  <published/MQe_API.h>
```

Link

É necessário estabelecer links com as duas bibliotecas a seguir:

HMQ_nativeAPI.lib
// a biblioteca de APIs

HMQ_nativeCnst.lib
// a biblioteca constante estática MQeString

Você precisa incluir ambos os arquivos. Um linker de otimização remove os links para todas as funções e constantes que você não tenha utilizado.

As demais bibliotecas do MQe estão estática e dinamicamente vinculadas à principal biblioteca de APIs e são incluídas conforme a necessidade.

Utilizando o Embedded Visual C++

É possível compilar aplicativos utilizando o IDE (Integrated Development Environment) EVC ou, opcionalmente, por meio da linha de comando. No entanto, deve-se considerar o seguinte:

- Defina a ″Configuração do WCE Ativo″ utilizando a barra de ferramentas de Configuração do WCE. Para tanto, em **Sistema Operacional de Destino**, selecione PocketPC ou PocketPC 2002. Além disso, em Processador de Destino, selecione um dos seguintes:
  - Win32 (WCE x86em) Debug
  - Win32 (WCE x86em) Release
  - Win32 (WCE ARM) Debug
  - Win32 (WCE ARM) Release

  Nota: Algumas das opções do Processador de Destino ou do **Sistema Operacional de Destino** podem não estar disponíveis, dependendo dos SDKs que você instalou.
- Inclua os arquivos de cabeçalho do código-base C Nativo. Eles são compartilhados entre as duas versões do PocketPC e pelas Ligações C. O local do arquivo de cabeçalho é dentro do diretório de instalação, em `incluir`. Se você incluir o arquivo de cabeçalho raiz, MQe_API.h, incluirá todas as funções de que possa precisar. Uma vez que os arquivos de cabeçalho são compartilhados, é necessário definir qual versão do código-base está sendo utilizada, conforme mostrado no exemplo a seguir:

  ```
  #definir NATIVE
  #definir MQE_PLATFORM  PLATFORM_WINCE

  /*Alternativamente, você deve incluir isso nas Definições do Pré-processador
  no diálogo Configurações do Projeto. Inclua o seguinte no início
  da lista*/
  NATIVE,MQE_PLATFORM=PLATFORM_WINCE

  #incluir <published\MQe_API.h>
  ```
- Inclua uma entrada para o diretório de inclusão de nível superior do MQe em ″Diretórios de inclusão adicionais″. Isso varia de acordo com o local onde foi instalado o produto.
- Insira os seguintes nomes de arquivo .lib no diálogo ″Configurações do Projeto″, em **Link** → **Entrada**:
  - HMQ_nativeAPI.lib
  - HMQ_nativeCnst.lib

  Nota: Existem variações desses arquivos em cada release suportado, por exemplo, um para PocketPC 2000 ARM, um para PocketPC 2000 x86em e assim por diante. Para garantir que você utilize a versão correta, forneça o nome de arquivo completo para cada destino construído.

É recomendável desenvolver aplicativos utilizando o emulador do PocketPC ou do PocketPC2002, uma vez que isso normalmente proporciona um ambiente mais rápido de compilação e depuração. Entretanto, os emuladores atuais são emuladores de API, o que significa que eles não emulam hardware ARM. Eles emulam chamadas de API do PocketPC, mas o código ainda é o x86, que está sendo executado em uma máquina virtual x86 na caixa de emulação do PocketPC 2002. Assim, recomendamos que você teste regularmente o aplicativo no dispositivo de destino real, já que muitos problemas, como o alinhamento de bytes, somente se tornam aparentes no dispositivo real.

Nota: Os arquivos binários do emulador do MQe são fornecidos apenas para fins de desenvolvimento e não são adequados para implementação em um ambiente de produção.

Encadeamento

O código-base nativo é projetado para ser reentrante. O código-base real não utiliza encadeamentos, mas isso não impede o uso de vários encadeamentos no aplicativo. Por exemplo, você pode criar um encadeamento do aplicativo para repetidamente chamar `mqeQueueManager_triggerTransmission()`. Se quiser utilizar vários encadeamentos, você não precisa chamar nenhuma API específica.

Embora não seja obrigatório, é recomendável ter um bloco de exceção por encadeamento. Se você utilizar um bloco de exceção compartilhado entre encadeamentos, o bloco de exceção de um encadeamento que falha pode ser sobrescrito pelo bloco de exceção de um encadeamento bem-sucedido.

Nota: Chame `mqeSession_initialize` ou `mqeSession_terminate` apenas uma vez, antes que qualquer encadeamento utilize uma chamada de API do MQe. Para assegurar isso, chame-o no encadeamento principal antes que qualquer encadeamento de aplicativo seja criado. Por exemplo, **não** utilize o seguinte:

```
mqeSession_initialize();
mqeSession_initialize();
mqeSession_terminate();
mqeSession_terminate();
```

Convenções de Chamada

A convenção de chamada para todas as APIs foi explicitamente definida como `_cdec1`. Entretanto, você pode utilizar uma convenção de chamada padrão diferente em seu aplicativo.

Handles e Itens

Um aplicativo precisa de um mecanismo para acessar itens do MQe tais como o gerenciador de filas, campos, cadeias e outros. Os handles utilizam os itens do MQe. O handle aponta para uma área da memória utilizada para armazenar as informações específicas daquela instância do item. Informações de tipo são guardadas para cada item. Desse modo, é preciso ter cuidado para inicializar o handle corretamente.

Para utilizar um handle, você deve inicializá-lo. Faça isso chamando a nova função do item associado a ser utilizado. Por exemplo, para criar uma MQeString, em primeiro lugar chame a função `mqeString_new()` e transmita um ponteiro para `MQeStringHndl` com aquela finalidade. A função `mqeString_new()` aloca memória para a estrutura interna e configura os valores padrão necessários pelo MQeString. Depois de concluída com sucesso, a função devolve o handle, que agora pode ser utilizado em chamadas subseqüentes para as funções do MQeString.

Depois de terminar com um item, é importante chamar a função `free()` do item com o qual o handle está associado. As funções `free()` liberam todos os recursos do sistema utilizados por aquele item. Configurar o handle como `NULO` introduz uma fuga de memória no aplicativo e o sistema pode ficar sem recursos. Para evitar isso, configure o handle como `NULO` depois que ele tiver sido liberado.

Nota: Recomendamos não tentar liberar um handle mais que uma vez, pois isso pode causar resultados imprevisíveis.

Utilize os handles somente com os seus itens associados. Também é necessário inicializá-los e liberá-los da maneira correta. A única instância em que o aplicativo não é responsável pela inicialização do handle é quando um ponteiro para um handle é transmitido como parâmetro de entrada para uma API do MQe. Em instâncias como essa, um handle completamente inicializado é retornado para o aplicativo sem que o usuário tenha que chamar a função new() pertinente. Um exemplo disso é o `mqeQueueManager_BrowseMessages()`, que possui um ponteiro para um `MQeVectorHndl` como um parâmetro de entrada. No entanto, em casos como esse, o aplicativo ainda é responsável pela liberação do handle.

Funções de Memória do MQe

O MQe fornece as seguintes funções para gerenciamento de memória:

- `mqeMemory_allocate`
- `mqeMemory_free`
- `mqeMemory_reallocate`

Essas funções utilizam as mesmas rotinas de gerenciamento de memória utilizadas no código-base do MQe. Estas estão disponíveis para utilização pelos programas aplicativos. Um aplicativo pode, em geral, utilizar sua própria escolha de gerenciamento de memória. No entanto, algumas chamadas de API, por exemplo, `mqeAdministrator_QueueManager_inquire`, precisam retornar blocos de memória contendo informações. Nesse caso, a memória deve ser liberada utilizando a função `mqeMemory_free`.

Uma vantagem adicional de utilizar as funções do mqeMemory é que seu uso é rastreado junto com o processamento do MQe. Porém, jamais misture as chamadas de alocação de memória. Por exemplo, não libere a alocação de memória com `mqeMemory_allocate` com a chamada `free()` de tempo de execução em C, pois o aplicativo pode se tornar instável.

MQeString

A classe MQeString contém cadeias definidas pelo usuário e de sistema. É uma abstração de cadeia de caracteres utilizada em toda a API C quando uma cadeia é requerida. A MQeString permite que você crie uma cadeia em diversos formatos, como matrizes contendo pontos de código Unicode, com cada ponto de código armazenado em espaço de memória de 1, 2 ou 4 bytes e UTF-8. A implementação atual da MQeString suporta apenas formatos externos.

Nota: Embora sejam transmitidas utilizando uma MQeString, algumas chamadas de API exigem que a cadeia real permaneça dentro do intervalo ASCII válido.

Cadeias Constantes

São fornecidas diversas cadeias constantes. Elas estão definidas nos seguintes arquivos de cabeçalho:

- MQe_Admin_Constants.h
- MQe_Adapter_Constants.h
- MQe_Attribute_Constants.h
- MQe_Connection_Constants.h
- MQe_MQe_Constants.h
- MQe_MQeMessage_Constants.h
- MQe_Queue_Constants.h
- MQe_Registry_Constants.h

Constructor

```
MQERETURN osaMQeString_new(MQeExceptBlock* pExceptBlock,
                    MQEVOID*        pInputBuffer,
                    MQETYPEOFSTRING type,
                    MQeStringHndl * phNewString
                         );
```

Essa função cria um novo objeto MQeString a partir de um buffer contendo dados de caracteres. Os dados podem estar em diversos formatos suportados, incluindo matrizes de caracteres de byte único com terminação nula (isto é, cadeias char* de C normal), matrizes de caracteres Unicode de byte duplo com terminação nula, matrizes de caracteres Unicode de quatro bytes com terminação nula e matrizes UTF-8 com terminação nula. O parâmetro de tipo diz para a função em que formato está o buffer.

Destructor

```
MQERETURN osaMQeString_delete(MQeExceptBlock* pExceptBlock,
                              MQeString_*     pString
                             );
```

Essa função destrói um objeto MQeString que foi criado utilizando osaMQeString_new, MQeString_duplicate ou MQeString_getMQeSubstring.

Getter

```
MQERETURN osaMQeString_get(MQeExceptBlock*      pExceptBlock,
                       MQEVOID*             pOutputBuffer,
                       MQEINT32*            pBufferLength,
                       MQETYPEOFSTRING      requiredType,
                       MQECONST MQeStringHndl hString
                       );
```

Essa função alimenta um buffer de caracteres com o conteúdo de uma conversão de desempenho MQeString onde for necessário. São executadas somente as conversões simples. Nenhuma conversão de página de códigos é tentada. Por exemplo, se uma cadeia SBCS foi inserida na cadeia, tentar extrair os dados como dados DBCS (Unicode) funciona corretamente. Se os dados foram inseridos como DBCS, entretanto, e você tentar extraí-los como SBCS, isso somente funcionará caso os dados não tenham qualquer valor que não pode ser representado com um único byte. Quando get() é utilizado para SBCS, DBCS ou QBCS, cada caractere é representado por seu valor de ponto de código Unicode.

```
MQERETURN osaMQeString_getSubstring(MQeExceptBlock* pExceptBlock,
                  MQEVOID*        pOutputBuffer,
                  MQEINT32*       pBufferLength,
                  MQETYPEOFSTRING requiredType,
                  MQECONST MQeStringHndl hString,
                  MQEINT32 from,
                  MQEINT32 to
                  );
```

Essa função é muito semelhante à osaMQeString_get, exceto pelo fato de que ela recebe somente uma subcadeia (de **from** até **to**, inclusive).

```
MQERETURN osaMQeString_getMQeSubstring(MQeExceptBlock* pExceptBlock,
                  MQeStringHndl *      phOutput,
                  MQECONST MQeStringHndl   hString,
                  MQEINT32 from,
                  MQEINT32 to
                  );
```

Essa função é muito semelhante à osaMQeString_getSubstring, exceto pelo fato de que retorna seu resultado como uma MQeString.

```
MQERETURN osaMQeString_duplicate(MQeExceptBlock  * pExceptBlock,
                  MQeStringHndl * phNewString,
                  MQECONST MQeStringHndl   hString
                  );
```

Essa função duplica uma MQeString.

```
MQERETURN osaMQeString_codePointSize(MQeExceptBlock* pExceptBlock,
                  MQEINT32 * pSize,
                  MQECONST MQeStringHndl   hString
                  );
```

Essa função determina o tamanho de memória (em bytes) requerido pelo maior caractere da cadeia.

```
MQERETURN osaMQeString_getCharLocation( MQeExceptBlock* pExceptBlock,
                  MQEINT32*       pOutIndex,
                  MQECONST MQeStringHndl    hString,
```

```
MQECHAR32       charToFind,
MQEINT32        startFrom,
MQEBOOL         searchForward
);
```

Essa função retorna o índice de local (começando de 0) da primeira aparência de um caractere especificado, especificado como seu valor de ponto de código Unicode. Você pode especificar o ponto de partida da sua procura e a direção da procura.

Tester

```
MQERETURN osaMQeString_isAsciiOnly(MQeExceptBlock*     pExceptBlock,
                        MQEBOOL*            pIsAsciiOnly,
                        MQECONST MQeString_* pString
                        );
```

Essa função determina se a cadeia contém algum caractere ASCII não invariante.

```
MQERETURN osaMQeString_equalTo(MQeExceptBlock*   pExceptBlock,
                        MQEBOOL*          pIsEqual,
                        MQECONST MQeString_* pString,
                        MQECONST MQeString_* pEqualToString
                        );
```

Essa função determina se duas cadeias são equivalentes.

```
MQERETURN osaMQeString_isNull(MQeExceptBlock  * pExceptBlock,
                        MQEBOOL * pIsNull,
                        MQECONST MQeStringHndl  hString
                        );
```

Essa função determina se uma cadeia é uma cadeia nula. Um handle NULO é considerado também uma cadeia nula.

O SBCS (Single Byte Character Set) é o modo padrão de operação com C em uma página de códigos ASCII. Java funciona somente em Unicode e pode haver plataformas de suporte que não carreguem uma página de códigos SBCS, por exemplo, em alguns países, os idiomas são representados em DBCS. Como não inclui o ponteiro de caracteres, o item da cadeia permite que você crie cadeias em uma máquina ASCII sem considerar os requisitos do Unicode. O MQe executa qualquer conversão necessária. Utilize a representação UTF-8 da cadeia, que pode lidar com qualquer representação de caractere e fazer a conversão para você. Depois de criada, uma MQeString não pode ser alterada. No entanto, diversas funções facilitam o uso do tipo da MQeString. Você também pode criar MQeStrings constantes do mesmo modo que usa `#definir NAME "mystring"`. A utilização da MQeString garante a portabilidade do aplicativo.

Passo a Passo: Criando um Aplicativo Básico

Este tópico contém instruções passo a passo para a criação de um aplicativo simples do MQe. Ele descreve as etapas a percorrer para criar e configurar o seu primeiro gerenciador de filas e depois confirmar se ele é capaz de enviar e receber mensagens de outro gerenciador de filas.

Além de descrever o que você deve fazer, ele também lhe diz quais comandos do MQe_Script podem ser utilizados para desempenhar cada tarefa com simplicidade. O MQe_Script utiliza padrões para diversos atributos, os quais, de outro modo, você teria que especificar caso fosse escrever um código equivalente.

O MQe_Script está disponível como parte do Server Support SupportPac no Web site da IBM. Esse SupportPac fornece a documentação completa sobre o uso de todos os comandos do MQe_Script, incluindo detalhes sobre os padrões e explicações sobre como alterá-los, se necessário.

Você também pode realizar várias das etapas envolvidas nesse processo utilizando o MQe_Explorer, incluído no mesmo SupportPac.

Finalmente, o tutorial fornece links para pedaços do código de exemplo, mostrando como desempenhar várias das etapas de maneira programática.

Depois que um gerenciador de filas foi criado e iniciado, toda a configuração (incluindo a criação de filas, definições de conexão, definições de filas remotas e listeners) é realizada por meio de mensagens de administração.

1. Criar um Gerenciador de Filas (QM1)

Ao criar um gerenciador de filas, é necessário definir os seguintes atributos:

- Nome do gerenciador de filas
- Registro público ou privado
- Localização do registro
- Adaptador do armazenamento de mensagens
- Filas padrão
  - AdminQ
  - AdminReplyQ
  - DeadLetterQ
  - System.default.local.Q

Também é possível configurar outros atributos (opcionais) nesse momento, inclusive uma descrição, tempo-limite do canal, nome da regra do atributo do canal e regra do gerenciador de filas, mas estes não estão incluídos neste tutorial.

Criando o QM1 Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para criar um gerenciador de filas denominado QM1:

```
mqe_script_qm -create -qmname QM1
```

Esse comando cria um gerenciador de filas denominado QM1, com as seguintes características:

- Registro público
- Localização básica em C:\arquivos de programas\mqe\java\mqe_script. O registro padrão e os diretórios das filas estão localizados em subdiretórios nesse caminho
- Utiliza o armazenamento de mensagem padrão e salva as informações em disco
- Contém 4 filas padrão

Um arquivo ini também é criado de modo que as informações do gerenciador de filas sejam salvas e possam ser iniciadas novamente transmitindo-se a localização desse arquivo para um método apropriado

2. Iniciar o Gerenciador de Filas (QM1)

Depois de criar o gerenciador de filas denominado QM1, você precisa iniciá-lo.

Iniciando o QM1 Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para iniciar o gerenciador de filas denominado QM1:

```
mqe_script_qm -load
```

Quando nenhum nome é fornecido, esse comando inicia o gerenciador de filas que acaba de ser criado. Se você quiser saber como carregar um gerenciador de filas e especificar o arquivo INI, consulte a documentação fornecida com o MQe_Script.

3. Criar uma Fila Local (Q1)

Depois de iniciar o gerenciador de filas QM1, você pode criar uma fila local denominada Q1:

Criando o Q1 Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para criar uma fila local denominada Q1:

```
mqe_script_appq -create -qname Q1
```

Esse comando cria uma fila local básica (também chamada de *fila de aplicativos*) denominada Q1, no gerenciador de filas QM1.

4. Criar uma Definição de Conexão

Depois de criar sua fila local (Q1), é necessário criar uma definição de conexão, especificando o seguinte:

- O nome do gerenciador de filas ao qual deseja conectar-se (o gerenciador de filas remotas)
- A porta na qual o gerenciador de filas remotas estará monitorando
- O adaptador de comunicações.

Criando uma Definição de Conexão Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para criar uma definição de conexão:

```
mqe_script_condef -create -cdname QM2 -port 1881
```

Esse comando cria uma definição de conexão para um gerenciador de filas denominado QM2, que está monitorando na porta 1881. Não é necessário que o QM2 exista quando a conexão é criada, mas ele deve existir quando você tentar enviar uma mensagem para uma fila remota daquele gerenciador de filas. Como nenhum adaptador está especificado, o adaptador Http é utilizado por padrão.

5. Criar uma Definição de Fila Remota

Depois de criar uma definição de conexão, é necessário criar uma definição remota de uma fila local no gerenciador de filas QM2.

Criando uma Definição de Fila Remota Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para criar uma definição de fila remota:

```
mqe_script_sproxyq -create -qname Q2 -destination QM2
```

Esse comando cria uma fila de proxy síncrona, que é uma definição remota de uma fila local no QM2. Não é necessário que o QM2 exista quando a definição de fila remota é criada. Entretanto, você deve criar uma definição de conexão (consulte “4. Criar uma Definição de Conexão”) antes que possa criar esta definição de fila remota.

6. Criar um Listener (L1)

Depois de criar uma definição de fila remota, é necessário criar um listener.

Criando um Listener Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para criar um listener denominado L1 (no gerenciador de filas QM1):

```
mqe_script_listen -create -listenname L1 -port 1882
```

Cria um listener para o gerenciador de filas QM1 e monitora na porta 1882. O adaptador de comunicações padrão, adaptador Http, é utilizado.

7. Iniciar o Listener (L1)

Depois de criar um listener, é necessário iniciá-lo.

Iniciando um Listener Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para iniciar o listener L1:

```
mqe_script_listen -start -listenname L1
```

8. Criar um Segundo Gerenciador de Filas (QM2)

Depois que terminar de configurar o QM1 (conforme mostrado nas etapas anteriores deste tutorial), você precisa criar um segundo gerenciador de filas denominado QM2:

Criando o QM2 Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para criar um gerenciador de filas denominado QM2:

```
mqe_script_qm -create -qmname QM2
```

Esse comando cria um gerenciador de filas denominado QM2, com as seguintes características:

- Registro público
- Localização básica em C:\arquivos de programas\mqe\java\mqe_script. O registro padrão e os diretórios das filas estão localizados em subdiretórios nesse caminho
- Utiliza o armazenamento de mensagem padrão e salva as informações em disco
- Contém 4 filas padrão

Um arquivo ini também é criado de modo que as informações do gerenciador de filas sejam salvas e possam ser iniciadas novamente transmitindo-se a localização desse arquivo para um método apropriado

9. Iniciar o QM2

Depois de criar o gerenciador de filas denominado QM2, é necessário iniciá-lo.

Iniciando o QM2 Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para iniciar o gerenciador de filas denominado QM2:

```
mqe_script_qm -load
```

Quando nenhum nome é fornecido, esse comando inicia o gerenciador de filas que acaba de ser criado. Se você quiser saber como carregar um gerenciador de filas e especificar o arquivo INI, consulte a documentação fornecida com o MQe_Script.

10. Criar uma Fila Local (no QM2) Denominada Q2

Depois de iniciar o gerenciador de filas QM2, você pode criar uma fila local denominada Q2.

Criando o Q2 Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para criar uma fila local denominada Q2:

```
mqe_script_appq -create -qname Q2
```

Esse comando cria uma fila local básica denominada Q2, no gerenciador de filas QM2.

11. Criar uma Definição de Conexão (no QM2)

Depois de criar sua fila local (Q2), é necessário criar uma definição de conexão, especificando o seguinte:

- O nome do gerenciador de filas ao qual deseja conectar-se (o gerenciador de filas remotas)
- A porta na qual o gerenciador de filas remotas estará monitorando
- O adaptador de comunicações.

Criando uma Definição de Conexão Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para criar uma definição de conexão:

```
mqe_script_condef -create -cdname QM1 -port 1882
```

Esse comando cria uma definição de conexão para um gerenciador de filas denominado QM1, que está monitorando na porta 1882. Não é necessário que o QM1 exista quando a conexão é criada, mas ele deve existir quando você tentar enviar uma mensagem para uma fila remota daquele gerenciador de filas. Como nenhum adaptador está especificado, o adaptador Http é utilizado por padrão.

12. Criar uma Definição de Fila Remota (no QM2)

Depois de criar uma definição de conexão, é necessário criar uma definição remota de uma fila local no gerenciador de filas QM1.

Criando uma Definição de Fila Remota Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para criar uma definição de fila remota:

```
mqe_script_sproxyq -create -qname Q1 -destination QM1
```

Esse comando cria uma fila de proxy síncrona, que é uma definição remota de uma fila local no QM1. Não é necessário que o QM1 exista quando a definição de fila remota é criada, mas ele deve existir antes que uma mensagem seja enviada para ele.

13. Criar um Listener (no QM2) Denominado L2

Depois de criar uma definição de fila remota, é necessário criar um listener.

Criando um Listener Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para criar um listener denominado L2 (no gerenciador de filas QM2):

```
mqe_script_listen -create -listenname L2 -port 1881
```

Cria um listener para o gerenciador de filas QM2 e monitora na porta 1881. O adaptador de comunicações padrão, adaptador Http, é utilizado.

14. Iniciar o Listener L2 (no QM2)

Depois de criar um listener, é necessário iniciá-lo.

Iniciando um Listener Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para iniciar o listener L2:

```
mqe_script_listen -start -listenname L2
```

15. Enviar (PUT) uma Mensagem do QM1 para o QM2

Agora que você criou e iniciou os dois gerenciadores de filas, criou suas filas e definições de conexão e criou e iniciou seus listeners, já pode enviar mensagens entre os dois gerenciadores de filas.

Enviando uma Mensagem Utilizando o MQe_Script:

No QM1, utilize o seguinte comando do MQe_Script para enviar uma mensagem do QM1 para o QM2:

```
mqe_script_msg -put -qname Q2 -qmname QM2
```

Esse comando envia uma mensagem para a fila Q2 do gerenciador de filas QM2.

16. Receber (GET) a Mensagem no QM2

Depois que a mensagem foi enviada para a fila do QM1, você pode receber a mensagem do QM2.

Recebendo uma Mensagem Utilizando o MQe_Script:

Utilize o seguinte comando do MQe_Script para receber a mensagem da fila:

```
mqe_script_msg -get -qname Q2 -qmname QM2
```

17. Exibindo Detalhes de Objetos do MQe

É possível exibir detalhes dos objetos do MQe que você criou emitindo o comando do MQe_Script `inquireall`. Por exemplo, para obter informações sobre o gerenciador de filas locais, utilize o seguinte comando:

```
mqe_script_qm -inquireall
```

Isso fará com que sejam exibidas todas as informações sobre o gerenciador de filas locais e mostrado quaisquer padrões que o MQe_Script tenha utilizado.

Também é possível exibir informações sobre outros objetos, especificando seu nome. Por exemplo:

```
mqe_script_condef -inquireall -cdname QM2
```

Aplicativo MQe de Exemplo (HelloWorld)

Esse tópico descreve como criar um aplicativo básico (denominado HelloWorld) utilizando as APIs Java e C do MQe. Ele contém informações sobre como projetar, desenvolver, implementar e executar o aplicativo.

″HelloWorld″ em Java

Esta seção descreve como projetar, desenvolver, implementar e executar um aplicativo básico ″HelloWorld″ em Java.

Projetando o Aplicativo Java

A finalidade desse aplicativo é criar e utilizar um único gerenciador de filas com uma fila local. Isso envolve enviar uma mensagem para a fila local e depois removê-la.

Você pode criar gerenciadores de filas para serem utilizados por um programa. Depois que esse programa estiver concluído, você pode executar um segundo programa que restabeleça a configuração anterior do gerenciador de filas.

Em geral, a configuração de novas entidades é um processo independente da sua utilização real. Após a configuração, a administração dessas entidades também requer um processo diferente de sua utilização. Esta seção focaliza mais a utilização do que a administração.

Presumindo-se que a entidade do gerenciador de filas já tenha sido configurada, o aplicativo HelloWorld tem o seguinte fluxo para ambos os códigos-base, C e Java:

1. **Iniciar o gerenciador de filas** Inicia o gerenciador de filas com base em informações já criadas
2. **Criar uma mensagem** Cria uma estrutura que você pode utilizar para enviar uma mensagem de um gerenciador de filas para outro
3. **Enviar para uma fila local** Envia a mensagem para a fila local
4. **Receber de uma fila local** Recupera a mensagem da fila local e verifica se a mensagem é válida
5. **Encerrar** Limpa e encerra o gerenciador de filas

Desenvolvendo o Aplicativo Java

O código a seguir está na classe `examples.helloworld.Run` em seu estado completo. As soluções que utilizam as classes do MQe são freqüentemente divididas em várias tarefas independentes:

- Instalação da solução
- Configuração do gerenciador de filas, deixando as informações de configuração no disco rígido local
- Utilização do gerenciador de filas
- Remoção do gerenciador de filas
- Desinstalação da solução

Antes de ler as informações contidas neste capítulo, é necessário configurar um gerenciador de filas. O programa `examples.helloworld.Configure` demonstra a configuração do gerenciador de filas. O programa `examples.helloworld.Unconfigure` demonstra a remoção do gerenciador de filas. Esta seção da documentação descreve como utilizar o gerenciador de filas.

Visão Geral de examples.helloworld.run:

O método principal controla o fluxo do aplicativo HelloWorld. A partir desse código, é possível ver que o gerenciador de filas foi iniciado, que uma mensagem foi enviada para uma fila, que uma mensagem foi recebida em uma fila e que o gerenciador de filas foi encerrado.

As informações de rastreio podem ser redirecionadas para o fluxo de saída padrão se a constante simbólica `MQE_TRACE_ON` teve seu valor alterado para '`verdadeiro`'.

```
public static void main(String[] args) {
        try {
            Run me = new Run();

            if (MQE_TRACE_ON) {
              me.traceOn();
            }
            me.start();
            me.put();
            me.get();
```

```
            me.stop();
              if (MQE_TRACE_ON) {
                me.traceOff();
              }
          } catch (Erro de exceção) {
              System.err.println("Erro: " + error.toString());
              error.printStackTrace();
          }
      }
```

Iniciar o Gerenciador de Filas:

O programa `examples.helloworld.Configure` cria uma imagem do gerenciador de filas `HelloWorldQM` em disco.

Antes que um gerenciador de filas possa ser utilizado, deve ser instanciado na memória e iniciado. O método de inicialização no programa de exemplo faz isso.

```
public void start() throws Exception {

       System.out.println("Iniciando o gerenciador de filas.");

         String queueManagerName = "HelloWorldQM";
         String baseDirectoryName =
           "./QueueManagers/" + queueManagerName;

         // Criar todas as informações de configuração
       necessárias para a construção do
         // gerenciador de filas na memória.
         MQeFields config = new MQeFields();

        // Construir os parâmetros da seção do gerenciador de filas.
         MQeFields queueManagerSection = new MQeFields();

         queueManagerSection.putAscii(MQeQueueManager.Name,
               queueManagerName);
         config.putFields(MQeQueueManager.QueueManager,
               queueManagerSection);

          // Construir os parâmetros da seção do registro.
          // Nestes exemplos, utilizamos um registro público.
         MQeFields registrySection = new MQeFields();

        registrySection.putAscii(MQeRegistry.Adapter,
           "com.ibm.mqe.adapters.MQeDiskFieldsAdapter");
         registrySection.putAscii(MQeRegistry.DirName,
               baseDirectoryName + "/Registro");

         config.putFields("Registro", registrySection);

         System.out.println("Iniciando o gerenciador de filas");
         myQueueManager = new MQeQueueManager();
         myQueueManager.activate(config);
         System.out.println("Gerenciador de filas iniciado.");
      }
```

Para iniciar o gerenciador de filas, você precisa conhecer, no mínimo, seu nome, localização e o adaptador que deve ser utilizado para ler as informações de configuração do gerenciador de filas em seu registro.

A ativação do gerenciador de filas faz com que os dados de configuração do disco sejam lidos utilizando o adaptador de campos do disco, e o gerenciador de filas estará então iniciado e em execução, disponível para utilização.

Criar uma Mensagem e Enviá-la para uma Fila Local:

O código a seguir constrói uma mensagem, inclui um campo Unicode com um valor do ″Hello World!″ e, em seguida, a mensagem é enviada para SYSTEM.DEFAULT.LOCAL.QUEUE no gerenciador de filas HelloWorldQM local.

```
public void put() throws Exception {
        System.out.println("Enviando a mensagem de teste");
        MQeMsgObject msg = new MQeMsgObject();

    // Incluir meu texto do hello world na mensagem.
        msg.putUnicode("myFieldName" , "Hello World!");

        myQueueManager.putMessage(queueManagerName,
        MQe.System_Default_Queue_Name, msg, null, 0L);
        System.out.println("Enviar a mensagem de teste");
    }
```

Receber Mensagem de uma Fila Local: O código a seguir recebe a primeira mensagem da fila local, SYSTEM.DEFAULT.LOCAL.QUEUE, verifica se foi recebida uma mensagem com o campo myFieldName e exibe o texto contido no campo Unicode.

```
public void get() throws Exception {
        System.out.println("Recebendo a mensagem de teste.");
        MQeMsgObject msg = myQueueManager.getMessage( queueManagerName,
                                                      MQe.System_Default_Queue_Name,
                                                      null, null, 0L );

        if (msg != null) {
            System.out.println("Mensagem de teste recebida.");

            if (msg.contains("myFieldName")) {
                String textGot = msg.getUnicode("myFieldName");

                System.out.println("Mensagem continha o texto '" + textGot + "'");
            }
        }
    }
```

Parando e Excluindo o Gerenciador de Filas:

Esta seção descreve como encerrar um gerenciador de filas e excluir a definição do gerenciador de filas.

Parando o Gerenciador de Filas

Você pode encerrar o gerenciador de filas utilizando um encerramento controlado.

```
public void stop() throws Exception {
        System.out.println("Parando o gerenciador de filas.");
        myQueueManager.closeQuiesce(QUIESCE_TIME);
        myQueueManager = null;
        System.out.println("Gerenciador de filas parado.");
    }
```

Excluindo a Definição do Gerenciador de Filas do Disco

Você pode utilizar o programa examples.helloworld.Unconfigure para remover o gerenciador de filas do disco.

Executando o Aplicativo Java

Em um prompt de comandos, configure seu caminho de classe para referência aos arquivos de classe do MQe. Estes estão disponíveis no diretório Java em que você instalou o produto MQe.

Certifique-se de que o seu shell tenha a capacidade de criar e modificar o `diretório ./QueueManagers` no sistema. Se não tiver, altere a origem dos `programas examples.helloworld`, de modo que se refiram a um diretório acessível, e recompile o código java.

Chame o Programa de Configuração a fim de criar o gerenciador de filas. A sintaxe depende da JVM (Java Virtual Machine) utilizada. A JVM da IBM é chamada utilizando o comando ″java″, por exemplo, `java examples.helloworld.Configure`. Isso cria o gerenciador de filas no disco.

Execute o programa hello world `java examples.helloworld.Run`. Isso envia uma mensagem para uma fila local, recebe a mensagem de volta e a exibe parcialmente.

Você já pode destruir o gerenciador de filas no disco utilizando `java examples.helloworld.Unconfigure`.

″HelloWorld″ em C

Esta seção descreve como projetar, desenvolver, implementar e executar um aplicativo ″HelloWorld″ em C.

Projetando o Aplicativo em C

A finalidade desse aplicativo é criar e utilizar um único gerenciador de filas com uma fila local. Isso envolve enviar uma mensagem para a fila local e depois removê-la.

Você pode criar gerenciadores de filas para serem utilizados por um programa. Depois que esse programa estiver concluído, você pode executar um segundo programa que restabeleça a configuração anterior do gerenciador de filas.

Em geral, a configuração de novas entidades é um processo independente da sua utilização real. Após a configuração, a administração dessas entidades também requer um processo diferente de sua utilização. Esta seção focaliza mais a utilização do que a administração.

Presumindo-se que a entidade do gerenciador de filas já tenha sido configurada, o aplicativo HelloWorld tem o seguinte fluxo para ambos os códigos-base, C e Java:

1. **Iniciar o gerenciador de filas** Inicia o gerenciador de filas com base em informações já criadas
2. **Criar uma mensagem** Cria uma estrutura que você pode utilizar para enviar uma mensagem de um gerenciador de filas para outro
3. **Enviar para uma fila local** Envia a mensagem para a fila local
4. **Receber de uma fila local** Recupera a mensagem da fila local e verifica se a mensagem é válida
5. **Encerrar** Limpa e encerra o gerenciador de filas

Nota: O código-base C não possui uma função equivalente à função Coleta de Lixo Java. Assim, limpar o gerenciador de filas representa mais em C.

Desenvolvendo o Aplicativo em C

Esta seção abrange a codificação de alto nível requerida para o aplicativo ″HelloWorld″ em C.

O código dos exemplos a seguir está no exemplo HelloWorld_Runtime.c em seu estado completo. O exemplo contém um código para manipular as especificidades da execução de um programa em um computador de bolso, que envolve principalmente gravar um arquivo para lidar com a falta de opções de linha de comandos. Utilize a função de exibição para gravar um arquivo, conforme mostrado nos exemplos contidos nas seções a seguir.

Visão Geral do HelloWorld_Runtime.c:

É necessário incluir apenas um arquivo de cabeçalho para acessar as APIs. Inclua a definição NATIVE para indicar que não se trata das Ligações C. Defina também em que MQE_PLATFORM pretende executar o aplicativo.

```
#definir   NATIVE
#definir   MQE_PLATFORM = PLATFORM_WINCE
#incluir<published/MQe_API.h>
```

Todo o código, incluindo as declarações variáveis, encontram-se dentro do método principal. É necessário ter estruturas para verificação de erros. A estrutura MQeExceptBlock é transmitida a todas as funções para receber de volta as informações de erro. Além disso, todas as funções retornam um código indicando êxito ou falha, que é armazenado em cache em uma variável local:

```
    /* ... Sinalizador de retorno local */
     MQERETURN            rc;
     MQeExceptBlock       exceptBlock;
```

Crie diversas cadeias, por exemplo, para o nome do gerenciador de filas:

```
    MQeStringHndl     hLocalQMName;

           ...

     if ( MQERETURN_OK == rc ) {
        rc = mqeString_newUtf8(&exceptBlock,
                &hLocalQMName,
                "LocalQM");
    }
```

A primeira chamada de API feita é inicialização de sessão:

```
    /* ... Inicializar a sessão */
    rc = mqeSession_initialize(&exceptBlock);
```

Iniciar o Gerenciador de Filas:

Esse processo envolve duas etapas:

1. Criar o item do gerenciador de filas.
2. Iniciar o gerenciador de filas.

Criar o gerenciador de filas exige dois conjuntos de parâmetros, um para o gerenciador de filas e outro para o registro. Ambos os conjuntos de parâmetros são inicializados. O *armazenamento de filas* e o registro necessitam de diretórios.

Nota: Todas as chamadas requerem um ponteiro para ExceptBlock e um ponteiro para o handle do gerenciador de filas.

```
     if (MQERETURN_OK == rc) {

    MQeQueueManagerParms qmParams  = QMGR_INIT_VAL;
    MQeRegistryParms     regParams = REGISTRY_INIT_VAL;
    qmParams.hQueueStore           = hQueueStore;
    qmParams.opFlags               = QMGR_Q_STORE_OP;

    /* ... criar parâmetros de registro -
        o mínimo requerido */
    regParams.hBaseLocationName     =  hRegistryDir;
    display("Carregando o Gerenciador de Filas a partir do registro \n");
    rc = mqeQueueManager_new(  &exceptBlock,
                            &hQueueManager,
                             hLocalQMName,
                             &qmParams,
                             &regParams);
}
```

Você já pode iniciar o gerenciador de filas e executar as operações do sistema de mensagens:

```
 /* Iniciar o gerenciador de filas  */

if ( MQERETURN_OK == rc ) {
   display("Iniciando o gerenciador de Filas\n");
   rc = mqeQueueManager_start(hQueueManager,
          &exceptBlock);
}
```

Criar uma Mensagem:

Para criar uma mensagem, em primeiro lugar crie um novo objeto de campos. O exemplo a seguir inclui um único campo. Observe que as cadeias do rótulo de campo são transmitidas:

```
MQeFieldsHndl hMsg;

display("Criando uma nova mensagem\n");
rc = mqeFields_new(&exceptBlock,&hMsg);
if ( MQERETURN_OK == rc ) {
  rc = mqeFields_putInt32(hMsg,&exceptBlk,
        hFieldLabel,42);
}
```

Enviar Mensagem para uma Fila Local:

Depois de criar a mensagem, você pode enviá-la para uma fila local utilizando a função *putMessage*. Observe que os nomes da fila e do gerenciador de filas são transmitidos. NULO e 0 são transmitidos para os parâmetros de segurança e entrega garantida, uma vez que não são necessários neste exemplo. Após o envio da mensagem, você pode liberar o objeto do MQeFields:

```
if ( MQERETURN_OK == rc ) {
      display("Enviando uma mensagem \n");
       rc = mqeQueueManager_putMessage(hQueueManager,
                                       &exceptBlock,
                                        hLocalQMName,
                                        hLocalQueueName,
                                       hMsg,
                                        NULL,
                                        0);

     (void) mqeFields_free(hMsg,NULL);
  }
```

Receber Mensagem de uma Fila Local:

Depois de enviar a mensagem para uma fila, é possível recuperá-la e verificá-la. Opções semelhantes são transmitidas para a função getMessage. A diferença é que um ponteiro para o handle de um campo é transmitido. Um novo objeto de Campos é criado, removendo a mensagem da fila:

```
MQeFieldsHndl hReturnedMessage;
display("Recebendo a mensagem de volta \n");

rc = mqeQueueManager_getMessage(hQueueManager,
                      &exceptBlock,
                      &hReturnedMessage,
                      hLocalQMName,
                      hLocalQueueName,
                            NULL,
                            NULL,
                            0);
}
```

Após receber a mensagem, você pode verificá-la para saber o valor que foi inserido. Obtenha isso utilizando a função getInt32. Se o resultado for válido, será possível imprimi-lo.

```
if (MQERETURN_OK == rc) {
  MQEINT32 answer;
   rc = mqeFields_getInt32(hReturnedMessage,
                           &exceptBlock,
                           &answer,
                           hFieldLabel);

if (MQERETURN_OK == rc) {
  display("Resposta é %d\n",answer);
}
else {
   display("\n\n %s (0x%X) %s (0x%X)\n",
       mapReturnCodeName(EC(&exceptBlock)),
       EC(&exceptBlock),
       mapReasonCodeName(ERC(&exceptBlock)),
       ERC(&exceptBlock)  );
  }

}
```

Encerrar:

Em seguida à remoção da mensagem da fila, é possível encerrar e liberar o gerenciador de filas. Você também pode liberar as cadeias que foram criadas. Finalmente, termine a sessão:

```
(void)mqeQueueManager_stop(hQueueManager,&exceptBlock);
(void)mqeQueueManager_free(hQueueManager,&exceptBlock);

/* Limpar */
(void)mqeString_free(hFieldLabel,&exceptBlock);
(void)mqeString_free(hLocalQMName,&exceptBlock);
(void)mqeString_free(hLocalQueueName,&exceptBlock);
(void)mqeString_free(hQueueStore,&exceptBlock);
(void)mqeString_free(hRegistryDir,&exceptBlock);


(void)mqeSession_terminate(&exceptBlock);
```

Compilando:

Para simplificar o processo de compilação, o diretório de exemplos inclui um arquivo pronto. Esse é o arquivo pronto exportado do EVC (eMbedded Visual C). Um arquivo em batch executa esse arquivo pronto. Esse arquivo em batch configurará os caminhos para os diretórios do EVC, juntamente com os caminhos para a instalação do MQe. Talvez seja necessário editar o arquivo em batch, dependendo de como você deseja instalar o MQe.

A execução do arquivo em batch compila o exemplo. Por padrão, o arquivo em batch compila para Depurar PocketPC 2000 (Emulador ou processador ARM).

Implementando o Aplicativo em C

A fim de implementar o aplicativo ″HelloWorld″, é necessário criar um gerenciador de filas. Existem várias maneiras de fazer isso, as quais são abordadas em outra parte deste centro de informações. Neste caso, é utilizado o programa HelloWorld_Admin. Execute-o conforme descrito abaixo.

As instruções a seguir se aplicam tanto ao emulador como a um dispositivo real:

1. Copie todas as DLLs para a raiz do dispositivo. Elas podem ser encontradas nos diretórios do emulador arm ou x86.
2. Construa o código de exemplo utilizando o arquivo pronto fornecido.

   Nota: É necessário compilar os arquivos HelloWorld_Admin.c e HelloWorld_Runtime.c.

3. Copie esses arquivos binários para o dispositivo ou emulador que está executando o PocketPC ou o Emulador.

Executando o Aplicativo em C

Esta seção descreve como executar o aplicativo ″HelloWorld″ em Java e em C, no PocketPC ou no emulador.

Esse exemplo envolve duas etapas:

1. Crie o gerenciador de filas. Para tanto, execute o programa HelloWorld_Admin. A execução desse programa cria a representação de disco persistente do QueueManager.
2. Execute o programa HelloWorld_Runtime. Isso inicia um QueueManager baseado no registro estabelecido. Para verificar se o programa funcionou corretamente, consulte o arquivo de registro que foi gerado. Por padrão, ele estará na raiz do dispositivo.

Utilizando as Ferramentas de Desenvolvimento e Administração do MQe

A seguir estão algumas das ferramentas que podem ser utilizadas para desenvolver ou administrar os aplicativos do MQe:

MQe_Explorer
O MQe_Explorer fornece uma interface gráfica com o usuário para o gerenciamento de uma rede do MQe e sua interconexão com o MQ. Ele permite que os gerenciadores de filas do MQe e seus objetos associados, tais como filas, conexões e pontes, sejam configurados local ou remotamente. O MQe_Explorer também proporciona uma maneira simples de criar gerenciadores de filas locais, que podem então ser configurados para atender às necessidades dos aplicativos. Oferece ainda um ambiente de ativação e depuração para aplicativos do MQe. O MQe_Explorer está disponível como parte do Server Support SupportPac.

MQe_Script
O MQe_Script é uma ferramenta do MQe baseada em linha de comandos MQe, e é independente de plataforma. Ele permite que os gerenciadores de filas do MQe e seus objetos associados como filas, conexões, listeners e objetos de ponte, sejam configurados local ou remotamente. Mensagens de teste também podem ser enviadas para as filas a fim de validar a operação da rede. Assim como o MQe_Explorer, o MQe_Script proporciona uma maneira simples de criar gerenciadores de filas locais, que você pode depois configurar e estender para o uso do seu aplicativo. O MQe_Script está disponível como parte do Server Support SupportPac.

MQe_Service
O MQe_Service é uma ferramenta baseada em assistente para a criação e operação de gerenciador de filas locais do MQe. Adicionalmente, ele ativa a configuração automática do gateway do MQe e dos gerenciadores de filas do MQ, quando é necessária a transmissão de mensagens entre as redes do MQe e do MQ.

Plataforma de Desenvolvimento de Software Rational

Eclipse
Eclipse é uma plataforma suportada pela indústria e baseada em padrão aberto para ferramentas de desenvolvimento de softwares. Ela fornece uma estrutura baseada em plug-in que facilita a criação, integração e utilização de ferramentas de software. Para obter mais informações sobre o Eclipse, visite a página da Web:

`http://www.eclipse.org`

Índice Remissivo