IBM

WebSphere MQ Everyplace V2.0.2

Contents

Java programming samples

Introduction to the set of Java™ examples provided with MQe

This topic provides a brief description of the set of Java programming examples provided with MQe. Each example demonstrates how to use or extend a feature of MQe, and most of them are described more fully in the relevant topics in this information center.

Adapters (examples.adapters)

This package provides two example classes that conform to the MQe adapters specification.

MQeDiskFieldsAdapter
: This example class is identical in functionality to the disk fields adapter found in com.ibm.mqe.adapters. It supports the reading and writing of data on the local file store.

WESAuthenticationGUIAdapter
: Wrappers the WESAuthenticationAdapter found inside com.ibm.mqe.adapters. This example enhances the WESAuthenticationAdapter by displaying a dialog box that prompts the user for login information when connecting to a WebSphere® Everyplace® proxy.

Command line administration (examples.administration.commandline)

This package contains a suite of example tools for creating base MQe objects from the command line. Each program is a simple example of how to send administration messages and how to interpret the replies.

Using these tools and a script, you can reliably set up exactly the same configuration on a number of machines.

GUI administration (examples.administration.console)

This package contains a set of classes that implement a simple graphical user interface (GUI) for managing MQe resources.

Admin
: Front end to the example administration GUI.

Additionally there is a suite of classes that provides the graphical user interface for each MQe managed resource.

The GUI can be invoked in any of the following ways:
- Using the batch file ExamplesAdminConsole.bat
- From the command line:
  ```
  C example
  java examples.administration.console.Admin
  ```
- From a button on the example server `examples.awt.AwtMQeServer`

Simple administration (examples.administration.simple)

This package contains a set of examples that show how to use some of the administrative features of MQe in your programs. As with the application examples, these examples can work with either a local or a remote queue manager.

Example1
: Create and delete a queue.

Example2
: Add a connection definition for a remote queue manager.

Example3
: Inquire on the characteristics of a queue manager and the queues it owns.

ExampleAdminBase
: The base class that all administration examples inherit from.

Interaction with a queue manager (examples.application)

This package contains a set of examples that demonstrate various ways to interact with a queue manager. These include putting a message to and getting a message from a queue. All the examples can be used with either a local queue manager or a remote queue manager. Before you can use any of these applications, the queue managers that are to be used must be created. You can use the CreateExampleQM.bat batch file on Windows®, or the CreateExampleQM shell script on UNIX®, to create queue managers `ExampleQM`.

Example1
: Simple put and get of a message.

Example2
: Put several messages and then get the second one using a match field.

Example3
: Use a message listener to detect when new messages arrive.

Example5
: Lock messages then get, unlock, and delete them.

Example6
: Simple put and get of a message using assured message delivery.

Example7
: Simple put and get of a message through a Websphere Everyplace proxy.

ExampleBase
: The base class that all application examples inherit from.

These examples can be run as follows:

Windows
: Using batch file ExamplesMQeClientTest.bat

```
ExamplesMQeClientTest <JDK> <example no>
    <remoteQMgrName> <localQMgr ini file>
```

UNIX Using shell script ExamplesMQeClientTest

```
ExamplesMQeClientTest  <example no>
<remoteQMgrName> <localQMgr ini file>
```

where

<*JDK*> is the name of the Java environment. The default is IBM®

Note: This parameter is not used on UNIX.

<example no>
: is the number of the example to run (suffix of the name of the example). The default is 1 (Example1).

<remoteQMgrName>
: is the name of the queue manager that the application should work with. This can be the name of the local or a remote queue manager. If it is a remote queue manager, a connection must be configured that defines how the local queue manager can communicate with the remote queue manager.

  By default the local queue manager is used, as defined in ExamplesMQeClient.ini.

<localQMgrIniFile>
: is an ini file containing startup parameters for a local queue manager. By default ExamplesMQeClient.ini is used.

Security (examples.attributes)

This package contains a set of classes that show how to write additional components to extend MQe security. However, they are not designed to be used for asynchronous messaging and do not provide very strong security.

NTAuthenticator
: An authenticator that authenticates a user to the Windows NT® security database. To authenticate correctly the user must have the following User Rights set on the target NT system:

  - Act as part of the operating system
  - Logon locally
  - Logon as a service

  The NT authenticator uses the Java native interface (JNI) to interact with Windows NT security. The code for this can be found in the examples.nativecode directory. The dll built from this code must be placed in the PATH of the NT machine that owns the target resource.

UnixAuthenticator
: An authenticator that authenticates a user using the UNIX password or shadow password system. The UNIX authenticator uses the JNI to interact with the host system. The code for this can be found in the examples.nativecode directory. If your system supports the shadow password file, you must recompile this native code with the `USE_SHADOW` preprocessor flag defined. You must also ensure the code has sufficient privileges to read the shadow password file when it executes. This example does not work if your system uses a distributed logon service (such as Lightweight Directory Access Protocol (LDAP)).

LogonAuthenticator
: Base logon authentication support.

UseridAuthenticator
: Support for base *userID* authentication.

  This example requires a UserIDS.txt file as input. This file must have the format:

```
  [UserIDs]

User1Name=User1Password

...

UserNName=UserNPassword
```

Adding a small GUI to an application (examples.awt)

This package provides a toolkit for building applications that require a small graphical interface. It also contains example applications that provide a graphical front end to MQe functions.

AwtMQeServer
: A graphical front end to the `examples.queuemanager.MQeServer` example. The MQeTraceResourceGUI class provides a resource bundle that contains internationalized strings for use by the GUI. MQeTraceResourceGUI is in package examples.trace.

  You can use the batch file ExamplesAwtMQeServer.bat to run this application.

AwtMQeTrace
: A graphical front end to examples.trace.MQeTrace.

Classes **AwtDialog**, **AwtEvent**, **AwtFormat**, **AwtFrame**, and **AwtOutputStream** provide a toolkit for building small footprint awt-based graphical applications. These classes are used by many of the graphical MQe examples.

Managing mini-certificates (examples.certificates)

This package contains examples for managing mini-certificates.

ListWTLSCertificates
: This example uses methods in the class com.ibm.mqe.attributes.MQeListCertificates to implement a command line program which lists mini-certificates in a registry, to varying levels of detail.

RenewWTLSCertificates
: This example uses methods in the class com.ibm.mqe.registry.MQePrivateRegistryConfigure to implement a command line program which renews mini-certificates in a registry. This should be used only on a private registry.

Logging events (examples.eventlog)

This package contains some examples that demonstrate how to log events to different facilities.

LogToDiskFile
: Write events to a disk file.

LogToNTEventLog
: Write events to the Windows NT event log. This class uses the JNI to interact with the Windows NT event log. The code for this is in the examples.nativecode directory.

LogToUnixEventLog
: Write events to the UNIX event log (which is normally /var/adm/messages). This class uses the JNI to interact with the UNIX event logging system. The code for this can be found in the examples.nativecode directory. The syslog daemon on your system should be configured to report the appropriate events.

Creating and deleting queue managers (examples.install)

This package contains a set of classes for creating and deleting queue managers.

DefineQueueManager
: A GUI that allows the user to select options when creating a queue manager. When the options have been selected, this example creates an ini file containing the queue manager startup parameters, and then creates the queue manager.

CreateQueueManager
A GUI program that requests the name and directory of an ini file that contains queue manager startup parameters. When the name and directory are provided, a queue manager is created.

SimpleCreateQM
A command line program that takes a parameter that is the name of an ini file that contains queue manager startup parameters. It also optionally takes a parameter that is the root directory where queues are stored. Provided a valid ini file is found, a queue manager is created.

DeleteQueueManager
A GUI program that takes the name of an ini file that contains queue manager startup parameters. Provided a valid ini file is found, the queue manager is deleted.

SimpledDeleteQM
A command line program that takes a parameter that is the name of an ini file that contains queue manager startup parameters. Provided a valid ini file is found, the queue manager is deleted.

GetCredentials
A GUI program that takes the name of an ini file that contains queue manager startup parameters. Provided a valid ini file is found, new credentials (private/public key pair and public certificate) are obtained for the queue manager. The mini-certificate server must be running and the request for a new certificate must have been authorized for this to succeed.

All the configuration files use the resources and utilities provided in **ConfigResource**, and **ConfigUtils**.

Extending the MQ bridge (examples.mqbridge.awt)

This package contains a set of classes that show how to use and extend the MQ bridge. Some of the examples extend other MQe examples.

AwtMQBridgeServer
This is an example of a graphical interface for the underlying examples.mqbridge.queuemanager.MQBridgeServer class.

The MQBridgeServer class source code demonstrates how to add bridge functionality to your MQe server program, following these guidelines.

To start the bridge enabled server:

1. Instantiate the base MQe queue manager, and start it running.
2. Instantiate a com.ibm.mqe.mqbridge.MQeMQBridges object, and use its activate() method, passing the same .ini file information as you passed to the base MQe queue manager.

The bridge function is then usable.

To stop the bridge-enabled server:

1. Disable the bridge function by calling the MQeMQBridges.close() method. This stops all the current MQ bridge operations cleanly, and shuts down all the MQ bridge function.
2. Remove your reference to the MQeMQBridges object, allowing it to be garbage-collected.
3. Stop and close the base MQe queue manager.

ExamplesAwtMQBridgeServer.bat
This file provides an example of how to invoke the MQBridgeServer using the Awt server. It also shows how to control the initial settings of the AwtMQBridgeTrace module.

ExamplesAwtMQBridgeServer.ini
This file provides an example configuration file for a queue manager that supports MQ bridge functionality.

Administering objects for an MQ bridge (examples.mqbridge.administration.commandline)

This package contains a suite of example tools, similar to those in the examples.administration.commandline package, designed to administer the objects required for an MQ bridge.

Testing communication between MQ and MQe (examples.mqbridge.application.GetFromMQ)

The example programs in this package are useful for proving that MQe and MQ can communicate with each other. These examples are MQ bindings programs that use the Java classes and are driven by a simple command-line syntax.

GetFromMQ

This class destructively reads any message appearing on a specified MQ queue, and provides timing statistics on when the message arrives. Optionally the message content can be dumped to the standard output screen.

This example is useful when testing a link between MQe and MQ, to see what throughput is being achieved between the two systems. Scripts dealing with connectivity between MQe and MQ can refer to and use this class.

PutFromMQ

This class puts a message to an MQ queue, such that the user can specify the target queue and the target queue manager. It specifically uses the long form of the MQQueueManager.accessQueue() method to make use of any MQe queue manager alias definitions that might be defined on the MQ queue.

MQe interface (examples.midp.mqeexampleapp)

This package contains two example applications to aid your understanding of the MQe interface. The example code can be split into 3 parts:

The message service (example.midp.mqeexampleapp.messageservice)

This runs MQe, controls a queue manager and performs functions such as queue creation and message sending. This is the core of the examples and allows them to be written with minimal calls to the MQe API. This also means that to see the code required to create a local queue for example, a user can simply look at the relevant function within MQeMessageService.

Example 1: The message pump (examples.midp.mqeexampleapp.msgpump)

This is a very simple application consisting of a single server and client. The client is set to send a message to the server every 3 seconds which, when received by the server, will be displayed to the user. Queues are asynchronous. Implementations of the client are available for both MIDP and J2SE, while the server is only available for J2SE.

Example 2: The text application (examples.midp.mqeexampleapp.textapp)

This is slightly more complex than the first example, consisting of 2 servers and a client. When initiating, the client is required to register with the registration server. The registration server adds the client to a store-and-forward queue on the gateway server and replies with a success or failure message. The client can then send user-defined messages to the gateway server (which it will display). The aim of this application is to show how a separate server can be used to create resources necessary for a new client on the system to aid scalability of large MQe networks.

JNI implementation (examples.nativecode)

Several of the examples require access to operating system facilities on Windows NT, or UNIX (AIX® and Solaris). MQe accesses these functions using the JNI. For Windows, the code in the examples\native directory provides the JNI implementation required by `examples.attributes.NTAuthenticator` and `examples.eventlog.LogToNTEventLog`. For UNIX, the code in the file examples/native/JavaUnix.c provides the JNI implementation required by the `examples.attributes.UnixAuthenticator` and `examples.eventlog.LogToUnixEventLog`.

Running a QM as a client, server, or servlet (examples.queuemanager)

A queue manager can run in many different types of environment. This package contains a set of examples that allow a queue manager to run as a client, server, or servlet:

MessageWaiter
: An example of how to wait for messages without using the deprecated waitFormessage method.

MQeClient
: A simple client typically used on a device.

MQePrivateClient
: A client that can be used with secure queues and secure messaging.

MQeServer
: A server that can connect concurrently to multiple queue managers (clients or servers). This is typically used on a server platform. Batch file ExamplesAwtMQeServer.bat can be used to run the `examples.awt.AwtMQeServer` example which provides a graphical front end to this server.

MQePrivateServer
: Similar to MQeServer but allows the use of secure queues and secure messaging.

MQeServlet
: An example that shows how to run a queue manager in a servlet.

MQeChannelTimer
: An example that polls the channel manager so that it can time-out idle channels.

MQeQueueManagerUtils
: A set of helper methods that configure start various MQe components.

Rules classes (examples.rules)

You can control and extend the base MQe functionality using rules. Some components of MQe invoke rules classes. These rules provide a means of changing the functionality of the component. This package contains the following example rules classes:

ExamplesQueueManagerRules
: Example queue manager rules class makes regular attempts to transmit any held messages.

AttributeRule
: Example attribute rule that controls the use of attributes.

Trace handling (examples.trace)

This package contains an example trace handler that can be used for debugging an application during development, and for tracing a completed application.

MQeTrace
: The base MQe trace class.

AwtMQeTrace, which is in the examples.awt package, provides a graphical front end to the MQeTrace class.

MQeTraceResource
A resource bundle that contains trace messages that can be output by MQe.

MQeTraceResourceGUI
This class contains all the translatable text for the trace window controls.

Index