IBM

WebSphere MQ Everyplace V2.0.2

Contents

Chapter 1. Installing and uninstalling MQe 1
Before you install 1
Prerequisites 1
Licensing 4
Installing MQe. 5
Installation procedure 5
Silent installation 6
Accessibility mode command line option 8
Installing from a zip file 8
Installed components 8
Components on the Web 10
Verifying your installation 10
Modifying your installation 11
Uninstalling MQe 11
Windows 11
Unix. 12
Silent uninstallation 12

Chapter 2. Applying maintenance to MQe 15
Migrating from 1.2.7 to 2.0 or 2.0.1 15
Aliases in MQeFields 15
MQeFields. 15
MQeChannel 16
MQeAttribute 16
Deprecated methods and classes 16
Security 16
Communication settings 17

Index 19

Chapter 1. Installing and uninstalling MQe

Before you install

Prerequisites

Supported platforms

You can install MQe on certain server platforms only.

To transfer programs and Java™ classes to other platforms, you must use an appropriate download or file transfer program (not supplied).

Directly supported with installation support: MQe can be installed on the following platforms using platform-specific tools:

- Windows® 2000 Server
- Windows 2000 Professional
- Windows XP Professional
- Windows 2003 Server
- AIX® Version 5.1 and Version 5.2
- Sun Solaris Version 8 and Version 9
- HP-UX Version 11i
- Linux® on Intel® Kernel Version 2.2 and 2.4 (installed using a zip file)
- Linux on zSeries® Kernel Version 2.4

iSeries™ is a supported server platform but does not have native install support. Instead, you must use the QShell supplied as part of the operating system, and then use the UNIX® instructions for installing using a zip file.

- iSeries Version 5.1 and 5.2

For all other platforms, you can install MQe using an appropriate file transfer program (not supplied).

Java environment

The following Java environments are supported for MQe Version 2.0 deployment, runtime and development:

- J2ME CLDC/MIDP*
- J2ME CDC/Foundation*
- WECE jclRM*
- Java 1.1.8**
- J2SE 1.3*

* or greater functionality; ** limited to 4690 only

MQe is only supported on JVMs carrying the ″Java Compatible″ Sun logo or the “Java Powered” Sun logo.

The following operating systems are the reference platforms for Java. If a problem is experienced within a Java environment IBM® may ask for the problem to be recreated on one of the following platforms:

- J2ME, CLDC/MIDP, J2M3 CDC/Foundation, WECE

  - PocketPC 2002 (ARM)
  - PocketPC 2003 (ARM)
- J2ME, CLDC/MIDP, J2M3 CDC/Foundation, WECE, J2SE 1.4.2
  - Windows 2000 Professional
  - Windows XP Professional
  - Linux Intel Kernel 2.4
  - Linux Intel Kernel 2.2
- Java 1.1.8 (4690)
- J2SE 1.4.2
  - Windows 2000 Server
  - Windows 2003 Server
  - AIX 5.1
  - AIX 5.2
  - 4690
  - Solaris 8
  - Solaris 9
  - HPUX11i
  - Linux
  - zSeries V2R4
  - iSeries 5.2
  - iSeries 5.3

To send messages to and receive messages from WebSphere® MQ, the MQ Classes for Java are required.
- These are packaged with MQ Version 5.3 and later versions.
- If you are using an earlier version of MQ, you can download the MQe Classes for Java as SupportPac™ ma88 from the IBM MQ Web site at
  `http://www.ibm.com/software/integration/support/supportpacs/product.html#wmq`

For more information about system requirements and Java implementation, see http://www.ibm.com/software/integration/wmqe/requirements/

J2ME: MQe is compliant with the J2ME technologies:
1. Connected Device Configuration (CDC)
2. Connected Limited Device Configuration (CLDC)

or more explicitly their most popular profiles:
1. CDC/Foundation
2. CLDC/Midp

When deciding which to use, consider the following:
1. Using CDC/Foundation enables the full functionality of MQe, excluding any example code requiring the AWT GUI package.
2. The use of CLDC/Midp, however, restricts the application to solely 'client' side behavior and a limited range of built-in compressors and encryptors.

   Note: Devices that are CLDC/Midp enabled might have severe memory limitations that can preclude the direct use of the MQe core package.

C Bindings environment: The C Bindings APIs require J2SE 1.3 or later and may only be installed and used on:
- Windows 2000 Server
- Windows 2003 Server
- Windows 2000 Professional
- Windows XP Professional

JMX Interface: The MQe Java Management Extensions Interface (MQe JMX) executes as an application running in a Java Virtual Machine (JVM). All it requires is an activated local queue manager. Given this, the interface can then manage the instrumented local queue manager, and the queue manager's resources. It can also manage any remotely activated MQe queue managers (and their resources) for which the local queue manager is able to connect directly to the MQe network.

MQe JMX API has been developed in compliance with the JMX specification v.1.2, and therefore requires a compliant implementation of the JMX specification. If you do not have such an implementation, a JMX reference implementation is provided by Sun:http://java.sun.com/products/JavaManagement/.

C native environment

For general information see *C Programming Reference*, in particular the page *Compilation Information*, however that page is now a little out of date, and this topic provides an update.

For the native C code base, support is provided for five platforms:
- PocketPC2000
- PocketPC2002
- PocketPC2003
- WinCE .NET 4.2
- Windows 32bit.

Support for the WebSphere MQ Everyplace® C DLLs on WinCE .NET 4.2 is provided using the Platform Invocation Services provided by the .NET environment.

For PocketPC, binaries are provided for both the device, and the emulator that is available in the Integrated Development Environment Microsoft® Embedded Visual C++. The binaries provided for the devices are compiled for ARM processors.

Binary files

The root of the binary files, as well as the documentation and examples, is the C directory below the directory where you choose to install MQe.

Then in the C directory, the files are located as follows:

PocketPC2000

ARM

DLLs C\PocketPc2000\arm\bin

LIBs C\PocketPc2000\arm\lib

Emulator

DLLs C\PocketPc2000\x86emulator\bin

LIBs C\PocketPc2000\x86emulator\lib

PocketPC2002

ARM

DLLs C\PocketPc2002\arm\bin

LIBs C\PocketPc2002\arm\lib

Emulator

DLLs C\PocketPc2002\x86emulator\bin

LIBs C\PocketPc2002\x86emulator\lib

PocketPC2003 and WinCE .NET 4.2

ARM

DLLs C\PocketPc2003\arm\bin

LIBs C\PocketPc2003\arm\lib

Emulator

DLLs C\PocketPc2003\x86emulator\bin

LIBs C\PocketPc2003\x86emulator\lib

Windows 32bit

DLLs C\Win32\Native\bin

LIBs C\Win32\Native\lib

Header files

The header files are common to all the Native platforms, and may be found in the include directory below the installation directory.

MQe_API.h

This is the "root" header file. If this is included all relevant header files will be included for you.

In order to ensure the correct files and definitions are included you must indicate you are running the Native code base as follows:

```
#define   NATIVE                   // or specify this as an option to the compiler
#include  <published/MQe_API.h>
```

Linking

You should link against the following two libraries:

HMQ_nativeAPI.lib
// the API library

HMQ_nativeCnst.lib
// the static constant MQeString library

Generally you should include both these files. Then an optimizing linker should remove links to any functions and constants you have not used.

The other MQe libraries are statically and dynamically linked with the main API library and will be included as required.

Licensing

MQe is a toolkit that enables users to write MQe applications and to create an environment in which to run them. Before deploying this product, or applications that use it, please make sure that you have the necessary licenses.

1. The pricing of licenses for use of the Program on **servers** is based on ***Processor License Units***. Use of each copy of the Program on a server requires one *Processor License Unit* to be acquired for each processor or symmetric multiprocessor contained in the server on which the copy of the Program is to

run. Different types of *Processor License Units* and ***Device Use Authorizations*** are required, depending on whether the Program is running on point-of-sale, that is retail, equipment or on another type of computer. Use of the Program on retail equipment requires a ***Retail*** server license, whereas use on other (non-retail) equipment requires a ***Network*** server license.

2. Additional *Device Use Authorization* is required for any use of the Program on a separate **client device**, except those included in the *Network* server license described in 3. below.
3. Each *Network* server license includes authorization for the restricted use of the Program with no more than one hundred (100) client devices, on condition that all such copies are used in the same economic enterprise or organization as the server copy.

Please refer to http://www.ibm.com/software/integration/wmqe/ for details of these restrictions.

Device platform use authorizations, which are recorded on Proof of Entitlement documents and valid to support the use of MQe, are required to use the product (other than for purposes of code development and test) on specified client platforms. These licenses do not entitle the user to use the MQe Bridge, or to run on the server platforms specified in the MQe pricing group lists published by IBM and also available on the Web via the URL mentioned below.

Please refer to http://www.ibm.com/software/integration/mqfamily/ for details of these restrictions.

Installing MQe

The information in this chapter guides you through the installation of MQe on machines that are to be used to develop MQe applications.

The MQe installation program is a Java jar file that has platform-specific launchers, which can be run straight from the product CD. The installation program extracts the working files to a temporary directory, copies the MQe files onto your computer, and cleans up the working files.

Note that, in this release, the application and solution provider is responsible for deploying MQe to pervasive devices.

Installation procedure

The information in this section applies to installation on Windows, AIX, Linux, Solaris, and HP-UX.

At any time during the installation, you can click the **Back** button on a screen to take you back to previous screens and review or change information. To exit the install procedure and cancel the installation, click the **Cancel** button on any screen.

Before you start:

- You are strongly advised to uninstall any previous versions of MQe before installing or reinstalling this new version (see "Uninstalling MQe" on page 11). The installation program does not detect versions of the product prior to Version 1.2.4, and does not display any warnings.
- If you are using **Windows**, check that your user id has administrator access. If it does not, the Start Menu icons for IBM MQe may not appear.
- If you are using **AIX**, you must be logged on as the root user in order to run the installation successfully.

To install MQe:

1. Insert the product CD into your CD-ROM drive.
2. Start the installation either from the platform specific launcher, or from the setup.jar file:
   - **Installing from the platform-specific launcher:**

a. The launchers are held in the platform-specific subdirectories on the product CD. To begin the installation process, run the correct launcher for your platform (for example, setup.exe for Windows).
b. If you copy the launcher to your local target machine, you also need to copy the setup.jar file into the parent directory of the directory into which you have copied the launcher. The launcher cannot be run from a root directory.
c. If you run the installation from a launcher and you see a message box with the text `No matching JVM was found`, the installer was unable to find a Java environment to use. If you see this message you need to use the `setup.jar` file for installation.
d. To tell the launcher to use a specific JVM, use the following flag:

   ```
   setup.exe -is:javahome c:\jdk1.3
   ```

   To tell the launcher to use a specific directory to store temporary files, use the following launcher flag:

   ```
   setup.exe -is:tempdir c:\mytempdir
   ```

- **From the setup.jar file**:

  Change to the product CD directory where the setup.jar is stored, and run the installation program using the Java command on your computer. This command is typically `java`, `jre`, or `jview`. For example:

  On Windows

  ```
  set classpath=.\setup.jar;%classpath%
  jview run
  ```

  (If you use JVM 1.2.2 or higher, you can execute the jar file by double-clicking it.)

  On Linux, AIX, Solaris, and HP-UX

  ```
  CLASSPATH=./setup.jar:$CLASSPATH
  export CLASSPATH
  java run
  ```

3. On the Welcome screen, confirm that you want to install the MQe program by clicking the **Next** button and then follow the prompts to complete your installation.

Silent installation

You can run the installer in silent mode. This means that no panels are displayed during the installation and there are no prompts for input. There are two ways to run the install silently:

From the jar file

Run the installation in the same way as previously described, but append the `-silent` flag. For example:

On Windows

```
set classpath=.\setup.jar;%classpath%
jview run -silent
```

On Linux, AIX, Solaris, and HP-UX

```
CLASSPATH=./setup.jar:$CLASSPATH
export CLASSPATH
java run -silent
```

From a platform specific launcher

Add the `-is:silent -silent` flags. For example:

On Windows

```
setup.exe -is:silent -silent
```

On AIX

```
Setupaix.bin -is:silent -silent
```

On Linux

```
Setuplinux.bin -is:silent -silent
```

On Solaris

```
Setupsolaris.bin  -is:silent -silent
```

On HP-UX

```
Setuphp-ux.bin  -is:silent -silent
```

Silent installation directories

By default the installation program installs MQe in the directories shown in the table below. If you have any old versions of MQe on the computer, a silent install uses your current directory and not the default shown in this table:

Table 1. Default installation directories

Platform	Default installation directory
iSeries	/opt/MQe
AIX	/opt/MQe
Linux	/opt/MQe
Solaris	/opt/MQe
Win32	\Program Files\MQe
HP-UX	/opt/MQe

To set a different installation directory for a silent install, append the `-P MQe.installLocation` flag to the install command as follows:

From the jar file

```
java run -silent -P MQe.installLocation="C:\my new install directory"
```

From a platform specific launcher

```
setup.exe -is:silent -silent -P MQe.installLocation=
"C:\my new install directory"
```

Note: When using the *-P MQe.installLocation* parameter to override the install location during silent install, ensure that you specify the full path of the the new destination.

Silent installation with option files

When running the install silently you can specify an options file. The options file allows you to:

- Set the install to silent
- Change the install location
- Select which features to install

The following example options file sets the install to run silently, sets the install location to C:\MQe, and chooses to install all features.

```
#specify silent install
-silent
#set features to active
-P Java.active=true
-P Documentation.active=true
-P CBindings.active=true
-P Native.active=true
-P Palm.active=false
#Set the install location
-P MQe.installLocation="C:\MQe"
```

Note:

1. Include the `-is:silent` flag in the options file if running the install from a launcher.
2. Do not leave any blank lines in the options.txt file.
3. Start all lines with # or a valid command.
4. You can have multiple commands on a single line.

The following examples show how to run the installer with an options file:

From the jar file

```
java -cp setup.jar run -options C:\options.txt
```

From a launcher

```
setup.exe -options C:\options.txt
```

Accessibility mode command line option

The accessibility mode command line option sets the installation to automatically be in console mode, and provides additional audible information for visually impaired end users. Example usage:

```
java -cp <classpaths> setup.jar run -accessibility
```

This flag is also valid at uninstall time. Example usage:

```
java -cp <classpaths> uninstall.jar run -accessibility
```

Installing from a zip file

The MQe classes are also provided as a zip file. You can use this file to install MQe on devices where the graphical installer is not suitable or not supported. On a UNIX based system (such as Linux and HP-UX) you need to create a folder, copy the appropriate zip file into it, and then use an unzip utility to extract the class files. For example:

```
mkdir mqe
cp /cdrom/unixinst.zip mqe
cd mqe
unzip unixinst.zip
chmod -R +x *
```

Once the class files have been extracted, configure your environment to run MQe programs.

Installed components

You can select various features during the installation of MQe. These features are described in the following topics, along with the components that are installed if the feature is selected.

MQe for Java

MQe Java classes
A set of classes that implement all of the MQe function. Subsets of these classes can be used to provide different MQe configurations such as a subset for a device, or a subset for a server.

Helper classes
A set of classes derived from the base classes that implement some commonly used functions.

Example classes
A set of classes that demonstrate how to use many of the features of MQe. The source code for these classes is also provided.

Utilities
Tools to assist with the programming and administration of MQe.

MQe C bindings

C Bindings specific classes

Includes the package com.ibm.mqe.bindings. These are only required for the C Bindings and do not affect existing functionality.

Header and binary files

C Bindings adds in extra binary files, and header files. These files are fully documented in the *MQe C Bindings Programming Guide*

MQe C Bindings Programming Guide SC34-6280–01

This book contains guidance and procedural information for writing MQe C applications and administering your systems. The filename is `hmq9al_WMQE_C_BindingsProgrammingGuide.pdf`.

See also the html version here C Bindings Programming Reference.

MQe for Native platforms

Header files

The set of header files required are common for all native platforms are also shared with C Bindings.

Binary files

There are DLL and LIB files, built against the following SDKs:

- PocketPC 2002 SDK, ARM processor and PocketPC 2002 Emulator
- PocketPC 2003 SDK, ARM processor and PocketPC 2003 Emulator

Examples

Some examples are also documented in the MQe Native Platforms Programming Reference.

PocketPC Version information tool

The MQeVersion information tool is shipped as part of the native code base. This is an executable which allows you to check the version information for the installed native DLL files. This tool is built for PocketPC 2000, 2002, and 2003.

The program file is in the `C:\tools\Version\PocketPc2000\arm` or `C:\tools\Version\PocketPc2002\arm` directory of the installation. This file needs to be copied to a suitable place on the device. When running, the tool attempts to load all the native DLL files. A file is written to the root of the device with information on the DLL files.

- Run the tool directly from the file explorer or a command line, with no arguments, and the tool loads the DLL files that are found using the standard load path of the device.
- Run the tool at a command line to specify a directory as a parameter. The specification is partly device dependant, but you can opt to use one of the command line interpreters available for the PocketPC device. When a directory is specified, only the DLL files that exist in that directory are checked.

For each DLL file , the tool gives the main version number, a possible EFix level, that is the fourth digit of the version number, and the build ID that this DLL file came from. If there are DLL files that you have chosen not to copy to the device, you can see message reports to this effect.

Note: If you see DLL files from different builds or versions, please double check that you have copied the correct files. Within the installation there are DLLs for the PocketPC emulators as well as the devices. Be careful to copy the correct files. Failure to do so results in errors when trying to execute programs.

Documentation

The manuals shipped in previous versions of MQe have been substantially modified and restructured into this MQe Information Center.

Components on the Web

Typically you will want to install more than just the MQe product in order to develop your applications.

Verifying your installation

This section describes how to run some examples that verify the successful installation of an MQe development kit.

Java installation verification

After you have installed MQe you can use the following procedures to run some examples that determine whether the installation of the development kit was successful.

- Ensure that the Java environment is set up as described in "Java environment" on page 1. When running any of the Windows batch files described in this section, the first parameter of each is the name of the Java development kit to use. If you do not specify a name, the default is IBM.

  Note: The UNIX shell scripts do not have a corresponding parameter.
- Move to the correct directory:

  Windows
  Change to the <MQeInstallDir>\Java\demo\Windows directory.

  UNIX Change to the <MQeInstallDir>/Java/demo/UNIX directory.
- Create a queue manager as follows:

  Windows
  Run the batch file:

  ```
  C example
  CreateExampleQM.bat <JDK>
  ```

  UNIX Run the shell script:

  ```
  C example
  CreateExampleQM
  ```

  to create an example queue manager called `ExampleQM`.

  Part of the creation process sets up directories to hold queue manager configuration information and queues. The example uses a directory called ExampleQM that is relative to the current directory. Within this directory are two other directories:
  - Registry - holds files that contain queue manager configuration data.
  - Queues - for each queue there is a subdirectory to hold the queue's messages. (The directory is not created until the queue is activated.)
- Run a simple application as follows:

  Once you have created a queue manager you can start it and use it in applications. You can use the batch file ExamplesMQeClientTest.bat or the shell script ExamplesMQeClientTest to run some of the simple application examples.

  The batch file runs examples.application.Example1 by default. This example puts a test message to queue manager ExampleQM and then gets the message from the same queue manager. If the two messages match, the application ran successfully.

  There is a set of applications in the examples.application package that demonstrate different features of MQe. You can run these examples as follows:

  Windows
  Pass parameters to the batch files:

  ```
  C example
  ExamplesMQeClientTest <JDK> <ExampleNo>
  ```

  UNIX Pass parameters to the shell scripts:

```
C example
ExamplesMQeClientTest &gt;ExampleNo&lt;
```

where *ExampleNo* is the suffix of the example. This can range from 1 to 6.

- Delete a Queue manager.

  When a queue manager is no longer required you can delete it. To delete the example queue manager `ExampleQM`:

  Windows
  Run the batch file

  ```
  C example
  DeleteExampleQM.bat <JDK>
  ```

  UNIX Run the shell script

  ```
  C example
  DeleteExampleQM
  ```

  Once you have deleted a queue manager you cannot start it.

  Note: The examples use relative directories for ease of set up. You are strongly recommended to use absolute directories for anything other than base development and demonstration. If the current directory is changed, and you are using relative directories, the queue manager can no longer locate its configuration information and queues.

C installation verification

After you have installed MQe you can run some examples from the MQe C Programming Reference to verify your installation.

Modifying your installation

If you want to remove your installation, see "Uninstalling MQe." Note that it is not possible to use *Add/Remove Programs* to modify the options you have installed, only to completely uninstall.

To modify your installation, for example to add a feature or component that you did not select when you previously installed, insert the product CD into your drive, and then follow the instructions in "Installation procedure" on page 5.

Uninstalling MQe

Follow the instructions that relate to your operating system.

Windows

Choose one of the following ways to uninstall MQe from your Windows system:

Using the Windows Control Panel

1. Click **Start**->**Settings**->**Control Panel**.
2. Double-click the **Add/Remove Programs** icon.
3. In the *Add/Remove Programs* dialog box click on **IBM MQe**.
4. Click the **Add/Remove...** button to start the uninstall program.

Follow the on screen prompts or instructions until the program indicates that the uninstall is complete.

Using uninstall.exe

In the <MQe install directory> double-click uninstall.exe, or use the command:

```
<MQe install directory>\Uninst\uninstall.exe
```

where <MQe install directory> is the directory where you installed MQe.

Follow the prompts until the program indicates that the uninstall is complete.

Using uninstall.jar

Use the uninstall.jar file as follows:

```
set classpath=<MQe directory>\Uninst\uninstall.jar;%classpath%
jview run
```

Unix

Use these instructions to uninstall MQe on:

- AIX
- HP-UX
- Linux
- Solaris
- Unix

Choose one of the following ways to uninstall MQe from your system:

Note: On AIX always use one of these methods, ***do not use SMIT*** because it will not remove the product properly.

Note: On Solaris always use one of these methods, ***do not use pkgrm*** because it will not remove the product properly.

Using uninstall.bin

Enter the command:

```
<MQe install directory>/Uninst/uninstall.bin
```

`<MQe install directory>` is the directory where you installed MQe. This defaults to /opt/MQe, but you can change this during the installation procedure.

Follow the prompts until the program indicates that the uninstall is complete.

Using uninstall.jar

Use the following commands to invoke the uninstall.jar:

```
CLASSPATH=<MQe directory>/Uninst/uninstall.jar:$CLASSPATH
export CLASSPATH
java run
```

Follow the prompts until the program indicates that the uninstall is complete.

Silent uninstallation

You can run the uninstaller in silent mode. This means that no panels are displayed during the uninstall and there are no prompts for input. There are two ways to run the uninstall silently:

From the jar file

Run the uninstall in the same way as previously described, but append the `-silent` flag. For example:

On Windows

```
set classpath=<MQe directory>\Uninst\uninstall.jar;%classpath%
Jview run -silent
```

On Linux, AIX, Solaris, and HP-UX

```
CLASSPATH=<MQe directory>/Uninst/uninstall.jar:$CLASSPATH
export CLASSPATH
java run -silent
```

From a platform specific launcher

Add the `-is:silent` -silent flags. For example:

On Windows

```
uninstall.exe -is:silent -silent
```

On Linux, AIX, Solaris, and HP-UX

```
uninstall.bin -is:silent -silent
```

Silent uninstall with options files

When running the uninstall silently you can specify an options file. The options file allows you to:

- Set the uninstall to silent
- Select which features to uninstall

The following example options file sets the uninstall to run silently, and chooses to uninstall all the features except the Documentation feature.

```
#specify silent uninstall
-silent
#set features to active
-P Java.active=true
-P Documentation.active=false-P CBindings.active=true
-P Native.active=true
```

Note:

1. Include the `-is:silent` flag in the options file if running the uninstall from a launcher.
2. Do not leave any blank lines in the options.txt file.
3. Start all lines with `#...`, or a valid command.
4. You can have multiple commands on a single line.

The following examples show how to run the uninstaller with an options file:

From the jar file

```
java -cp uninstall.jar run -options C:\options.txt
```

From a launcher

```
uninstall.exe -options C:\options.txt
```

Chapter 2. Applying maintenance to MQe

Maintenance updates for MQe are shipped as a complete new release.

There are two options when upgrading from one release to another:

Completely uninstall the current level, and install the new level in the same directory

Keep the install package for the current level, in case you want to restore it later.

Keep the existing level and install the new level into a new directory

After installation, check your `classpath` to ensure that the latest level of MQe is being invoked. If installing on Windows, make sure that you give the shortcuts folder for the new install a different name to the existing one.

For more general information on maintenance updates and their availability see the MQ family Web page at http://www.ibm.com/software/integration/mqfamily/.

Migrating from 1.2.7 to 2.0 or 2.0.1

If you are upgrading to Version 2.0 or Version 2.0.1, you need to consider how the changes described in this section will affect your MQe application.

Aliases in MQeFields

In Version 1, the MQeFields structure passed to the MQeQueueManager allowed the specification of the following two aliases:

- `(ascii)AttributeKey_2=com.ibm.mqe.attributes.MQeSharedKey`
- `(ascii)AttributeKey_1=com.ibm.mqe.MQeKey`

Those aliases specified the default class names to use when loading attribute keys, where an attribute key class was not specified.

These values are hard-coded in the Version 2.0 and 2.0.1 code base, and cannot be changed using the alias mechanism. If the values are specified in .ini files, or calls to the MQeQueueManager, they are ignored.

MQeFields

In order to comply with Java 2 Platform Micro Edition's (J2ME) Connected Limited Device Configuration (CLDC) / Mobile Information Device Protocol (MIDP) specification, several methods have been modified or removed from MQeFields:

- The explicit use of the floating point types, `float` and `double`, has been removed.

  For example, you might have used `putFloat("Val1", -1.234)`.

  Under Java platforms that enable the use of `float/double`, this functionality can be mimicked by explicitly converting the data into the equivalent `int` or `long` using the base types `Java Object convert` method.

  In this case, the above method is replaced with `putFloatAsInt("Val1",Float.floatToIntBits(-1.234))`.

  Note: Version 1 applications can retrieve these values as normal.
- Methods `dumpToFile` and `restoreFromFile` have been removed.

  Applications that used these functions must now dump the MQeFields object and write the byte array to the specified file.

- XOR'ing of dumped data has also been removed.

MQeChannel

The

```
com.ibm.mqe.MQeChannel
```

class has been moved and is now known as

```
com.ibm.mqe.communications.MQeChannel
```

Any references to the old class name in administration messages are replaced automatically with the new class name.

MQeAttribute

The following changes have been implemented in relation to MQeAttribute:

- The implementation of the

  ```
  equals()
  ```

  method on MQeAttribute and its subclasses in Version 1.2.7 (and earlier versions), has been renamed as

  ```
  isAcceptable()
  ```
- An MQeAttributeRule now ships with the product.

  You should now extend your attribute rules from this class instead of MQeRule.

  All methods on MQeAttribute and its subclasses, which used to take an MQeRule object as one of its parameters, now take an MQeAttributeRule object instead.

Deprecated methods and classes

The classes listed here have been removed from the product.

We recommend that you update any applications written to make use of these classes to use instead the equivalent function provided in MQe Version 2.0 and Version 2.0.1.

To enable existing applications to be run before being updated, MQe provides the `MQeDeprecated.jar` jar file.

The `MQeDeprecated.jar file` contains the following classes:

- MQeMQBridge.class
- MQeChannelListener.class
- MQeChannelListenerTimer.class
- MQeChannelManager.class
- MQeTraceInterface.class

For more details on replacements for the above classes, refer to the listing for each class in the Java Programming Reference.

Security

The following changes have been made to security:

1. The MQeCL and MQeRandom classes have been replaced with cryptoLite's CL class.
2. The old style mini-certificate support has been withdrawn from Version 2.0 onwards.

These changes have the following implications:

1. The CL class is supplied as a cryptoLite.zip. In order to use MQe security, the zip file must be placed in the Java class path.

2. MQeMiniCertificateServer no longer supports the old style mini-certificate.

Communication settings

In Version 2, the default Channel Timeout has been reduced from 1 hour to 5 minutes. Also, the system properties used to specify certain adapter variables (for example, socket timeouts) have been modified.

This may affect clients running over slow or fragile networks.

Index

A
applying maintenance 15

E
environments, software 1

I
installation
  migration notes 5

M
maintenance, applying 15
migration 5

O
operating systems, supported 1

P
prerequisite information 5
prerequisites 1

R
required operating systems 1

S
software environments 1
supported operating systems 1

U
upgrading 15

V
verifying your installation
  C 10
  Java 10