WebSphere MQ Telemetry Java Classes Version 1.0

27 February, 2003

SupportPac author
Ian Harwood
Jonathan Woodford

ian_harwood@uk.ibm.com

jonathanw@uk.ibm.com

Take Note!

Before using this report be sure to read the general information under "Notices".

First Edition, February 2003

Table of Contents

Notices

Trademarks and service marks

Summary of Amendments

Date	Changes
27 February 2003	Initial release

Preface

This SupportPac provides a set of Java classes that implement the WebSphere MQ Telemetry transport protocol (formerly known as MQIsdp). The classes provide a clean API that can be used to quickly WebSphere MQ Telemetry transport enable Java applications.

Bibliography

- *WebSphere MQ Event Broker 2.1 Programming Guide, IBM Corporation, SC34-6095-00*
- *WebSphere MQ Event Broker 2.1 Introduction and Planning, IBM Corporation , GC34-6088-00*

Chapter 1. Introduction

This SupportPac provides a Java implementation of the client side of the WebSphere MQ Telemetry transport publish/subscribe protocol. The API is encapsulated in one class, which contains verbs such as connect, publish, subscribe and unsubscribe for communicating with WebSphere MQ Integrator.

The code with this SupportPac is packaged in three different ways to suit different operating environments. The SupportPac contains three subdirectories called J2SE, OSGi and J2ME_MIDP-2.0, which contain code as follows:

- **J2SE**
  MQIsdp.jar contains the API implementation as documented below packaged for the J2SE environment.
  MQIsdpTraceFormat.jar contains trace formatting utilities. This jar file does not need to be present at runtime. It is only required on the machine on which the trace is to be formatted.
  MQIsdpSample.jar contains a sample WebSphere MQ Telemetry transport application which has a Swing user interface. The source code for this application is supplied in package com.ibm.MQIsdp.utility

- **J2ME_MIDP-2.0**
  MQIsdp.jar contains the API implementation as documented below for the J2ME MIDP-2.0 environment.
  MQIsdpSample.jar contains a sample MIDlet application that provides a J2ME user interface. The source code for this MIDlet is supplied in package com.ibm.MQIsdp.midpapp

- **OSGi**
  This contains the J2SE classes packaged with a manifest file that enables the code to be deployed onto the OSGi Service Platform. . The OSGi bundle does not implement any services. It simply repackages the J2SE classes. See http://www.osgi.org for more information about OSGi.

Chapter 2. ClientMQIsdp Java class and the programming model

The WebSphere MQ Telemetry transport protocol is accessible via a single class called com.ibm.MQIsdp.ClientMQIsdp.class, which is in the MQIsdp jar file. This class provides methods for interfacing to WebSphere MQ Integrator such as publish, subscribe and unsubscribe. There are also methods for setting attributes of the WebSphere MQ Telemetry transport connection, such as timeouts and retries. There is also a callback interface so that an application can be notified when events occur such as a publication arriving or a publication send completing.

The ClientMQIsdp class provides a default set of methods for the callback interface, which do nothing. An application can extend this class and override the default callback methods to provide functionality as appropriate.

Any references to WebSphere MQ Integrator broker include the following products:

WebSphere MQ Event Broker V2.1

WebSphere MQ Integrator Broker V2.1

WebSphere MQ Integrator V2.1

Programming model

After instantiating the ClientMQIsdp class the application can set any session parameters as described in section ‘Session methods’. The application should then call one of the connect methods to establish a connection with the WebSphere MQ Integrator broker.

There is no limit on the number of times an application may connect and disconnect, but each instance of the class can only have one connection at a time.

When an application has finished using the ClientMQIsdp class it should call the terminate() method to shut down all threads started by the class. After the terminate method has been called no API methods may be used. A new instance of the class must be instantiated before a new connection can be established.

Callback methods may be invoked whilst the application is running. A complete list is defined in section ‘Callback methods’**Error! Reference source not found.**. Two important methods that should be implemented are publishArrived and connectionLost.

- publishArrived must be implemented if the application needs to receive publications.
- connectionLost must be implemented to handle the MQIsdp connection breaking. Typically the connectionLost method should invoke the connect method to reconnect. If the cleanstart flag is true then the connectionLost method should resubscribe for any topics the application is interested in.

To implement the callback methods a class may extend the ClientMQIsdp class and override the default methods, or it may implement one of the callback interfaces and register itself as the class that will handle the callback events.

Chapter 3. Com.ibm.MQIsdp.ClientMQIsdp.java

Constructor

```
public ClientMQIsdp(String server, int port);
```

Instantiate the ClientMQIsdp class with an IP address and port number.

server - IP address or hostname of the WebSphere MQ Integrator broker e.g. 127.0.0.1

port - IP port number for the SCADAInput node

Publish / Subscribe methods

Connect

```
public int connect( String clientid,
                    boolean cleanstart,
                    short keepalive) throws Exception
```

```
public int connect( String clientid,
                     boolean cleanstart,
                     short keepalive,
                     String willtopic,
                     int     willQoS,
                     String willmsg,
                     boolean willretain) throws Exception
```

Two connect methods are provided. The second one allows a Last Will and Testament message to be specified, which will be delivered in the event of the connection unexpectedly terminating.

Parameters are:

clientid – A string of up to 23 characters that uniquely identifies the application to the WebSphere MQ Integrator broker.
cleanstart – A boolean indicating if the broker should delete subscriptions and outstanding publications if the application terminates unexpectedly.
keepalive – An interval in seconds. If the application does not communication with the broker within this interval, then the broker assumes the application has terminated. This class automatically keeps the connection alive whilst it is running.
willtopic – Topic on which a will message will be published in the event of the MQIsdp connection application terminating unexpectedly.
willQoS – The quality of service at which the will message should be published in the event of the MQIsdp connection terminating unexpectedly.
willmsg – Message that will be published on the will topic in the event of the MQIsdp connection terminating unexpectedly.
willretain – A boolean indicating whether the will message should be published as a retained publication or not.

Returns:

-1	A socket error
0	Successful connect
1	Unacceptable protocol version
2	Client identifier rejected

3   : Broker unavailable

Disconnect

```
public void disconnect() throws Exception
```

Disconnect from the WebSphere MQ Integrator broker. Publications can no longer be sent or received.

Ping

```
public void ping() throws Exception
```

Send a ping message to the WebSphere MQ Integrator broker. An application does not need to call this method. Pinging the connection to keep it alive is automatically handled by the underlying Java classes.

Publish

```
public int publish( String topic,
                    byte[] message,
                    int QoS,
                    boolean retained ) throws Exception
```

Publish data on a specified topic to the WebSphere MQ Integrator broker.

Parameters are:

topic – A hierarchical topic.
message – Application specific data
QoS – The quality of service the MQIsdp protocol should use to deliver the message.
0 – At most once
1 – At least once
2 – Exactly once
retained - A boolean flag indicating whether the publication should be retained by the broker or not.

Returns:

The message identifier of the published message.

Subscribe

```
public byte[] subscribe( String[]  topics,
                         int[] reqQoS ) throws Exception
```

Register an interest in receiving publications on particular topics.

Parameters are:

topics – An array of topics to subscribe to. Wildcard characters may be used in the topic names.
reqQoS – An array indicating the quality of service that publications should be delivered at to the application. A different quality of service may be specified for each topic.

Returns:

An array indicating the Quality of Service that publications will be delivered at for each topic. Each quality of service granted will be less than or equal to the quality requested for each topic.

Unsubscribe

public void unsubscribe(String[] topics) throws Exception

Deregister an interest in receiving publications on particular topics.

Parameters are:

topics – An array of topics to unsubscribe from. Wildcard characters may be used in the topic names.

Callback methods

Callback methods are invoked when particular events occur. Default callback methods are supplied in the ClientMQIsdp class, which do nothing.

If applications want to be more sophisticated and react to events occurring in the underlying protocol then there are a number of ways of doing this:

- The ClientMQIsdp class can be extended and a subset or all of the default callback methods can be overridden.
- A class can implement the MQIsdpSimpleCallback interface and register itself using the registerSimpleHandler method
- A class can implement the MQIsdpAdvancedCallback interface and register itself using the registerAdvancedHandler method

The methods in the MQIsdpSimpleCallback interface are connectionLost and publishArrived as defined below.

The methods in the MQIsdpAdvancedCallback interface 'published' plus the methods in the MQIsdpSimpleCallback interface.

Registering a callback interface

If you choose not to extend the ClientMQIsdp class then you can register another class to be notified of callback events. There is a simple and advanced callback interface. The class which implements one of these callback interfaces must include 'implements MQIsdpSimpleCallback' or 'implements MQIsdpAdvancedCallback' as part of the class declaration.

public void registerSimpleHandler(MQIsdpSimpleCallback simpleCallback)

public void registerAdvancedHandler(MQIsdpAdvancedCallback advCallback)

Connection

public void connectionLost() throws Exception

The connectionLost() method is invoked whenever the MQIsdp connection or the underlying TCP/IP connection is broken. An application should reconnect using the one of the connect methods. If cleanstart is being used, then topics should also be resubscribed to in this method.

Publications

public void publishArrived(String topic,
byte[] message,
int QoS,
Boolean retained)

The publishArrived method is invoked whenever a publication is received from the WebSphere MQ Integrator broker.

Parameters are:

topic – The topic associated with the published data.
message – The published data.
QoS – The Quality of Service at which the publication was delivered.
retained – Was this a retained publication?

NOTE: If this method is overridden, then an application should NOT call any of the other API methods in this method as this will cause a deadlock. Blocking for a significant amount of time is also not advised. If API calls or blocking are required in this method, then the method should start up a new thread to do the work.

public void published(int msgId)

The published method is invoked when a Quality of Service 1 or 2 message has been successfully published to the WebSphere MQ Integrator broker. That is when all the MQIsdp protocol flows have successfully completed for the particular Quality of Service.

Parameters are:

msgId – The message identifier of the message that was successfully published.

Session methods

Session status

public String getServerAddress ()
Return the TCP/IP address or hostname to which the protocol is connected.

public int getPortNumber()
Return the TCP/IP port to which the protocol is connected.

public boolean isConnected()
A boolean indicating the current state of the connection to the WebSphere MQ Integrator broker.

public void terminate()
The terminate method should be called when the application has finished using the ClientMQIsdp class. The terminate method stops all threads started by the class. Once terminate has been called no further API calls can be made. A new instance of the class is required if a new connection is required.

Session timing

public void setBlockingTimeout(int timeout)
public int getBlockingTimeout()

The timeout value controls how long the subscribe or unsubscribe methods wait for a subscribe or unsubscribe to complete. Default is 30 seconds. With a retry interval of 10 seconds (see setRetry below) then up to three attempts will be made to deliver a subscribe or unsubscribe message to the broker.

public void setRetry(int retryNum)
public int getRetry()

Set and get methods for the retry interval in seconds. If a response has not been received in this time from the broker indicating that the message has been received then the message will be retried. Default is 10 seconds.

Message status

public boolean outstanding(int msgId)

The status of any QoS 1 or 2 publication can be established by using the message identifier return by the publish method. Outstanding will return true if the message has not yet been successfully delivered.

Diagnostics

Exceptions

Exceptions will be thrown for both user error and runtime exceptions. For user error exceptions the exception getMessage() method of the exception should yield some useful text.

Trace

public void startTrace()
public void stopTrace()

The startTrace() and stopTrace() methods control the collection of trace if required. A binary trace file called mqe0.trc is generated in the current directory. This trace file may be moved to another system, or formatted in situ using the MQIsdpTraceFormat jar file.

To format the trace execute:

```
java – jar MQIsdpTraceFormat.jar mqe0.trc
```

Or place MQIsdpTraceFormat.jar in the classpath and execute:

```
java com.ibm.MQIsdp.trace.MQeTraceFromBinaryFile mqe0.trc
```

Formatted trace will be written to stdout.

NOTE: Tracing is only available in the J2SE implementation. The startTrace() and stopTrace() methods will have no effect in the J2ME MIDP-2.0 environment.

Chapter 4. Using the sample applications

J2SE sample

The jar file J2SE\MQIsdpSample.jar contains a sample swing user interface for publish/subscribe, which uses the MQIsdp Java classes supplied in this SupportPac. The source code for this user interface is supplied for reference purposes in package com.ibm.MQIsdp.utility.

To run the user interface: Make sure MQIsdpSample.jar and MQIsdp.jar are in the same directory. Then execute: java –jar MQIsdpSample.jar

Or

Place MQIsdpSample.jar and MQIsdp.jar in the classpath. Then execute Java com.ibm.MQIsdp.utility.MQIsdpFrame

Compiling and packaging

The source code is for the sample is provided in com\ibm\MQIsdp\utility Use the following java utilities to compile and package the sample application:

- Compile the code using the following javac command line:

  ```
  javac -d <build output directory> com\ibm\MQIsdp\utility\*.java
  ```

- To package the code as a jar file execute the following jar command line. The manifest file specifies that the com.ibm.MQIsdp.utility.MQIsdpFrame class contains the main method for the jar file.

  ```
  jar cvfm MQIsdpSample.jar  com\ibm\MQIsdp\utility\MANIFEST.MF  <build output directory>\com\ibm\MQIsdp\utility\*.class
  ```

Navigating the user interface

Connection

Specify the TCP/IP address and port number of the SCADAInput node of your WebSphere MQ Integrator broker. Prior to connecting the following options can be set by clicking the options button:

Trace Start/Stop – Trace may be started and stopped at any time. A binary trace file will be produced in the current directory. See the section on Diagnostics.

Client Identifier – The application identifier that the MQIsdp protocol uses to connect

Clean Session – In the event of the MQIsdp connection unexpectedly terminating, should the WebSphere MQ Integrator broker remove all subscriptions and publications for the previously connected client.

Keep Alive - If the WebSphere MQ Integrator broker does not receive any data within this interval it will assume the client application has stopped functioning. The MQIsdp Java classes automatically manage keeping the connection alive, providing the TCP/IP connection is alive, by sending a MQIsdp ping message.

Last Will and Testament – Specify a topic and data that should be published by the broker in the event of the MQIsdp connection being terminated unexpectedly.

Subscriptions

Subscribe Topic and Request QoS – Specify a topic to subscribe to or unsubscribe from and the Quality of Service at which the application wants publications delivered to it for this topic.

Received Topic – When the application receives a publication from the broker it displays the topic in this field.

Publications

Topic, QoS and retained – Publish a message on this topic, at the request Quality of Service. Also specify whether the publication should be retained by the broker.

J2ME MIDP-2.0

The jar file J2ME_MIDP-2.0\MQIsdpSample.jar contains a sample lcdui user interface for publish/subscribe, which uses the MQIsdp Java classes supplied in this SupportPac. The source code for this user interface is supplied for reference purposes in package com.ibm.MQIsdp.midpapp.

To run the user interface: Copy MQIsdpSample.jar and MQIsdpSample.jad onto the MIDP-2.0 device.

For IBM's j9 embedded java on PocketPC the shortcut syntax to launch the MIDlet is below. This assumes the jar and jad file are installed in \MQIsdp.

```
180#j9.exe -jcl:mpng:loadlibrary=ivempng20 -
Xbootclasspath:\ive\lib\jclMidpNG\classes.zip;\MQIsdp\MQIsdpSample.jar
javax.microedition.lcdui.AppManager \MQIsdp\MQIsdpSample.jad
```

Compiling and packaging

To compile the application you need the J2ME Wireless Toolkit 2.0 which is available from http://java.sun.com.

Create a project called MQIsdpSample and copy the J2ME_MIDP2.0\MQIsdp.jar file into the lib subdirectory of the project. Copy the java source in com\ibm\MQIsdp\midpapp into the src directory of the project. The resulting directory structure should be src\com\ibm\MQIsdp\midpapp\*.java.

You can then build the J2ME application using the KToolbar user interface. Selecting package from the project menu will create a jar and jad file that can be used on MIDP-2.0 compliant device.

Navigating the user interface

Before running the MIDlet make sure that the MIDP-2.0 device is able to make a TCP/IP connection to a WebSphere MQ Integrator broker.

The MIDlet exposes the raw publish/subscribe API, allowing topics and Quality of Service to be explicitly expressed.

Connection

Specify the TCP/IP address and port number of the SCADAInput node of your WebSphere MQ Integrator broker. Optionally you can change the Client Identifier by which the MIDlet will use to identify itself to the broker.

Subscriptions

To receive data from the broker specify a topic to subscribe for.

This may be a topic another application is publishing on, a topic on which you will be publishing data from this MIDlet (so all data sent by the MIDlet is also received by it), or wildcard topic #, which will cause the MIDlet to receive all data published to the broker by all applications.

Publications are received in the Inbox and the associated topics are displayed. You then have options to read the publication or delete it.

Publications

To send data from the MIDlet specify a topic on which data will be sent.

This may be to a topic for which this MIDlet is already subscribed or a topic that another application is subscribed to. Specify some textual data and select a Quality of Service at which the message should be published – 0, 1 or 2.

--- End of Document ---