WebSphere MQ Integrator - JavaCICSClient plug-in
Version 1.0

01 March, 2002

Vicente Suarez
IBM UK Ltd
Hursley Park
Winchester, Hampshire
United Kingdom

Vicente_suarez@uk.ibm.com

Property of IBM

Take Note!

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

First Edition, March 2002

This edition applies to Version 1.0 of JavaCICSClient and to all subsequent releases and modifications unless otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2002. All rights reserved. Note to US Government Users -- Documentation related to restricted rights -- Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule contract with IBM Corp.

Table of Contents

Notices

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product.

Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS-IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Trademarks and service marks

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

- IBM
- MQSeries
- WebSphere MQ
- WebSphere MQ Integrator
- MQSI
- CICS

The following terms are trademarks of other companies:

- Windows NT/2002

Acknowledgments

Summary of Amendments

Date	Changes
01 March 2002	Initial release

Preface

This SupportPac is an implementation in Java of the functionality of the SupportPac IA74 – CICS Client plug-in node (written in C/C++). This SupportPac runs on WebSphere MQ Integrator (WMQI) V2.1 or above. Java enables the use of the JavaCICSClient plug-in node in z/OS and all the other platforms supported by the WMQI V2.1 Brokers and the CICS Transaction Gateway (CICS TG). The SupportPac IA74 – CICS Client plug-in node is not supported in z/OS.

The JavaCICSClient plug-in node does the following:

- It connects to a CICS application server region in z/OS,
- it requests the execution of a CICS application program,
- it passes to this program the message received from the ‘in’ terminal on a CICS commarea,
- the CICS program processes the message and returns a reply message on the CICS commarea to the plug-in,
- then this returned message is propagated to the ‘out’ terminal as BLOB to the rest of the message flow.

Bibliography

- *IBM WebSphere MQ Integrator Version 2.1 Working with Messages,* IBM Corporation. SC34-6039.
- *IBM WebSphere MQ Integrator for z/OS Version 2.1 Customization and Installation Guide,* IBM Corporation. SC34-5919
- *IBM WebSphere MQ Integrator Version 2.1 Programming Guide,* IBM Corporation. SC34-5603
- *IBM WebSphere MQ Integrator Version 2.1 Using the Control Center,* IBM Corporation. SC34-5602
- *CICS Family: Client/Server Programming,* IBM Corporation. SC34-5947
- *CICS Transaction Gateway Version 4.0 Programming, IBM Corporation. SC34-5938*
- *MQSI SupportPac IA74 – MQSeries Integrator – CICS Client plug-in V1.0*

Chapter 2. Overview

Overview

The JavaCICSClient node provides a way to pass data and to execute programs in a CICS application server (CICS region) using synchronous communication as part of the process of a WMQI message flow.

The JavaCICSClient node initiates a synchronous connection to a CICS application server using the CICS TG Client daemon that is part of the CICS Transaction Gateway (CICS TG). The CICS application server must be configured to allow incoming client connections and the CICS TG Client must be configured to connect to the CICS application server.

The JavaCICSClient node issues an External Call Interface (ECI) request to execute a specified application program in a CICS application server. The name of the program to be executed is configured as a node attribute. The input message is copied to a COMMAREA (maximum size 32K) and sent to the CICS application server. On return from the CICS application server, the resulting message in the COMMAREA is propagated onwards as a BLOB message body which can be reparsed if needed by a Reset Content Descriptor node.

In the event of an error, an user exception is generated and the original message is propagated to the *failure* terminal and an error message is logged in the event log.

The JavaCICSClient node utilizes the services of the IBM CICS TG Client. This product must be installed and configured on each system that the WMQI broker runs an instance of a message flow that contains the JavaCICSClient node.

The JavaCICSClient node runs in all the platforms supported by the WMQI V2.1 broker and the CICS TG Client. This includes z/OS.

Chapter 3. Installing the plug-in node

SupportPac contents

The supplied ia0w.zip file should be unzipped in a temporary (NT/2000) directory or imported as a Java package to WebSphere Studio Application Developer (refer to Chapter 6). The following files and sub-directories are created:

\example (test message flow and test CICS program)

\com\isv (.java and .class files)

\tool\com\isv (.properties file)

\tool\images (.gif files)

\tool\help\EN_us\com\isv (.htm help file)

\Resources (miscellaneous files including this document (.pdf))

Prerequisites

This SupportPac provides a plug-in node to be used with the IBM WebSphere MQ Integrator Version 2.1 and above and the IBM CICS Transaction Gateway (CICS TG) version 4.0 and above. If any changes are to be made to the plug-in node, an appropriate Java Development Kit (JDK) V1.3 or WebSphere Studio application Developer V4.0 (or above) is required.

Supported Platforms

This SupportPac has been developed and tested in Microsoft Windows NT and z/OS environments. But this plug-in node should run on any environment that is supported by the WMQI Broker V2.1 and the CICS TG Client.

Installing the plug-in node on a broker

Update the following environment variables on the WMQI broker environment where the plug-in is going to be deployed or update the ‘mqsicompcif’ and ‘ENVFILE’ files in z/OS as follows:

- Add to the CLASSPATH
    - :<wmqi_install_path>/classes/jplugin.jar:<ctg_install_path>/classes/ctgclient.jar
    - :<ctg_install_path>/classes/ctgserver.jar (To support ‘local’ connections to the CICS application server. If the CICS TG Client is used, it is not required to add this jar file)
- Add to the PATH (or LIBPATH in z/OS)
    - :<ctg_install_path>/bin (To support ‘local’ connections to the CICS application server. If the CICS TG Client is used, it is not required to add this directory)
- If the CICS TG Client is not used then add to STEPLIB or LINKLST (z/OS only),
    - The following CICS library: hlq.SDFHEXCI

The plug-in ‘**JavaCICS**.**jar**’ file should be installed by copying or moving the file to the following directory:

- <wmqi_install_path>\jplugin (Windows)

- <wmqi_install_path>/lil (Unix or z/OS)

The file JavaCICS.jar is included in the ia0w.zip file of this SupportPac in the /Resources directory.

You must stop and restart the broker to activate the new '**jar**'.

Integrating the plug-in node into the Windows Control Center

The necessary files for integrating the plug-in into the Windows Control Center are provided in the \tool\ directory in the ia0w.zip file.

Use the following table to copy the files to their correct location. These locations should already exist providing you have deployed at least one message flow. Append your ***<wmqi_install_path>*** to the **Copy to location** value.

Filename	Copy to location
\tool\images\JavaCICSClient.gif	\Tool\images
\tool\images\JavaCICSClient30.gif	\Tool\images
\tool\images\JavaCICSClient42.gif	\Tool\images
\tool\images\JavaCICSClient58.gif	\Tool\images
\tool\images\JavaCICSClient84.gif	\Tool\images
\tool\com\isv\JavaCICSClient.properties	\Tool\com\isv\
\tool\help\EN_us\com\isv\MessageProcessing NodeType_JavaCICSClient.htm	\Tool\help\EN_us\com\isv\

Importing the node definition to the configuration repository

When you have installed the files in the appropriate directories, as described in the previous section, you must import the plug-in definition to the Control Center.

1. Start the Control Center. The user ID you are using must be a member of the WebSphere MQ Integrator group ***mqbrdevt***. You are recommended to use the superuser **IBMMQSI2** to complete this task.

2. Select the Message Flows view.

3. Select an existing Message Flow Category, or create a new one (it is recommended to create one called Plugins).

4. Import the plug-in definition selecting *File->Import to Workspace*

5. In the Import Resources window, set the check on message flows and click browse. Locate and select the file called ***\Resources\JavaCICSClient.xml***. Click import.

6. Select the Message Flow Category where the plug-in is going to be placed. Right-click on the Message Flow Category and select *add to workspace->message flow*.

7. A list box is displayed showing all existing IBM-supplied primitive nodes and any defined message flows you have installed following the instructions provided.

8. Select the message flow called ***JavaCICSClient***.

   This node now appears within the message flow category you selected in the tree view in the left-hand pane.

9. Select your new node, and right-click. Select *Check In*.

Chapter 4. Using the plug-in node

Description

Using this node in a message flow requires the following:

- Access to a CICS application server region on z/OS
- Access to a CICS Transaction Gateway (CICS TG) on any platform that is connected to the CICS region on z/OS or if the broker is running on the same z/OS as the CICS region then use the 'local' connection mode instead of the CICS TG.
- Setting up of the plug-in properties (attributes)
- Connecting a least the in and out terminals of the JavaCICSClient node

Plug-in node terminals

Terminal	Description
In	The input terminal that accepts a message for processing by the node.
Failure	The output terminal to which the message is propagated if an error occurred.
Out	The output terminal to which the message is propagated if an error occurred.

Plug-in node properties

Attribute	Default	Description
Name of CICS Region	Mandatory	This is the APPLID of the CICS region that this node is going to connect to.
Name of CICS Program	Mandatory	Name of the CICS program to be executed to process the data in the COMMAREA.
CICS Userid		Userid to access the CICS region if required
Userid's Password		Password to access the CICS region.
CICS Transaction name		Name of the CICS Transaction to be used to run the CICS program. If this is not specified, CICS will use the mirror transaction CSMI.
CICS Transaction Gateway URL	local://	URL for the host where the CICS Gateway is running. Use *local://* if the gateway is running on the same host

		as the WMQI broker or use *tcp://hostname or IP address* if the gateway is running on a remote host.
CICS Transaction Gateway port	2006	Port number that CICS Transaction Gateway is listening.
Length of commarea	0	Length in bytes of the commarea that is passed to the CICS program. This length must match the size of the commarea expected by the program.
Remove MQMD	Check	To indicate if the MQMD is expected by the CICS program in the commarea.

Chapter 5. Recreating the plug-in node

Compiling the plug-in node

If there is a need to recreate the JavaCICS.jar file the following commands should be executed on any environment that has java, WMQI V2.1 and the CICS Transaction Gateway installed:

- javac \com\isv\JavaCICSClient.java
- jar –cvf JavaCICS.jar \com\isv\JavaCICSClient.class

Using the control center SmartGuide to define the plug-in

If you need to update any of the files for the JavaCICSClient node, you must follow these steps:

1. Start the Control Center. If you checked in the node using the WebSphere MQ Integrator superuser ID IBMMQSI2, you must be logged on with this user ID to make any changes. If not, you can use any user ID that is a member of the WebSphere MQ Integrator group **mqbrdev**t.
2. Select the Message Flows view.
3. Select the node that you want to update. Document any attributes (and default values) set on the node that you would like to retain. They will not be available later.
4. Delete the selected node.
5. Use the SmartGuide to create the updated node. None of the properties imported with the node definition will be retained; you have to reset all the attributes and properties, although you can reuse any optional resources, for example the properties file or icon files, from the original node.
    - Start the plugin SmartGuide: Message Flow Types -> Right click Message Flows -> click on Create Plugin Node
    - Define the Label, Identifier and Terminals:

Field name	**Field Value**
Node Label	JavaCICSClient
Node Identifier	ComIsvJavaCICSClient
In Terminal	in
Out Terminal	failure
Out Terminal	out

    - Define Package and Attribute Group:

Field name	**Field Value**
Package	com.isv
Create Attribute Group	Basic

- Define Attributes:

Name	Type	Mandatory
regionName	String	Yes
programName	String	Yes
userid	String	No
password	String	No
transId	String	No
gatewayUrl	String	No
gatewayPort	Integer	No
commareaLength	Integer	No
removeMQMD	Boolean	No

- Set initial values for Attributes:

Name	Initial Value
regionName	blank
programName	blank
userid	blank
password	blank
transId	blank
gatewayUrl	local://
gatewayPort	2006
commareaLength	0
removeMQMD	Check

- Resources: click finish

- Check in the new JavaCICSClient node

6. Install the updated files for this node into the appropriate directories (described in “Integrating the plug-in in the Windows Control Center” on chapter 2). If you have updated any optional resource files, you must remember to install the updated files on every system on which the Control Center is used.

7. Check in the new node definition and export a copy of the node (update the copy in \Resources\JavaCICSClient.xml) for distribution to other configuration managers (WMQI domains).

Chapter 6. Using WebSphere Studio Application Developer

WebSphere Studio Application Developer (WSAD) can be used to edit, compile and export all the class and the resource files that are part of the JavaCICSClient node. WSAD has all the tools to edit and compile java, edit and display XML, edit and display HTML and to debug and trace the execution of the Java node in a WMQI broker on NT/2000. WSAD can be used to export the files to the Control Centre and the broker.

Import the ia0w.zip file as a Java package in WSAD

- Start WSAD
- Create a new project: File -> New -> Project -> Java -> click Next -> enter project name
- Select File -> Import -> Select Zip file -> click Next -> Browse and select the location of the ia0w.zip file -> select all -> Select destination folder (same as the project name created earlier) -> check override resource without warning -> click finish
- The project is created with all the folders, files and resources required to change, recompile and export the JavaCICSClient plugin node using WSAD

How to use WSAD to debug a Java plugin?

Activating the WMQI broker JVM listener debugging port

- It is possible to attach the WSAD Java debugger to the JVM running inside the broker thus allowing developers to debug their Java nodes. The debugger can be attached on a per execution group basis. In order to attach WSAD Java debugger you must first change the broker configuration to tell the JVM to listen on a particular port. The command mqsiservice can be used to change the broker configuration in the following way:
    - mqsiservice <broker> -r debugJava="<ExecutionGroup1Label=port1>;<ExecutionGroup2Label=port2> ..."
        - <broker> - The broker name.
        - <ExecutionGroup1Label=port1> - This tells the JVM in execution group 1 to listen on port1.
        - <ExecutionGroup2Label=port2> - This tells the JVM in execution group 2 to listen on port2.
        - Several entries can be added for each execution group, these are separated by semicolons as illustrated above.
- The entry can be removed in the following way:
    - mqsiservice <broker> -r debugJava
- Misusing mqsiservice can seriously damage the broker configuration. It should only be used in the way described above. This command is not available on z/OS. The broker must be restarted for these changes to take effect. The JVM in the execution group will now listen on the specified port and can be attached to in the usual manner.

Activating remote debugging in WSAD

- Start the WMQI control center, configuration manager and broker (with JVM listening for debugging sessions)
- Select the Java package that has the plugin node to debug
- Select the java source code to be debugged and set breakpoints where it is required to initiate the debugging
- Click on the debug button
- Select the remote java application
- Enter the remote host IP address (where the WMQI broker is running with the JVM port active and where the java node is going to be deployed to)
- Enter the port number of the execution group where the java node will be deployed
- Click finish to initiate the debugging session
- Deploy a flow with the Java plugin node to an execution group that the JVM has a listening port active.
- During the deploy process, the execution group will call the java constructor, the getNodeName and then the attribute setters methods of the java plugin.
- Every time a message is processed by the message flow, the evaluate method is called.

Chapter 7. Example using the plug-in node

The subdirectory \example contains a test CICS program (ec01.c), the JCL to compile and to link the test CICS program in z/OS and a test message flow (TEST_JAVA_CICS).

To install and to run the example follow the following steps:

- Compile and link the CICS program *ec01.c* using the JCL *ec01c.jcl*
- Define, install and load the program in the CICS region using CEDA and CEMT
- Import the test message flow *TEST_JAVA_CICS* using the WMQI Control Centre
- Set the CICS region name and the CICS Gateway URL in the properties of the JavaCICSClient node in the test message flow
- Check in the test message flow
- Deploy the test message flow to the broker where the JavaCICSClient node has been installed
- Define the IN, OUT and FAILURE queues in the broker queue manager
- Put a test message on the input queue IN
- Check the message on the output queue OUT. The output message will be of the same length as the input message but the text replaced by digits

---------- End of Document ----------