Advanced WebSphere MQ RPC Server Functionality

This chapter covers the following topics:

- Support for Dynamic Queue Names
- Support for Request/Reply Scenarios
- Dynamic IDL/RPC Parameters for WebSphere MQ RPC Server
- Handling of Correlation ID
- Support for the MQRFH Header
- Character Encoding Issues
- User Exit for Message Processing
- Transactional Behavior

Support for Dynamic Queue Names

In the properties file, only one input and one output queue can be specified. These are the default queues used by the WebSphere MQ RPC Server. The queue name can also be specified on each call, using a dynamic parameter. Since different queue names can be specified on different calls, multiple input or output queues are supported. The default input and output queues can also be used with the dynamic parameter. Queues specified by a dynamic parameter are opened when they are used for the first time and closed when the WebSphere MQ RPC Server terminates.

Support for Request/Reply Scenarios

A synchronous request/reply call to MQ is possible. In this case, the remote procedure call has to have both `INPUT` and `OUTPUT` parameters, or an `INOUT` parameter has to be specified. The WebSphere MQ RPC Server issues an MQ `PUT` call with message type "Request" on the default output queue. The `Reply To Queue Name` field is set to the name of the default input queue. After the `PUT` call, the WebSphere MQ RPC Server issues an MQ `GET` on the default input queue and then it waits for the reply message (note that for this scenario the input queue must be different from the output queue).

Dynamic queue names can also be used for the request/reply scenario. Use the `MQ_QUEUE_NAME` parameter for the output (request) queue and the `MQ_REPLY_QUEUE` parameter for the input (reply) queue.

You can also use the dynamic `Reply To Queue` parameter to indicate that a reply is expected for this message. In this case, sending the message and receiving the reply is decoupled and is performed by two separate RPC requests.

The request and reply messages are correlated by the correlation ID. The reply message has to have the same correlation ID as the request message.

Dynamic IDL/RPC Parameters for WebSphere MQ RPC Server

With the WebSphere MQ RPC Server it is possible that certain parameters of a remote procedure call are dynamic parameters which are evaluated by the WebSphere MQ RPC Server. Dynamic parameters have a fixed name; they can be defined only on level 1 in the parameter definition in the IDL file and before any variable length parameter, and have a specific format. The WebSphere MQ RPC Server uses the following parameters:

Parameter	Format	Description
`Queue name (MQ_QUEUE_NAME)`	`A48`	Overrides the default input or output queue name specified in the properties file. `IN` parameter only.
`Correlation ID (MQ_CORREL_ID)`	`A48`	The MQ Correlation ID can be used in Request Reply scenarios. If it is an `IN` (or `INOUT`) parameter, it is assigned to the correlationId parameter of the MQ message which is sent to MQ. If it is an `OUT` (or `INOUT`) parameter, it contains the corresponding value of the received MQ message. This parameter is defined of type alphanumeric but its contents is an hexadecimal encoded binary array.
`Reply To Queue (MQ_REPLY_QUEUE)`	`A48`	The replyToQueueName parameter of the MQ message.If it is an `IN` (or `INOUT`) parameter, it specifies that the MQ message is of type Request. If it is an `OUT` (or `INOUT`) parameter, it contains the corresponding value of the received MQ message.
`Wait Interval (MQ_WAIT_INTERVAL)`	A with fixed length	The wait interval in milliseconds for an MQ `GET` call. Overrides the default value `entirex.wmqbridge.waittime` from the properties file. `IN` parameter only.
`MQRFH Header data (MQ_RFH_*)`	A with fixed length	Arbitrary number of parameters. The names following the prefix `"MQ_RFH_"` are used as the names of the name value pairs of the MQRFH header.

If dynamic parameters are to be used, generate a properties file from the corresponding Software AG IDL file and specified with the `entirex.bridge.names.file` property. To generate this property file, use the template *bridge.tpl* in the *template* subdirectory of the EntireX installation. For batch generation, run `erxidl.bat` (Windows) or `erxidl.bsh` (UNIX) with the parameters "`-t <path to template directory>/bridge.tpl <idlFile>`".

Alternatively, you can also use the *EntireX Workbench*. Go to the **Preferences for EntireX** and create a new custom wrapper. Specify a name and browse to the *bridge.tpl* template. If the Custom Wrapper has been created (and the Workbench restarted), you can generate the properties file from an IDL file, using the context menu item **Other** > **Generate...**.

Handling of Correlation ID

The correlation ID can be explicitly used in get message and put message operations using the dynamic parameter `MQ_CORREL_ID`.

For Request Reply scenarios there is also an implicit usage of the correlation ID by the WebSphere MQ RPC Server: If `MQ_CORREL_ID` is not explicitly specified, MQ is instructed to generate a correlation ID. The option `MQPMO_NEW_CORREL_ID` is set internally to achieve this. When reading the reply the option `MQMO_MATCH_CORREL_ID` is specified, thus the reply message has to use the same correlation as specified in the request message.

Support for the MQRFH Header

MQ messages may have custom specific headers. The `MQRFH` header (rules and formatting header) consists basically of name value pairs. Restriction: only one header per MQ message is possible; MQ allows an arbitrary number of headers per message.

When sending a message to MQ: a `MQRFH` header is built if at least one parameter in the IDL file has a name with prefix `"MQ_RFH_"`. All IN (or INOUT) parameters with this prefix are used to build the header. If for example the IDL file contains two fields `MQ_RFH_H1` and `MQ_RFH_H2` with the values `v1` and `v2`, the resulting MQRFH header will have two name value pairs, `H1 v1` and `H2 v2`.

If a message is received from MQ: if the message has a `MQRFH` header, all value entries in the name value pairs are copied to the corresponding `OUT` (or `INOUT`) parameter in the IDL file. The name has to match the part of the IDL parameter name after the prefix. In the above example consider that the MQ message has two name value pairs, `H1 v11` and `H3 v22`. Then the value `v11` will be assigned to the parameter `MQ_RFH_H1`, the parameter `MQ_RFH_H2` gets no value assigned, and the entry for `H3` will be ignored.

Character Encoding Issues

When the WebSphere MQ RPC Server is exchanging messages via the EntireX Broker with an RPC client, the usual rules apply. By default, the message is exchanged between the WebSphere MQ RPC Server and EntireX using the platform encoding of the JVM which executes the WebSphere MQ RPC Server.

If the payload of the MQ message is in XML format (`property entirex.bridge.xmm` has been set), the WebSphere MQ RPC Server converts the XML payload to the encoding used for the remote procedure call. If the WebSphere MQ RPC Server has to create the XML payload, it will use UTF-8. A different encoding can be used by setting the property `entirex.bridge.xml.encoding`.

If the payload of the MQ message is of type text, the translation of the MQ message payload is done by the IBM MQ Java classes. When sending a message, the WebSphere MQ RPC Server converts the message to the encoding specified by the CCSID (Coded Character Set IDentification) of the queue manager. When receiving a message, the WebSphere MQ RPC Server converts the message to the platform encoding of the JVM.

Note:
The default platform encoding of the JVM can be changed by setting the system property `file.encoding` in the startup script of the WebSphere MQ RPC Server.

User Exit for Message Processing

WebSphere MQ does not have a clearly defined message layout, it is basically a stream of bytes. In general it is up to the MQ application to know the exact semantics of an MQ message. This might include application-specific headers and formatting rules. The WebSphere MQ RPC Server supports a general but simplified model of message processing.

To better handle application specific message layout details a user exit (or callback routine) can be used. The user exit is working on the WebSphere MQ Java representation of an MQ message (class `com.ibm.mq.MQMessage`) and can change the MQ message. The user exit gets control:

1. after an MQ message has been constructed by the WebSphere MQ RPC Server and before the message is put to the MQ queue,

2. after an MQ message has been read from an MQ queue and before it is processed by the WebSphere MQ RPC Server.

The user exit can be used for example for an application specific processing of the `MQRFH`, `MQRFH2` or even custom headers.

To enable a user exit, use the property `entirex.wmqbridge.userexit` to specify the class name of the user exit implementation. The class will be loaded using the standard classpath. You can specify a separate classpath with the property `entirex.wmqbridge.userexit.classpath`. Note that for the classpath a file or HTTP URL must be specified. Your user exit class must implement the Java interface `com.softwareag.entirex.rpcbridge.WMQBridgeExit`. This Java interface has the following methods:

```
/**
 ** This method is called after the message has been created by the WMQBridge
 ** and before the message is sent to an MQ queue (MQPUT).
 ** The Message object and/or the MessageOptions object can be changed.
 **
 ** @param msg The MQ message object.
 ** @param pmo The MQPutMessageOptions object.
 **/
public void beforePut(com.ibm.mq.MQMessage msg, com.ibm.mq.MQPutMessageOptions pmo);
/**
 ** This method is called before a message is retrieved from an
 ** MQ queue (MQGET). The MessageOptions object can be changed.
 **
 ** @param gmo The MQGetMessageOptions object.
 **/
public void beforeGet(com.ibm.mq.MQGetMessageOptions gmo);
/**
 ** This method is called after a message has been retrieved from an
 ** MQ queue (MQGET) and before the message will be processed by the WMQBridge.
 ** The Message object can be changed.
 **
 ** @param msg The MQ message object.
 **/
public void afterGet(com.ibm.mq.MQMessage msg);
```

Transactional Behavior

Calls to MQ Series are non-transactional by default. Thus the request operates outside the normal unit-of-work protocols. When reading a message with MQ `GET`, the message is deleted from the queue immediately. If an error occurs in the further processing of the message within the WebSphere MQ RPC Server (for example the translation to RPC or XML results in an error), the message cannot be made available again. The same applies to sending a message, the MQ `PUT` operation makes the message available immediately.

If the RPC client application uses conversational RPC, the MQ calls are issued transactional (using the `SYNCPOINT` option). A Backout Conversation will send a backout to the queue manager, and a Commit Conversation will send a commit to the queue manager.

To understand the level of guaranteed delivery provided by the WebSphere MQ RPC Server we present the flow of control when reading a message from a queue or writing a message to a queue. Sending a message to an MQ queue:

To send a message to an MQ Queue

1. The RPC client application sends a send request to the WebSphere MQ RPC Server.
2. The WebSphere MQ RPC Server creates a corresponding MQ message and puts the message on the queue. If the remote procedure call is part of an RPC conversation, the message is not committed.
3. The WebSphere MQ RPC Server returns a positive acknowledgment back to the RPC client. If something fails in step 2, an error is returned to the RPC client.

To receive a message from an MQ Queue

1. The RPC client application sends a receive request to the WebSphere MQ RPC Server.
2. The WebSphere MQ RPC Server reads a message from the queue. If no message is available, an error is returned to the RPC client. If the remote procedure call is part of an RPC conversation, the message is not committed.
3. The WebSphere MQ RPC Server creates a corresponding RPC reply which is sent back to the RPC client. If something fails in step 2, an error is returned to the RPC client.

If the send or receive call is part of a conversational RPC, the MQ transaction will get a commit or backout when the RPC conversation is closed, depending on the type of the endConversation call.