webMethods EntireX

RPC Server for IBM® MQ

Version 9.12

October 2016

WEBMETHODS

This document applies to webMethods EntireX Version 9.12 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: EXX-MQRPC-912-20181116

Table of Contents

1 About this Documentation 1
- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

2 Introduction to the RPC Server for IBM MQ 5
- Overview 6
- Sending a Message to an IBM MQ Queue 7
- Receiving a Message from an IBM MQ Queue 8

3 Mapping RPC Data to the MQ Message Buffer 9
- Mapping IDL as XML 10
- Mapping IDL as Text 10

4 Administering the RPC Server for IBM® MQ 13
- Customizing the RPC Server 14
- Configuring the RPC Server Side 15
- Configuring the IBM MQ Side 18
- Starting the RPC Server 19
- Stopping the RPC Server 20
- Using SSL/TLS with the RPC Server 20
- Running the RPC Server as a Windows Service 22
- Activating Tracing for the RPC Server 23

5 Advanced RPC Server for IBM MQ Functionality 25
- Support for Request/Reply Scenarios 26
- Handling of Correlation ID 26
- Character Encoding Issues 26
- User Exit for Message Processing 27
- Transactional Behavior 28

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **http://documentation.softwareag.com**. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at **https://empower.softwareag.com/**.

You can find product information on the Software AG Empower Product Support website at **https://empower.softwareag.com**.

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at **https://empower.softwareag.com/public_directory.asp** and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at **http://techcommunity.softwareag.com**. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Introduction to the RPC Server for IBM MQ

- Overview 6
- Sending a Message to an IBM MQ Queue 7
- Receiving a Message from an IBM MQ Queue 8

The RPC Server for IBM® MQ runs as an RPC server and processes RPC client calls. It is used to send messages to and receive messages from an IBM MQ queue. This means that existing EntireX wrappers can be used for communication with IBM MQ. This chapter covers the following topics:

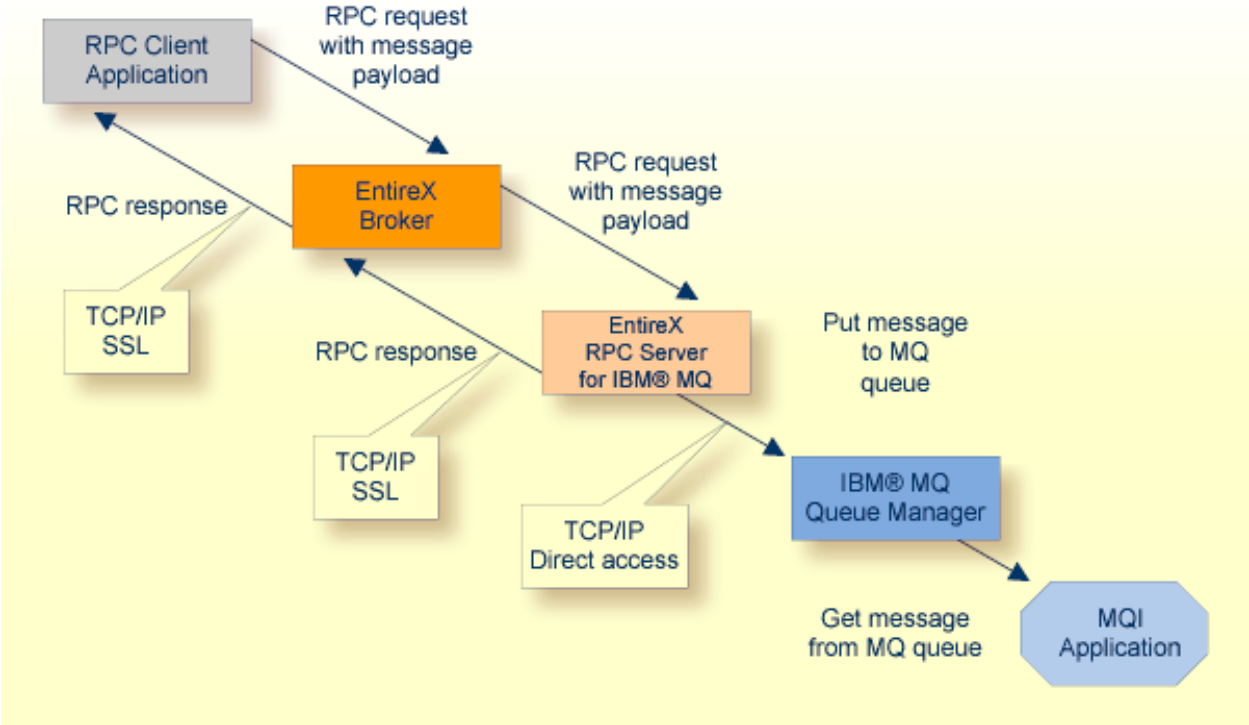
Overview

The RPC Server for IBM® MQ allows standard RPC clients to send and receive asynchronous and synchronous messages to an IBM® MQ queue manager. The RPC Server for IBM MQ uses the IBM® MQ base Java classes from IBM. It can connect to an IBM® MQ either as an IBM® MQ client using TCP/IP (*client mode*) or in so-called *bindings mode* where it is connected directly to IBM® MQ running on the same machine. Note that on z/OS, only bindings mode is supported.

The RPC Server for IBM MQ runs as an RPC server and processes RPC client calls. An RPC client can send an asynchronous message (MQ `PUT` call) if it uses a program with `IN` parameters. An RPC client can receive an asynchronous message if it uses a program with `OUT` parameters (MQ `GET` call). The receiver must use the same parameters in a program as the sender, but with the direction `OUT` instead of `IN`. Processing of synchronous messages (request/reply scenario) is possible if the program uses a mixture of `IN` and `OUT` parameters. For possibilites on how RPC data is mapped to MQ messages, see **Mapping RPC Data to the MQ Message Buffer**. The images below illustrate message transport when sending and receiving messages. If the RPC client application uses conversational RPC, the MQ calls are issued transactionally (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.

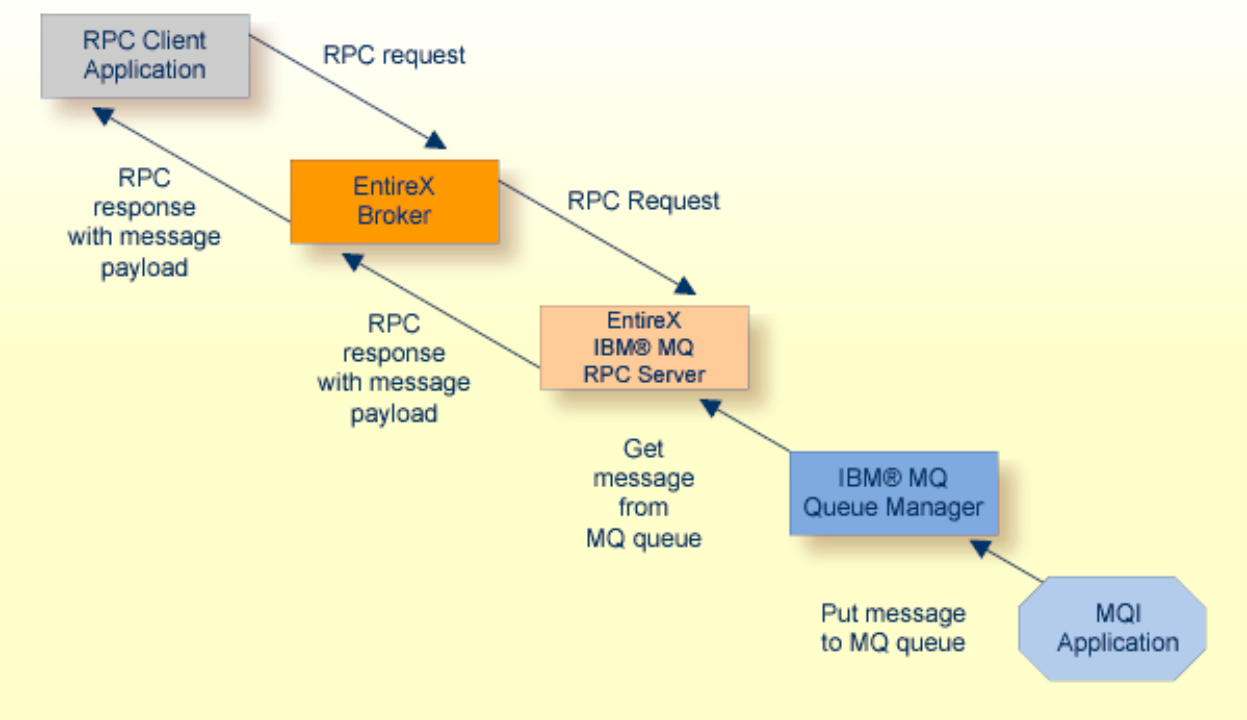
The RPC Server for IBM MQ registers to one RPC service. On the MQ side it uses one input queue and one output queue.

Sending a Message to an IBM MQ Queue



Note: All messages sent to an RPC Server for IBM MQ instance via a specific RPC service are put on the same MQ output queue.

Receiving a Message from an IBM MQ Queue



Note: All messages retrieved by the RPC Server for IBM MQ from the MQ input queue are passed to the same RPC service. Messages are retrieved in the order they appear on the queue.

3 Mapping RPC Data to the MQ Message Buffer

- Mapping IDL as XML 10
- Mapping IDL as Text 10

Mapping IDL as XML

IDL is mapped to XML format using the *XML Mapping Editor*. The outcome of this process is an XML mapping file (Workbench file with extension xmm). This resulting XMM file has to be specified by the configuration property `entirex.bridge.xmm`.

Mapping IDL as Text

The RPC Server for IBM MQ uses the predefined mapping of IDL data types to the MQ message buffer. See below. If your programs use arrays of groups, set the property `entirex.bridge.marshalling` to "Natural" or "COBOL". If your programs do *not* use arrays of groups, do not set `entirex.bridge.marshalling`.

Data Type	Description	Format	Note
A*number*	Alphanumeric	*number* bytes, encoding the characters.	
AV	Alphanumeric variable length	Bytes up to the end of the buffer.	1, 4
AV[*number*]	Alphanumeric variable length with maximum length	Bytes up to the end of the buffer, maximal length *number*.	1
K*number*	Kanji	Same as data type A.	
KV	Kanji variable length	Same as data type AV.	1, 4
KV[*number*]	Kanji variable length with maximum length	Same as data type AV[*number*].	1
I1	Integer (small)	*sign* (+, -) and 3 bytes (digits).	
I2	Integer (medium)	*sign* (+, -) and 5 bytes (digits).	
I4	Integer (large)	*sign* (+, -) and 10 bytes (digits).	
N*number1*[.*number2*]	Unpacked decimal	*sign* (+, -), *number1* bytes (digits) [*number2*] bytes (digits), no decimal point.	
P*number1*[.*number2*]	Packed decimal	*sign* (+, -), *number1* bytes (digits) [*number2*] bytes (digits), no decimal point.	
L	Logical	1 byte: X for true, all other false.	
D	Date	*YYYYMMDD*.	2
T	Time	*YYYYMMDDhhmmssS*.	3

Notes:

1. Only as last value.
2. *YYYY* year, *MM* month, *DD* day.
3. *YYYY* year, *MM* month, *DD* day, *hh* hour, *mm* minute, *ss* second, *S* tenth of a second.

4. Not possible when using COBOL.

Data types not supported:

- Binary (`B[n]`,`BV`, `BV[n]`)
- Floating point (`F4`, `F8`)

4 Administering the RPC Server for IBM® MQ

- Customizing the RPC Server 14
- Configuring the RPC Server Side 15
- Configuring the IBM MQ Side 18
- Starting the RPC Server 19
- Stopping the RPC Server 20
- Using SSL/TLS with the RPC Server 20
- Running the RPC Server as a Windows Service 22
- Activating Tracing for the RPC Server 23

The RPC Server for IBM® MQ runs as an RPC server and processes RPC client calls. It is used to send messages to and receive messages from an IBM MQ queue. This means that existing EntireX wrappers can be used for communication with IBM MQ.

Customizing the RPC Server

The following elements are used to set up the RPC Server for IBM MQ:

- Prerequisites
- Configuration File
- Start Script

Prerequisites

There are two parts: the RPC server and the IBM® MQ side. The RPC Server for IBM MQ uses the following:

- The IBM® MQ base Java classes from IBM. To run the RPC Server for IBM MQ, you need either the base Java classes or a full installation of IBM® MQ. See *Prerequisites for RPC Server and Listener for IBM® MQ* in the respective section of the EntireX Release Notes for the required JAR file(s). The IBM® MQ environment variables `MQ_JAVA_LIB_PATH` and `MQ_JAVA_INSTALL_PATH` must be set. Make sure that either the local IBM® MQ installation or the IBM® MQ Java classes are accessible.
- The WS-Stack. Therefore the start scripts contain references to JAR files in the WS-Stack directory. If you update these JAR files, you may need to customize the JAR file names in the script files.

Configuration File

The default name for the configuration file is *entirex.wmqbridge.properties*. The RPC Server for IBM MQ searches for this file in the current working directory. You can set the name of the configuration file with `-Dentirex.server.properties=` *`your file name`*. Use the slash "/" as file separator. The configuration file contains the configuration for both parts of the RPC Server for IBM MQ.

Start Script

The start script for the RPC Server for IBM MQ is called *wmqbridge.bsh* (UNIX) or *wmqbridge.bat (Windows)* and is provided in the *bin* folder of the installation directory. You may customize this file. The script uses the configuration file *entirex.wmqBridge.properties* in the folder *etc*. The RPC Server for IBM MQ itself is contained in the file *entirex.jar*.

Configuring the RPC Server Side

The RPC server side of the RPC Server for IBM MQ is configured like the Java RPC Server. The RPC Server for IBM MQ uses all properties starting with “entirex.server”.

The RPC server side can adjust the number of worker threads to the number of parallel requests. Use the properties `entirex.server.fixedservers`, `entirex.server.maxservers` and `entirex.server.minservers` to configure this scalability.

With `entirex.server.fixedservers=yes`, the number of `entirex.server.minservers` is started and the server can process this number of parallel requests.

With `entirex.server.fixedservers=no`, the number of worker threads balances between `entirex.server.minservers` and `entirex.server.maxservers`. This is done by a so-called attach server thread. On startup, the number of worker threads is `entirex.server.minservers`. If more than `entirex.server.minservers` are waiting for requests, a worker thread stops if its receive call times out. The timeout period is configured with `entirex.server.waitserver`.

Alternatively you can use the command-line option. The command-line parameters have a higher priority than the properties set as Java system properties, and these have higher priority than the properties in the configuration file. For a list of all of the command-line parameters, use `-help`.

Property Name	Parameter	Default	Explanation
`entirex.bridge.marshalling`			Define the type of marshalling (Natural or COBOL). Must be set only if the IDL file contains arrays of groups. See **Mapping RPC Data to the MQ Message Buffer**.
`entirex.server.brokerid`	`-broker`	`localhost`	Broker ID. See *URL-style Broker ID*.
`entirex.server.compresslevel`	`-compresslevel`	`0` (no compression)	Permitted values (you can enter the text or the numeric value):
			`BEST_COMPRESSION` → `9`
			`BEST_SPEED` → `1`
			`DEFAULT_COMPRESSION` → `-1`, mapped to `6`
			`DEFLATED` → `8`
			`NO_COMPRESSION` → `0`
			`N` → `0`
			`Y` → `8`

Property Name	Parameter	Default	Explanation
entirex.server.fixedservers		no	If no, use an attach server thread to manage worker threads, otherwise run the minimum number of server threads. See the properties entirex.server.maxservers and entirex.server.minservers.
	-help		Display usage of the command-line parameters.
entirex.server.logfile	-logfile		Name of the log file, the default is standard output.
entirex.server.maxservers		32	Maximum number of worker threads.
entirex.server.minservers		1	Minimum number of server threads.
entirex.server.password	-password		The password for secured access to the Broker. The password is encrypted and written to the property entirex.server.password.e. To change the password, set the new password in the properties file (default is *entirex.wmqbridge.properties*). To disable password encryption, set entirex.server.passwordencrypt=no (default for this property is yes).
entirex.server.properties	-propertyfile	entirex.wmqbridge.properties	The file name of the property file.
entirex.server.restartcycles	-restartcycles	15	Number of restart attempts if the Broker is not immediately available. This can be used to keep the Java RPC Server running while the Broker is temporarily down.
entirex.server.security	-security	no	no/yes/auto/Name of BrokerSecurity object.
entirex.server.serveraddress	-server	RPC/SRV1/CALLNAT	Server address.
entirex.server.serverlog	-serverlog		Name of the file where worker thread starts and stops are logged. Used by the Windows RPC Service.
entirex.server.userid	-user	WMQRPCServer	The user ID for the Broker for RPC. See entirex.server.password.
entirex.server.verbose	-verbose	no	Enable verbose output to the log file.
entirex.server.waitattach		600S	Wait timeout for the attach server thread.
entirex.server.		300S	Wait timeout for the worker threads.

Property Name	Parameter	Default	Explanation
waitserver			
entirex.timeout		20	TCP/IP transport timeout. See *Setting the Transport Timeout* under *Writing Advanced Applications - EntireX Java ACI*.
entirex.trace	-trace	0	Trace level (1,2,3).

Configuring the IBM MQ Side

These properties are used to configure the connection to the IBM® MQ queue manager.

Name	Parameter	Default Value	Explanation
entirex.wmqbridge.host			If host is not specified, bindings mode is used to connect to the local MQ Server. Otherwise specify the hostname or IP address of the MQ Server.
entirex.wmqbridge.port		1414	Port of the MQ Server. Not used in bindings mode.
entirex.wmqbridge.channel		SYSTEM.DEF.SVRCONN	Channel name used to the MQ Server. Not used in bindings mode.
entirex.wmqbridge.queuemanager	-wmqqmanager		Name of the (local or remote) queue manager. If not specified, a connection is made to the default queue manager.
entirex.wmqbridge.inputqueue	-wmqinqueue		Name of input queue (the queue that is used for MQ GET operations).
entirex.wmqbridge.outputqueue	-wmqoutqueue		Name of output queue (the queue that is used for MQ PUT operations).
entirex.wmqbridge.userid	-wmquser		UserID for MQ Server.
entirex.wmqbridge.password	-wmqpassword		Password for MQ Server.
entirex.wmqbridge.waittime			Wait interval for MQ Get operation in milliseconds.
entirex.wmqbridge.userexit			Class name for user exit.
entirex.wmqbridge.userexit.classpath			URL of the classpath for user exit (optional).
entirex.wmqbridge.ccsid		platform encoding	Coded Character Set Identification used by the RPC Server for IBM® MQ (which acts as an MQ client), unused in bindings mode.
entirex.wmqbridge.mqtrace		0	MQ tracing enabled if parameter > 0.
entirex.bridge.xmm			Name of XMM (XML Mapping) file; if MQ message payload is XML/SOAP. If this is specified, messages to/from MQ will be converted to XML.

Name	Parameter	Default Value	Explanation
`entirex.bridge.verbose`		`no`	Verbose/trace mode of RPC Server for IBM MQ
`entirex.bridge.xml.encoding`		`utf-8`	Encoding of the XML document that is sent to MQ (if `entirex.bridge.xmm` is used).
`entirex.wmqbridge.environment.sslCipherSuite`			Configuration for SSL connection to MQ Server. See the IBM® MQ documentation for details.
`entirex.wmqbridge.environment.sslFipsRequired`			Configuration for SSL connection to MQ Server. See the IBM® MQ documentation for details.
`entirex.wmqbridge.priority`			Message priority for messages sent to MQ (different from the default priority of the destination queue).

The RPC Server for IBM MQ can be run to

- only send messages to MQ (only output queue specified),
- only receive messages from MQ (only input queue specified), or
- transport messages in both directions (bidirectional communication).

If your programs use arrays of groups, you have to set `entirex.bridge.marshalling` to "Natural" or "COBOL". If your programs do not use arrays of groups, you must not set `entirex.bridge.marshalling`.

Alternatively the RPC data can be transformed to/from XML or SOAP as defined by an XMM mapping file from the XML/SOAP Wrapper. To achieve this, specify the parameter `entirex.bridge.xmm`.

Starting the RPC Server

To start the RPC Server for IBM MQ

- Use the *Start Script*.

  Or:

  Under Windows you can use the RPC Server for IBM MQ as a Windows Service. See *Running the RPC Server as a Windows Service*.

Stopping the RPC Server

To stop the RPC Server for IBM MQ

- Use the command `stopServer`. See Stop EntireX Broker Server Instances in Command Central's Command-line Interface.

  Or:

  Stop the server instance using Command Central's Graphical User Interface. See *Stopping a Server Instance*.

  Or:

  Use the command `stopService`. See Stop Running EntireX Broker Services in Command Central's Command-line Interface.

  Or:

  Stop the service using Command Central's Graphical User Interface. See *Stopping a Service*.

  Or:

  Use the command-line utility `etbcmd`. See `ETBCMD` under *Broker Command-line Utilities* in the platform-specific Administration documentation.

  Or:

  Use `CTRL-C` in the session where you started the RPC server instance.

  Or:

  Under UNIX, enter command `kill -process-id`.

Using SSL/TLS with the RPC Server

To use SSL with RPC Server for IBM MQ, you need to configure two sides:

- **IBM® MQ Side**
  See parameters `entirex.wmqbridge.environment.sslCipherSuite` and `entirex.wmqbridge.environment.sslFipsRequired` under **Configuring the IBM MQ Side**.

- **RPC Server side**

  On the RPC server side you can use Secure Sockets Layer/Transport Layer Security (SSL/TLS) as the transport medium. The term "SSL" in this section refers to both SSL and TLS. On the RPC server side, an SSL client is configured. The SSL server can be either the EntireX Broker, Broker SSL Agent, or Direct RPC in webMethods Integration Server (IS inbound). For an introduction see *SSL/TLS and Certificates with EntireX* in the EntireX Security documentation.

To use SSL

1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.

2 Set up the RPC server side for IBM MQ for an SSL connection.

Use the *URL-style Broker ID* with protocol `ssl://` for the Broker ID. If no port number is specified, port 1958 is used as default. Example:

```
ssl://localhost:22101?trust_store=C:\SoftwareAG\EntireX\etc\ExxCACert.jks&verify_server=no
```

If the SSL client checks the validity of the SSL server only, this is known as *one-way SSL*. The mandatory `trust_store` parameter specifies the file name of a keystore that must contain the list of trusted certificate authorities for the certificate of the SSL server. By default a check is made that the certificate of the SSL server is issued for the hostname specified in the Broker ID. The common name of the subject entry in the server's certificate is checked against the hostname. If they do not match, the connection will be refused. You can disable this check with SSL parameter `verify_server=no`.

If the SSL server additionally checks the identity of the SSL client, this is known as *two-way SSL*. In this case the SSL server requests a client certificate (the parameter `verify_client=yes` is defined in the configuration of the SSL server). Two additional SSL parameters must be specified on the SSL client side: `key_store` and `key_passwd`. This keystore must contain the private key of the SSL client. The password that protects the private key is specified with `key_passwd`.

The ampersand (&) character cannot appear in the password.

SSL parameters are separated by ampersand (&). See also *SSL/TLS Parameters for SSL Clients*.

3 Make sure the SSL server to which the RPC side connects is prepared for SSL connections as well. The SSL server can be EntireX Broker, Broker SSL Agent, or Direct RPC in webMethods Integration Server (IS inbound). See:

- *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation

- *Settting up and Administering the EntireX Broker SSL Agent* in the UNIX and Windows Administration documentation
- *Support for SSL/TLS* in the EntireX Adapter documentation (for Direct RPC)

Running the RPC Server as a Windows Service

For general information see *Running an EntireX RPC Server as a Windows Service*.

To run the RPC Server for IBM MQ as a Windows Service

1 Customize the ***Start Script*** according to your system installation.

Note: The script must pass external parameters to the RPC server and use the reduced signaling of the JVM (option `-Xrs`):

```
java -Xrs com.softwareag.entirex.rpcbridge.WMQBridge %*
```

If `-Xrs` is not used, the JVM stops and an entry 10164002 is written to the event log when the user logs off from Windows.

See also ***Starting the RPC Server***.

2 Test your RPC server to see whether it will start if you run your script file.

3 Use the *EntireX RPC Service Tool* and install the `RPCService` with some meaningful extension, for example `MyServer`. If your ***Start Script*** is *wmqbridge.bat*, the command will be

```
RPCService -install -ext MyServer -script install_path\EntireX\bin\wmqbridge.bat
```

The log file will be called *RPCservice_MyServer.log*.

4 In **Windows Services** menu (**Control Panel** > **Administrative Tools** > **Services**) select the service: `Software AG EntireX RPC Service [MyServer]` and change the property `Startup Type` from "Manual" to "Automatic".

Activating Tracing for the RPC Server

The trace level for the RPC server side is controlled by the usual `entirex.trace` property. Additional diagnostic output can be enabled by setting the property `entirex.server.verbose`.

The trace for the IBM MQ side (MQ-specific diagnostic output) is enabled by setting the property `entirex.bridge.verbose`. In addition, tracing of the IBM MQ classes can be influenced with the property `entirex.wmqbridge.mqtrace`.

Redirect the trace to a file with the property `entirex.server.logfile`. Set this to the file name of the log file, default is standard output.

All properties can be set in the configuration file.

5 Advanced RPC Server for IBM MQ Functionality

- Support for Request/Reply Scenarios 26
- Handling of Correlation ID 26
- Character Encoding Issues 26
- User Exit for Message Processing 27
- Transactional Behavior 28

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 RPC Server for IBM MQ 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 RPC Server for IBM MQ 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).

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.

Handling of Correlation ID

For Request Reply scenarios there is an implicit usage of the correlation ID by the RPC Server for IBM MQ: 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.

Character Encoding Issues

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

If the payload of the MQ message is in XML format (`property entirex.bridge.xmm` has been set), the RPC Server for IBM MQ converts the XML payload to the encoding used for the remote procedure call. If the RPC Server for IBM MQ 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 RPC Server for IBM MQ converts the message to the encoding specified by the CCSID (Coded Character Set IDentification) of the queue manager. When receiving a message, the RPC Server for IBM MQ 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 RPC Server for IBM MQ.

User Exit for Message Processing

IBM® 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 RPC Server for IBM MQ 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 IBM® 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 RPC Server for IBM MQ 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 RPC Server for IBM MQ.

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 RPC Server for IBM MQ (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 RPC Server for IBM MQ 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 RPC Server for IBM MQ.
2. The RPC Server for IBM MQ 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 RPC Server for IBM MQ 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 RPC Server for IBM MQ.
2. The RPC Server for IBM MQ 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 RPC Server for IBM MQ creates a corresponding RPC reply that 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.