SupportPac IAMA

WebSphere Message Broker

Support for TIBCO Rendezvous

Version 1.0.0

30th May, 2013

Note:
Before using this information and the product it supports, read the information in the "Notices" on page iv.

First Edition (May 2013)

This edition applies to version 1.0.0 of IAMA, WebSphere Message Broker Support for TIBCO Rendezvous and to all subsequent releases and modifications until otherwise indicated in new editions.

Table of Contents

Notices

Summary of Amendments

Date	Changes
29 May 2013	Initial release

Bibliography

- WebSphere Message Broker Version 8.0.0.1
  Installation
  IBM Corporation.
- WebSphere Message Broker Version 8.0.0.1
  User-defined Extensions
  IBM Corporation.
- WebSphere Message Broker Version 8.0.0.1
  Java Connector API
  IBM Coporation.

IAMA Overview

SupportPac IAMA supplies WebSphere Message Broker v8 Eclipse plug-in nodes that provide connectivity to TIBCO Rendezvous systems v8.0 or higher. SupportPac IAMA is built using the WebSphere Message Broker Java Connector classes which are new in WMB v8.0.0.1. The Connector classes provide a set of interfaces to easily build "connectivity" to any "provider". SupportPac IAMA provides a "connector provider" – TIBRV -- for integration with TIBCO Rendezvous.

The IAMA plug-in nodes allow message flows built with WebSphere Message Broker v8.0.0.1, or higher, to publish a subject-based message to a Rendezvous system, and to subscribe to messages based on a subject that may contain standard Rendezvous subject wildcards.

The TibRvOutput node allows a message flow to publish a TibrvMsg message using reliable or certified messaging delivery.

TibRvOutput

The TibRvInput node allows a message flow to subscribe to Rendezvous subjects and receive TibrvMsg's using reliable, certified or distributed queue messaging delivery.

TibRvInput

Message Exchange Patterns

IAMA allows messages to be exchanged between a Rendezvous system and a WMB message flow. The Message Exchange patterns provided by IAMA allow any type of TibrvMsg message content to be exchanged (fixed-length, tag delimited, XML), including nested TibrvMsg's. An XML schema – rv.xsd – defines XML syntax (RvXML) that can represent any TibrvMsg structure. On output, an RvXML syntax tree (built using the XMLNSC parser) is translated to a TibrvMsg structure. On input, a TibrvMsg is translated to an RvXML syntax tree using the XMLNSC parser.

The TibRvOutput node allows data to be exchanged between the WMB Message Syntax Tree and a TibrvMsg. And, the TibRvInput node allows data to be exchanged between a

TibrvMsg and the WMB Message Syntax tree. The TibRv Nodes provide the following message exchange patterns between a WMB message flow context and a TibrvMsg:

- **Output**
    - *BLOBToRVBinary*
      The Message Syntax tree is serialized into a bytestream. A single Field in the outbound TibrvMsg of type TIBRVMSG_OPAQUE contains the serialized bytestream. Any WMB parser and meta-data can be used to serialize the Message Syntax tree.
    - *BLOBToRVString*
      The Message Syntax tree is serialized into a string. A single Field in the outbound TibrvMsg of type TIBRVMSG_STRING contains the string. Any WMB parser and meta-data can be used to serialize the Message Syntax tree.
    - *BLOBToRVXML*
      The Message Syntax tree is serialized into a TibrvXml field. A single Field in the outbound TibrvMsg of type TIBRVMSG_XML contains the XML byte array. Any WMB XML parser and meta-data can be used to serialize the XML Message Syntax tree.
    - *XMLToRVMessage*
      The Message Syntax tree must contain a valid RvXML message using the XMLNSC parser. A XML schema – rv.xsd – that defines the RvXML syntax is provided with the SupportPac. The RvXML syntax is a direct mapping to a TibrvMsg, and all TIBRVMSG data types are supported. The Message Syntax tree is not serialized using the XMLNSC parser or meta-data; rather, the TibRvOutput node traverses the simple RvXML format, generating the outbound TibrvMsg in the process. The RvXML syntax is validated by the TibRvOutput node during this process.
- **Input**
    - *RVBinaryToBLOB*
      A single Field in the inbound TibrvMsg of type TIBRVMSG_OPAQUE contains a bytestream that will be parsed according to “Input Message Parsing”, “Parser Options” and “Validation” properties for the node. Any WMB parser and meta-data can be used to parse the bytestream and create the Message Syntax tree.
    - *RVStringToBLOB*
      A single Field in the inbound TibrvMsg of type TIBRVMSG_STRING contains a string that will be parsed according to “Input Message Parsing”, “Parser Options” and “Validation” properties for the node. Any WMB parser and meta-data can be used to parse the string and create the Message Syntax tree.
    - *RVXMLToBLOB*
      A single Field in the inbound TibrvMsg of type TIBRVMSG_XML contains a byte array that will be parsed according to “Input Message Parsing”, “Parser Options” and “Validation” properties for the node. Any WMB XML parser and meta-data can be used to parse the TibrvXml byte array and create the Message Syntax tree.

- *RVMessageToXML*
  The inbound TibrvMsg is processed sequentially creating an RvXML message syntax tree using the XMLNSC parser. A XML schema – rv.xsd – that defines the RvXML syntax is provided with the SupportPac. The RvXML syntax is a direct mapping to a TibrvMsg, and most TIBRVMSG types are supported (Inbound Unsigned Arrays are not supported). An XML byte stream is not created and parsed by the XMLNSC parser; rather, the TibRvInput node creates the simple RvXML syntax tree, using the XMLNSC parser and WMB Plugin Node API, as it sequentially processes each Field in the TibrvMsg.

Administration Commands

The TibRvInput and TibRvOutput nodes provide access to Rendezvous administration commands using the CMP API. IAMA provides a Java utility – ExecIAMAAdminCommands.jar -- to easily execute these node administration commands.

Both the TibRvInput and TibRvOutput nodes provide an administration command to enable/disable/modify Advisory Message Logging. In addition, the TibRvOutput node provides numerous "ledger file" administration commands when certified messaging is enabled. The following administration commands are provided to review and modify the ledger file associated with the TibRvOutput node:

- **reviewLedger** – Executes the reviewLedger command. Output from this command is written to the TIBRV resource manager's Activity Log where it can be viewed and filtered using MBX.
- **removeListener** – Executes the removeListener command on the CM correspondent's ledger file. Cancel certified delivery of the specified subject to the specified CM correspondent. Information about this command is written to the TIBRV resource manager's Activity Log and can be viewed using MBX.
- **disallowListener** – Executes the disallowListener command on the CM correspondent's ledger file. Cancel certified delivery to all listeners of the specified CM correspondent. Information about this command is written to the TIBRV resource manager's Activity Log and can be viewed using MBX.
- **allowListener** – Executes the allowListener command on the CM correspondent's ledger file. The specified CM correspondent can reinstate certified delivery for all its listeners overriding the effects of a previous *disallowListener* command. Information about this command is written to the TIBRV resource manager's Activity Log and can be viewed using MBX.
- **expireMessages** – Executes the expireMessages command on the CM correspondent's ledger file. Mark messages that match the specified "subject" and "sequence number" as expired. Information about this command is written to the TIBRV resource manager's Activity Log and can be viewed using MBX.
- **removeSendState** – Executes the removeSendState command on the persistent CM correspondent's ledger file. Recover ledger space for the specified obsolete

"subject." Information about this command is written to the TIBRV resource manager's Activity Log and can be viewed using MBX.

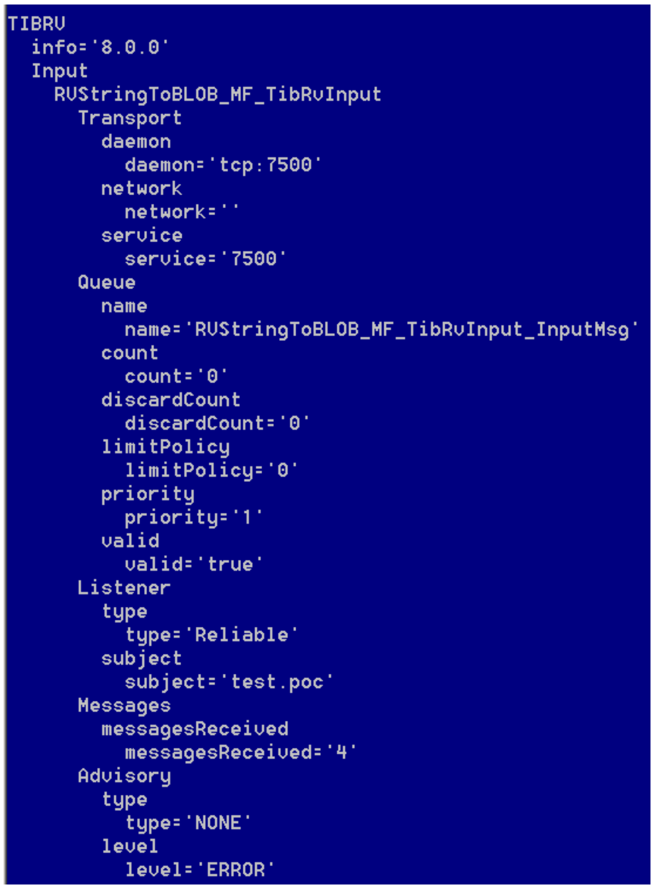
- NOTE: These admin commands must be executed against a deployed and running TibRvOutput node

Both the TibRvInput and TibRvOutput nodes support the following administration command to modify the node's Advisory Message Logging behavior:

- **advisory** – Modifies Rendezvous Advisory Message logging behavior for an TibRv node. The advisoryType (SYSTEM,CM,ALL,NONE) and advisoryLevel(INFO,WARN,ERROR,ALL) can be set for TibRvInput and TibRvOutput nodes.
- NOTE: The admin commands must be executed against a deployed and running TibRvOutput or TibRvInput node

TibRv Node Status

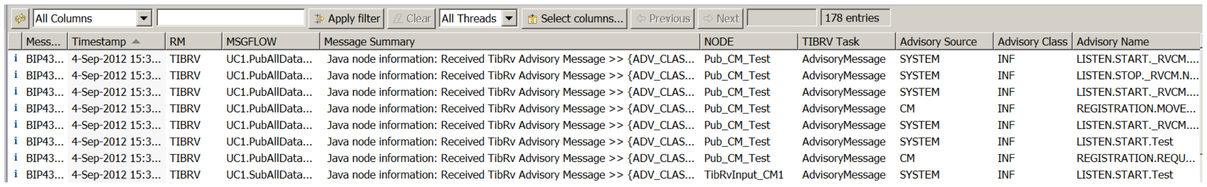
The TibRvInput and TibRvOutput node status can be displayed using the mqsireportproperties command. The command can be configured to display information for all deployed TibRv nodes or specific TibRvInput or TibRvOutput nodes.



```
TIBRV
  info='8.0.0'
  Input
    RVStringToBLOB_MF_TibRvInput
      Transport
        daemon
          daemon='tcp:7500'
        network
          network=''
        service
          service='7500'
      Queue
        name
          name='RVStringToBLOB_MF_TibRvInput_InputMsg'
        count
          count='0'
        discardCount
          discardCount='0'
        limitPolicy
          limitPolicy='0'
        priority
          priority='1'
        valid
          valid='true'
      Listener
        type
          type='Reliable'
        subject
          subject='test.poc'
      Messages
        messagesReceived
          messagesReceived='4'
      Advisory
        type
          type='NONE'
        level
          level='ERROR'
```

Rendezvous Advisory Message Logging

The TibRvInput and TibRvOutput nodes are configurable to subscribe to Rendezvous Advisory Messages for their underlying Rendezvous transport. The Advisory Messages that are received are written to the TIBRV resource manager Activity Log. The Message Broker Explorer (MBX) can be used to view and filter the Rendezvous Advisory messages.



	Mess...	Timestamp	RM	MSGFLOW	Message Summary	NODE	TIBRV Task	Advisory Source	Advisory Class	Advisory Name
i	BIP43...	4-Sep-2012 15:3...	TIBRV	UC1.PubAllData...	Java node information: Received TibRv Advisory Message >> {ADV_CLAS...	Pub_CM_Test	AdvisoryMessage	SYSTEM	INF	LISTEN.START._RVCM....
i	BIP43...	4-Sep-2012 15:3...	TIBRV	UC1.PubAllData...	Java node information: Received TibRv Advisory Message >> {ADV_CLAS...	Pub_CM_Test	AdvisoryMessage	SYSTEM	INF	LISTEN.STOP._RVCM.N...
i	BIP43...	4-Sep-2012 15:3...	TIBRV	UC1.PubAllData...	Java node information: Received TibRv Advisory Message >> {ADV_CLAS...	Pub_CM_Test	AdvisoryMessage	SYSTEM	INF	LISTEN.START._RVCM....
i	BIP43...	4-Sep-2012 15:3...	TIBRV	UC1.PubAllData...	Java node information: Received TibRv Advisory Message >> {ADV_CLAS...	Pub_CM_Test	AdvisoryMessage	CM	INF	REGISTRATION.MOVE...
i	BIP43...	4-Sep-2012 15:3...	TIBRV	UC1.PubAllData...	Java node information: Received TibRv Advisory Message >> {ADV_CLAS...	Pub_CM_Test	AdvisoryMessage	SYSTEM	INF	LISTEN.START._RVCM....
i	BIP43...	4-Sep-2012 15:3...	TIBRV	UC1.PubAllData...	Java node information: Received TibRv Advisory Message >> {ADV_CLAS...	Pub_CM_Test	AdvisoryMessage	SYSTEM	INF	LISTEN.START.Test
i	BIP43...	4-Sep-2012 15:3...	TIBRV	UC1.PubAllData...	Java node information: Received TibRv Advisory Message >> {ADV_CLAS...	Pub_CM_Test	AdvisoryMessage	CM	INF	REGISTRATION.REQU...
i	BIP43...	4-Sep-2012 15:3...	TIBRV	UC1.SubAllData...	Java node information: Received TibRv Advisory Message >> {ADV_CLAS...	TibRvInput_CM1	AdvisoryMessage	SYSTEM	INF	LISTEN.START.Test

The IAMA administration command – advisory -- provides a means of modifying Rendezvous Advisory Message logging for deployed TibRv nodes. This allows Advisory Message logging to be enabled only when required.

Certified Message Confirmation

Certified Messages received by a TibRvInput node must be confirmed. The TibRvInput node allows certified messages to be confirmed using different confirmation modes.

The TibRvInput Node provides three confirmation modes when using certified messaging:

- **Auto** – This is the default. The certified message is confirmed when received from Rendezvous and before the message flow is invoked.
- **OnSuccess** – The certified message is confirmed if the message flow completes successfully. The certified message will not be confirmed if the message flow rolls-back.
- **OnSuccessAndFailure** – The certified message is confirmed if the message flow completes successfully or if the message flow rolls-back. This allows the message flow to confirm a certified message in scenarios where the message flow must roll-back other operations such as database updates and messages put to MQ queues.

Receiving Previously Sent, Unconfirmed Messages

When using certified messaging, the TibRvInput node can specify the “RequestOld” option to receive certified messages that were previously sent to a CM correspondent with the same name but have not yet been confirmed. This allows the subscriber to receive all “old” messages bringing its last confirmed Sequence Number up-to-date with the publisher.

Preregister Listeners

When using certified messaging, the TibRvOutput node can specify a list of anticipated Listeners to add to its ledger file. All matching publications are saved in the ledger even though the Listener may not be active. When the preregistered Listener becomes active, it will receive the saved publications.

In addition the TibRvOutput node can preregister the cancelation of certified delivery to a list of CM correspondents. Any Listener from the CM correspondent list will be disallowed.

File or Memory-Based Ledger File

The TibRvInput and TibRvOutput nodes use a "ledger" when Certified Message Delivery is enabled. The node's "ledger" can be file-based or memory-based. A "file-based" ledger can be reused if the message flow restarts, while a "memory-base" ledger cannot survive a message flow restart.

If a "file-based" ledger file is used, the WMB broker must have the correct permissions to create and access the ledger file, in addition the ledger file should reside on a local file system.

TibRvOutput Node LocalEnvironment

The TibRvOutput Node supports override properties in the LocalEnvironment tree. The outbound Subject and Data Exchange Type (BLOBToRVBinary, BLOBToRVString, BLOBToRVXML, XMLToRVMessage):

- OutputLocalEnvironment.Destination.TIBRV.Output.subject
- OutputLocalEnvironment.Destination.TIBRV.Output.rvMessageType

The TibRvOutput Node provides written destination information for successfully published TibrvMsg:

- LocalEnvironment.WrittenDestination.TIBRV.byteSize – number bytes used by the TibrvMsg
- LocalEnvironment.WrittenDestination.TIBRV.numFields – number of Fields in the TibrvMsg
- LocalEnvironment.WrittenDestination.TIBRV.subject – the Send subject of the TibrvMsg
- NOTE: all fields are of type CHAR

TibRvInput Node LocalEnvironment

The TibRvInput node provides information, in the LocalEnvironment tree regarding the inbound reliable or certified TibrvMsg.

The following TibrvMsg information is provided in the LocalEnvironment syntax tree for reliable messages that are received by the TibRvInput node:

- LocalEnvironment.TIBRV.Input.isCM – "true" if the TibrvMsg is a certified message and "false" if the TibrvMsg is a reliable message.
- LocalEnvironment.TIBRV.Input.msgByteSize – The TibrvMsg size in bytes
- LocalEnvironment.TIBRV.Input.numFields – Number of fields in the TibrvMsg
- LocalEnvironment.TIBRV.Input.subject – The "send" subject of the TibrvMszg
- NOTE: all fields are of type CHAR

The following TibrvMsg information is provided in the LocalEnvironment syntax tree for certified messages that are received by the TibrvInput node:

- LocalEnvironment.TIBRV.Input.cmConfirmType – The type of CM confirmation being used for this TibrvMsg (Auto, OnSuccess, OnSuccessAndFailure)
- LocalEnvironment.TIBRV.Input.cmSender – The name of the certified correspondent that published this TibrvMsg
- LocalEnvironment.TIBRV.Input.isCM – “true”
- LocalEnvironment.TIBRV.Input.msgByteSize – The TibrvMsg size in bytes
- LocalEnvironment.TIBRV.Input.numFields – Number of fields in the TibrvMsg
- LocalEnvironment.TIBRV.Input.seqNum – Sequence number of the TibrvMsg
- LocalEnvironment.TIBRV.Input.subject – The “send” subject of the TibrvMsg
- NOTE: all fields are of type CHAR

Service and User Trace

The TibRvInput and TibRvOutput nodes send trace information to an Execution Group Service trace and normal User trace. Use mqsichangetrace and mqsireadlog/mqsiformatlog commands to capture TibRvInput and TibRvOutput trace information.

TibrvMsg String Encoding

Inbound TibrvMsg’s will convert TIBRVMSG_STRING fields to Unicode strings using the codepage tag if present. If the inbound TibrvMsg doesn’t contain a codepage tag then the “default” system codepage is used. Outbound TibrvMsg’s will convert Unicode strings to TIBRVMSG_STRING types using the “default” codepage. The “default” value can be specified per Execution Group via the TIBRV Configurable Service Definition, “property1” property.

The “property1” property specifies a comma-separated list of ExecutionGroup/Codepage pairs:

- property1 -- EG1=1250,EG2=1252,EG3=819
- property1 – EG1=ISO-8859-15,EG2=UTF-8

This setting affects all TibRvInput and TibRvOutput nodes that are deployed in an Execution Group and restricts TibRvInput/TibRvOutput nodes within an Execution Group to a single codepage for string fields.

IAMA Installation

Extract the contents of the IAMA.zip archive to a temporary location. The following sections will describe the installation of contents from the temporary location. Once your installation is complete, you may remove the contents of the temporary location.

IAMA SupportPac Contents

The supplied zip file, IAMA_1.0.0.zip, should be unzipped to a temporary directory. The following subdirectories and files will be created in an IAMA directory:

connectortibrv_1.0.0.jar – This is the TIBRV Connector provider implementation and must be installed on all WMB 8.0.0.1 installations that will host TibRv nodes

TibRvNode_1.0.0.jar – This is the TibRv Eclipse Plugin that must be installed on all WMB v8 Toolkits that will be used to build message flows containing TibRv nodes.

ExecIAMAAdminCommands.jar – This executable JAR file provides access to the TibRv Node administration commands via the CMP api.

IAMA_Sample_PI.zip – This Project Interchange contains the artifacts to install and run the sample message-flow.

rv.xsd – This XML schema defines the RvXML syntax that is used to model TibRvMsg structures/objects in XML.

IAMA.pdf – The IAMA Users Guide (this document).

Prerequisites

IAMA has the following software prerequisites:

- WMB v8.0.0.1 or higher
- WMB Toolkit v8.0 or higher
- WMB Explorer v8.0.0.1 or higher
- TIBCO Rendezvous v8.0 or higher
- Note: If you are using a 32bit broker you must use a 32bit Rendezvous installation. If you are using a 64bit broker you must use a 64bit Rendezvous installation.

Supported Platforms

IAMA SupportPac is supported on the following platforms:

- Any supported WMB v8.0.0.1platform that also supports TIBCO Rendezvous v8.0 or higher

- Windows XP
- Windows 7
- Linux
- AIX
- Solaris

Installing the IAMA Connector Provider on the Broker system

Perform the following tasks on each runtime broker that will host the TibRv nodes:

All Platforms:

1. Create a directory to hold the IAMA Connector Provider JAR file on the system which you will install the IAMA Connector Provider
2. Put the connectortibrv_1.0.0.jar file in this directory. Make sure that this file has read and execute permission for the user id the broker will run under.
3. **CAUTION: Do not put the connectortibrv_1.0.0.jar file under the WebSphere Message Broker "classes" directory, as this can cause problems loading the TIBRV connector provider.**
4. Validate that the CLASSPATH system environment variable does not contain an entry for tibrvnative.jar.
5. Validate that the PATH system environment variable contains a path to the "bin" directory of the Rendezvous installation. NOTE: rvd.exe must be executable by the user id the broker will run under.
6. Create and start the Broker that the TibRv nodes will be deployed to.
7. Create a configurable service on the broker based on the ConnectorsProvider template with the following property settings:
    - Name = TIBRV
    - Type = ConnectorProviders
    - connectorClassName = com.ibm.broker.connector.tibrv.TibRvConnectorFactory
    - jarsURL = {Path to directory containing connectortibrv_1.0.0.jar};{Path to tibrvnative.jar under the Revdezvous installation}

      NOTE: On Windows systems these paths are separated by a ";", and on Linux/Unix systems these paths are separated by a ":".

      Windows: C:\IBM\IAMA;C:\TIBCO\tibrv\8.2.2\lib\tibrvnative.jar
      Linux/Unix: /opt/ibm/iama:/opt/tibcorv/8.2.2/lib/tibrvnative.jar

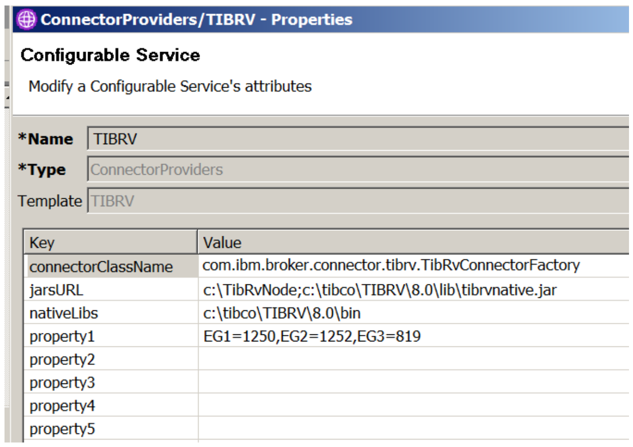
    - nativeLibs = Path to directory under the Rendezvous installation that contains the tibrv .dll files (Windows) or tibrv .so files (Unix/Linux).

      NOTE: This directory contains the Rendezvous native libraries (.dll on Windows or .so on Linux/Unix) that are required by the TIBRV provider. Typically, on Windows this is the "bin" directory under the Rendezvous installation, and on Unix/Linix this is the “lib” directory under the Rendezvous installation.

- NOTE: Override the “default” tibrv string encoding codepage for an Execution Group using “property1”. “property1” is a comma-separated list of EGName=Codepage pairs. (Example: EG1=1250,EG2=1252,EG3=819). The string encoding codepage specified in the TIBRV configurable service affects both TibRvInput and TibRvOutput nodes. See the TIBCO Rendezvous Concepts manual for details regarding the tibrv string encoding codepage setting.

- NOTE: The TIBRV ConnectorProviders configurable service definition can be created using the mqsicreateconfigurableservice command or by using MBX, and can be modified using the mqsichangproperties command. You must recycle the broker for changes to the TIBRV ConnectorProviders configurable service to take effect.

  ```
  mqsicreateconfigurableservice BROKER -c ConnectorProviders -o TIBRV -n " \"connectorClassName\", \"jarsURL\", \"nativeLibs\", \"property1\", \"property2\", \"property3\", \"property4\", \"property5\" " -v " \"com.ibm.broker.connector.tibrv.TibRvConnectorFactory\", \"c:\TibRvNode;c:\tibco\TIBRV\8.0\lib\tibrvnative.jar\", \"c:\tibco\TIBRV\8.0\bin\", \"EG1=1250,EG2=1252,EG3=819\", \"\", \"\", \"\", \"\" "
  ```



ConnectorProviders/TIBRV - Properties

Configurable Service

Modify a Configurable Service's attributes

*Name: TIBRV
*Type: ConnectorProviders
Template: TIBRV

Key	Value
connectorClassName	com.ibm.broker.connector.tibrv.TibRvConnectorFactory
jarsURL	c:\TibRvNode;c:\tibco\TIBRV\8.0\lib\tibrvnative.jar
nativeLibs	c:\tibco\TIBRV\8.0\bin
property1	EG1=1250,EG2=1252,EG3=819
property2	
property3	
property4	
property5	

8. On Linux and UNIX systems, validate that the LD_LIBRARY_PATH system environment variable contains a path to the “bin” directory under the Rendezvous installation.

   NOTE: On AIX validate LIBPATH instead of LD_LIBRARYPATH
9. Create an Execution Group on the Broker
10. Stop the Broker

11. Start the Broker and validate there are no exception messages associated with loading the TIBRV connector provider in the broker's log file.
12. Validation. Deploy and run the IAMA Sample message flows if validation is required.

Uninstalling IAMA from Runtime Broker

Perform the following tasks on each runtime broker that you which to uninstall the TibRv nodes from:

1. Remove all flows containing TibRv nodes from the Broker's Execution Groups
2. Remove the TIBRV Configurable Service Definition

   ```
   mqsideleteconfigurableservice BROKER –c ConnectorProvider –o TIBRV
   ```

   NOTE: MBX can also be used to delete the TIBRV Configurable Service Definition.
3. Stop the broker.
4. Delete the connectortibrv_1.0.0.jar file and the directory containing it.
5. Delete the ExecIAMAAdminCommands.jar file if it was installed on the broker system.
6. Start the broker.
7. Validate that no TIBRV connector provider error messages are written to the broker's log file

Integrating the IAMA Eclipse plug-in into the Message Broker Toolkit

The IAMA SupportPac provides Eclipse plug-ins for the following versions of the Message Broker Toolkit:

- IBM WebSphere Message Broker Toolkit version v8.0 or higher
- Note: The TibRv nodes can only be deployed to a WMB v8.0.0.1 or higher installation.

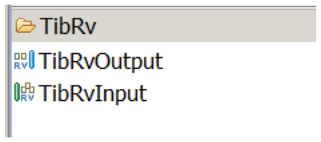
The IAMA Eclipse plug-ins must be installed on each WMB v8.0 Toolkit that will use the TibRv nodes.

For Message Broker Toolkit 8.0, copy the TibRvNode_1.0.0.jar file to the following toolkit directory:

- <wmbtk_root>\WMBT800\plugins, where <wmbtk_root> is the root installation directory for your Message Broker Toolkit v8.0.
- <wmbtk_root>\WMBT8001\plugins, where <wmbtk_root> is the root installation directory for your Message Broker Toolkit v8.0.0.1

You must stop and restart the Message Brokers Toolkit for the TibRv nodes to be available on the plug-in palette in the Message Flow Editor.

The TibRv nodes are contained in the TibRv plug-in category on the node palette.



Uninstalling the IAMA Eclipse plug-in from the WMB Toolkit

Perform the following tasks to uninstall the IAMA Eclipse plug-in from a WMB v8.0 Toolkit:

- Stop The WMB v8.0 Toolkit
- Delete the TibRvNode_1.0.0.jar file from the following toolkit directory:
- <wmbtk_root>\WMBT800\plugins, where <wmbtk_root> is the root installation directory for your Message Broker Toolkit v8.0.
- <wmbtk_root>\WMBT8001\plugins, where <wmbtk_root> is the root installation directory for your Message Broker Toolkit v8.0.0.1
- Start the WMB Toolkit
- Validate that the TibRv node category is no longer displayed in the Message Flow Editor

Using the TibRvInput Node

TibRvInput Node Description

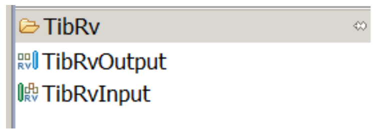
The TibRvInput node provides connectivity to a TIBCO Rendezvous system allowing the node to receive TibrvMsgs based on a Subject that may contain wildcards.

The TibRvInput node provides three messaging delivery modes:

- Reliable Messaging – Uses a TibrvRvdTransport
- Certified Messaging – Uses a TibrvCmTransport
- Distributed Queue Messaging – Uses a TibrvCmQueueTransport

TibRv Plug-in category

When supportpac IAMA is installed, the TibRvInput node will appear in the "TibRv" plug-in category in the Message Brokers Toolkit's Message-Flow editor.



TibRvInput Plug-in node terminals

Terminal	Description
Failure	The output terminal that the message assembly is routed to if a failure occurs in the node. The ExceptionList syntax tree will contain a description of the exception that caused the node failure. The TibrvMsg byte stream is contained under the BLOB parser. This byte stream can be saved and read as a TibrvMsg by a

	utility program.
Out	The output terminal that emits a message assembly containing the inbound TibrvMsg. Note: The content of the message syntax tree depends on the message exchange pattern being used and the Input Message Parsing settings. The LocalEnvironment syntax tree will contain information about the inbound TibrvMsg.
Catch	The output terminal that the message assembly is routed to if an unhandled message flow exception reaches the TibRvInput node. The ExceptionList syntax tree will contain a description of the exception that caused the message flow to rollback.

TibRvInput Node Message Exchange

The TibRvInput node provides the following message exchange patterns between the inbound TibRvMsg and a message flow:

- **RVBinaryToBLOB** – The inbound TibrvMsg contains a single field of type TIBRVMSG_OPAQUE that contains the message content that will be parsed using the properties set on the "TibRvMessage", "Input Message Parsing", "Validation" and "Parser Options" property pages.
- **RVStringToBLOB** – The inbound TibRvMsg contains a single field of type TIBRVMSG_STRING that contains the message content that will be parsed using the properties set on the "TibRvMessage", "Input Message Parsing", "Validation" and "Parser Options" property pages.
- **RVXMLToBLOB** – The inbound TibrvMsg contains a single field of type TIBRVMSG_XML that contains the XML byte stream that will be parsed using the preperties set on the "TibRvMessage", "Input Message Parsing", "Validation" and "Parser Options" property pages.
- **RVMsgToXML** – The inbound TibrvMsg is processed sequentially and converted to RvXML syntax defined by the rv.xsd schema. The XMLNSC parser is used to build the RvXML representation of the inbound TibrvMsg.

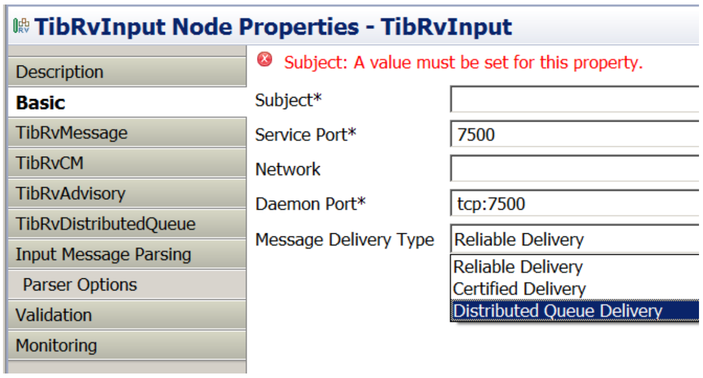
TibRvInput Node Properties

The TibRvInput node properties are displayed on the following Property Pages:

- **Basic** – Transport, Subject and Messaging Mode
- **TibRvMessage --** Message Exchange Pattern between the TibrvMsg and message flow
- **TibRvCM** – Certified Messaging
- **TibRvAdvisory** – Rendezvous Advisory Messages
- **TibRvDistributedQueue** – Distributed Queue Messaging
- **Input Message Parsing** – Message Domain, Set, Type, Format
- **Parser Options** – Parser Timing, Opaque Elements
- **Validation** – Validate and Failure Action
- **Monitoring** – Configure WMB Monitoring Events

TibRvInput Node Basic Group

The Basic property page defines the Rendezvous connectivity parameters, subject and message delivery type.


TibRvInput Node Properties - TibRvInput

Subject: A value must be set for this property.

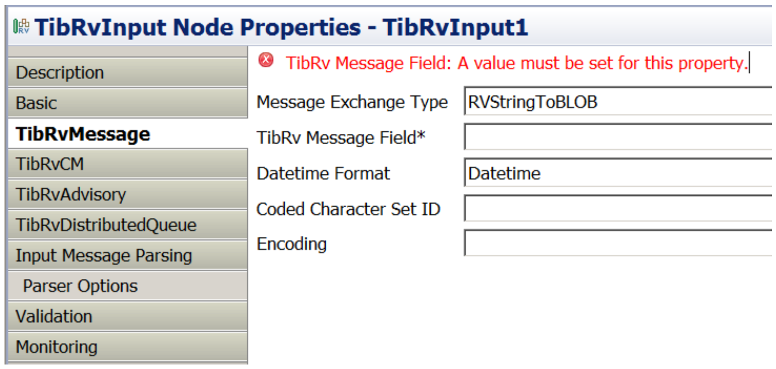
Subject*
Service Port* 7500
Network
Daemon Port* tcp:7500
Message Delivery Type Reliable Delivery

Property	M	C	Default	Description
Subject	Y	Y	Cleared	Subject specifies the messages to be received from Publishers in the Rendezvous system. Wild cards can be used to listen for multiple Subjects. See TIBCO Rendezvous Concepts manual for full description of Subject syntax.
Service Port	Y	Y	7500	This is the port number that will be used to listen for Rendezvous messages.

				See TIBCO Rendezvous Concepts manual for full description of the Service Port property.
Network	N	N	Cleared	The network interface that will be used for outbound messages published by this node. If no value is specified then the primary network interface for the host will be used. See TIBCO Rendezvous Concepts manual for full description of the Network property.
Daemon Port	Y	Y	tcp:7500	This is the port that the node's transport will use to communicate with the RV daemon. See TIBCO Rendezvous Concepts manual for full description of the Daemon Port property.
Message Delivery Type	Y	N	Reliable Delivery	Specifies the messaging delivery mode to be used by the node. Reliable – TibrvRvdTransport is used Certified – TibrvCmTransport is used Distributed Queue -- TibrvCmQueueTransport is used

TibRvInput Node TibRvMessage Group:

The TibRvMessage property page defines the message exchange pattern that will be used between inbound TibrvMsg's and the message syntax tree generated by the TibRvInput node.


TibRvInput Node Properties - TibRvInput1

TibRv Message Field: A value must be set for this property.

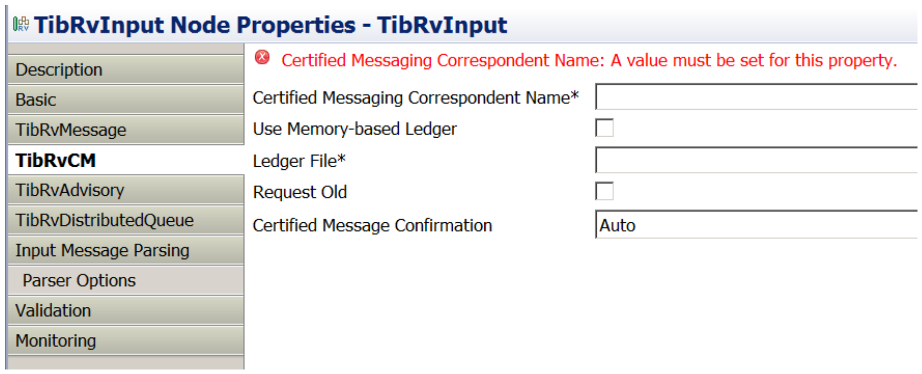
Message Exchange Type	RVStringToBLOB
TibRv Message Field*	
Datetime Format	Datetime
Coded Character Set ID	
Encoding	

Property	M	C	Default	Description
Message Exchange Type	Y	N	RVStringToBLOB	Specifies how the TibrvMsg content will be exchanged with the WMB Message Flow (RVStringToBLOB, RVBinaryToBLOB, RVXMLToBLOB, RVMessageToXML) RVStringToBLOB, RVBinaryToBLOB and RVXMLToBLOB must specify a name for the TibRv Message Field property that contains content to be processed.
TibRv Message Field	Y	N	Cleared	Specifies the name of the Field in the inbound TibrvMsg that contains the content to be processed. Must be specified if Message Exchange Type is RVStringToBLOB, RVBinaryToBLOB or RVXMLToBLOB. This property is disabled when Message Exchange Type is RVMessageToXML
Datetime Format	Y	N	Datetime	Only applies if Message Exchange

				Type is RVMessageToXML Specifies the RvXML syntax that will be used for fields of type TIBRVMSG_DATETIME. **Datetime** – format of data will be YYYYMMDDTHH:MM:SS.ssss **Sec/Nsec** – The Field element will contain two children tags -- <Sec> the number of seconds and <Nsec> the number of nanoseconds: <Field name=”DTField” type=TIBRVMSG_DATETIME> <Sec>774563</Sec> <Nsec>127836</Nsec> </Field>
Coded Character Set ID	N	N	Cleared	This property specifies the CCSID that will be used to parse character content as defined by “Input Message Parsing” properties. If left blank the default Broker CCSID will be used. NOTE: This property is not implemented in release 1.0.0. A work-around is to specify the CCSID in the Parse clause of a Create statement.
Encoding	N	N	Cleared	This Property specifies the Encoding that will be used to parse binary content as defined by “Input Message Parsing” properties. If left blank the default Broker Encoding will be used. NOTE: This property is not implemented in release 1.0.0. A work-around is to specify the Encoding in the Parse clause of a Create statement

TibRvInput Node TibRvCM Group:

The TibRvCM property page is only active when the Message Delivery Type is Certified Delivery.


TibRvInput Node Properties - TibRvInput

Certified Messaging Correspondent Name: A value must be set for this property.

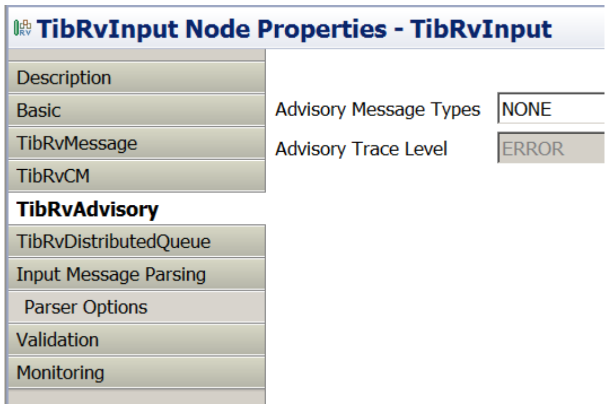
Certified Messaging Correspondent Name*	
Use Memory-based Ledger	☐
Ledger File*	
Request Old	☐
Certified Message Confirmation	Auto

Property	M	C	Default	Description
Certified Messaging Correspondent Name	Y	Y	Cleared	The CM correspondent name must be specified and must be non-null and cannot be an empty string (all spaces). The CM correspondent name must be unique as the CM name is bound to the TibrvCmTransport that is used by the TibRvInput node. See TIBCO Rendezvous Concepts manual for full description of the Certified Messaging Correspondent Name property.
Use Memory-based Ledger	N	N	Unchecked	When checked, the node will use a memory-based ledger and not a file-based ledger. The Ledger File property is disabled when this property is checked.
Ledger File	Y	Y	Cleared	Specifies the ledger file to be used by the TibRvInput node. The ledger file must exist on a local file system. The broker must have permission to create/read/write files in the named directory. If the ledger file already exists for the same CM name, it will be reused. If the ledger file does not exist it will be created.

				This property is disabled when Use Memory-based Ledger is checked.
Request Old	N	N	Unchecked	Specifies, at startup, if the CM correspondent requires delivery of certified messages, previously sent to a CM correspondent with the same name, that have not yet been confirmed. If checked, all unconfirmed certified messages, previously sent to a CM correspondent with the same name, will be delivered to this TibRvInput node.
Certified Message Confirmation	Y		Auto	Specifies how certified TibrvMsgs will be confirmed. Auto – This is the default. The TibrvMsg will is automatically confirmed when received from Rendezvous. This occurs before the message flow in invoked. ConfirmOnSuccess – The TibrvMsg is confirmed if the message flow completes successfully. ConfirmOnSuccessAndFailure – The TibrvMsg is confirmed if the message flow completes successfully and if the message flow rolls-back.

TibRvInput Node TibRvAdvisory Group:

The TibRvAdvisory property page allows the TibRvInput node to log Rendezvous Advisory messages associated with the underlying Rendezvous transport. Advisory messages are written to the TIBRV Activity Log under the Execution Group the node is deployed to.



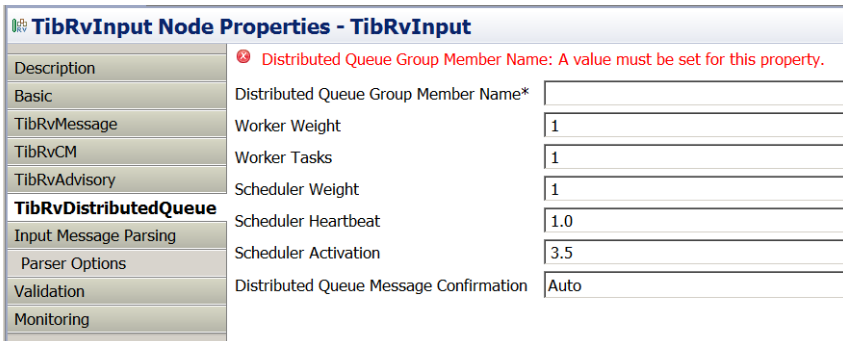
Advisory Message Types NONE
Advisory Trace Level ERROR

Property	M	C	Default	Description
Advisory Message Types	Y	N	NONE	• NONE – Advisory Messages are not logged. • SYSTEM – SYSTEM Advisory Messages are logged. • CM – Certified Messaging Advisory Messages are logged. • ALL – SYSTEM and CM Advisory Messages are logged.
Advisory Trace Level	N	N	ERROR	• ERROR – Only ERROR Advisory Messages are logged. • WARN – Only WARN Advisory Messages are logged. • INFO – Only INFO Advisory Messages are logged. • ALL – ERROR/WARN/INFO Advisory messages are logged. This property is disabled when Advisory Message Types is NONE.

TibRvInput Node TibRvDistributedQueue Group:

The TibRvDistributedQueue property page is only enabled when the Message Delivery Types is Distributed Queue Delivery.

Distributed Queue Delivery allows messages received by a listener to be distributed across members of a group. Typically, each group member is a separate operating system process. Each distributed queue group member must use the same identical name.


TibRvInput Node Properties - TibRvInput

Distributed Queue Group Member Name: A value must be set for this property.

Distributed Queue Group Member Name*	
Worker Weight	1
Worker Tasks	1
Scheduler Weight	1
Scheduler Heartbeat	1.0
Scheduler Activation	3.5
Distributed Queue Message Confirmation	Auto

Property	M	C	Default	Description
Distributed Queue Group Member Name	Y	Y		Specifies the name of the Distributed Queue Group that this TibRvInput node is a member of. This name must be unique amongst distributed queue group names. All members of the Distributed Queue Group must use the exact same name. See TIBCO Rendezvous Concepts manual for full description of the Distributed Queue Group Member Messaging Name property.
Worker Weight	N	N	1	The scheduler assigns tasks to the available worker with the greatest worker weight.
Worker Tasks	N	N	1	This property defines the maximum number of tasks that a worker can accept.
Scheduler Weight	N	N	1	Scheduler Weight defines a member's ability to fulfill the role of

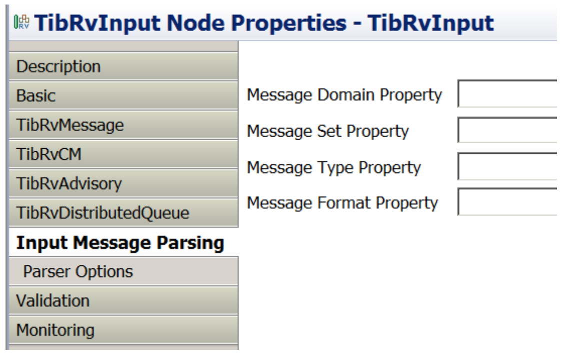
				scheduler, relative to other members within the same group.
Scheduler Heartbeat	N	N	1.0	This property defines the interval (in seconds) that the scheduler sends heartbeat messages.
Scheduler Activation	N	N	3.5	If the scheduler has not sent a heartbeat for this interval (in seconds), the group member with the greatest scheduler weight takes becomes the new scheduler.
Distributed Queue Message Confirmation	Y	N	Auto	Auto – This is the default. The TibrvMsg will is automatically confirmed when received from Rendezvous. This occurs before the message flow in invoked. ConfirmOnSuccess – The TibrvMsg is confirmed if the message flow completes successfully. ConfirmOnSuccessAndFailure – The TibrvMsg is confirmed if the message flow completes successfully and if the message flow rolls-back.

TibRvInput Node Input Message Parsing Group:

The Input Message Parsing property page defines the meta-data that will be used to parse (and validate) content from the inbound TibrvMsg.

These properties only apply when Message Exchange Type is RVStringToBLOB, RVBinaryToBLOB or RVXMLToBLOB. The meta-data defined here will be used to parse the content contained in the RV Message Field and construct the output message syntax tree.

If these properties are left blank the BLOB parser will be used.



Message Domain Property
Message Set Property
Message Type Property
Message Format Property

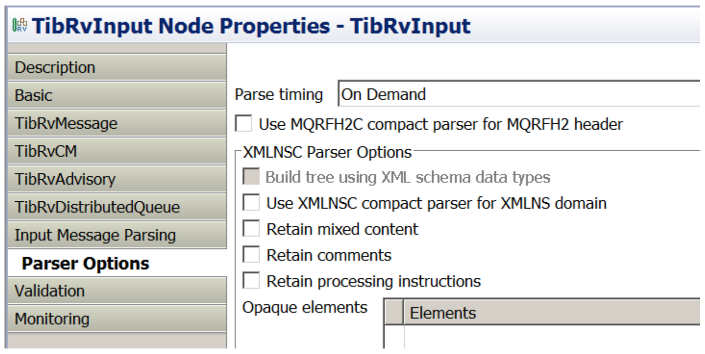
Property	M	C	Default	Description
Message Domain	N	N	Cleared	The domain that is used to parse the message. If the field is blank then the default is BLOB.
Message Set	N	N	Cleared	The name or location of the message model schema file in which the message is defined. This list is populated with all available message model schema files for the Message domain that you have selected.
Message Type	N	N	Cleared	The name or location of the message root within your message model schema file. This list is populated with all available messages that are defined in the Message model that you have selected.
Message Format	N	N	Cleared	The name of the physical format of the message. If you are using the MRM or IDOC parser, select the physical format of the incoming

				message from the list. This list includes all the physical formats that you have defined for the selected message model. If you set the Message domain property to DataObject, you can set this property to XML or SAP ALE IDoc. Set this property to SAP ALE IDoc when you have to parse a bit stream from an external source and generate a message tree.

NOTE: The default Broker CCSID and Encoding will be used to parse TibrvMsg content as defined by the "Input Message Parsing" properties. If the TibrvMsg character content uses a different codepage then that codepage must be specified for the Codepage property on the "TibRvMessage" property page. If TibrvMsg binary content uses a different encoding then that encoding must be specified for the Encoding property on the "TibRvMessage" property page. NOTE: These properties are not implemented in release 1.0.0. A work-around is to specify the CCSID and/or Encoding in the Parse clause of a Create statement.

TibRvInput Node Parser Options Group:

Properties on the Parser Options property page only apply when Message Exchange Type is RVStringToBLOB, RVBinaryToBLOB or RVXMLToBLOB.



Parse timing: On Demand
☐ Use MQRFH2C compact parser for MQRFH2 header
XMLNSC Parser Options
☐ Build tree using XML schema data types
☐ Use XMLNSC compact parser for XMLNS domain
☐ Retain mixed content
☐ Retain comments
☐ Retain processing instructions
Opaque elements: Elements

Property	M	C	Default	Description
Parse Timing	N	N	OnDemand	This property controls when an input message is parsed. Valid values are On Demand, Immediate,

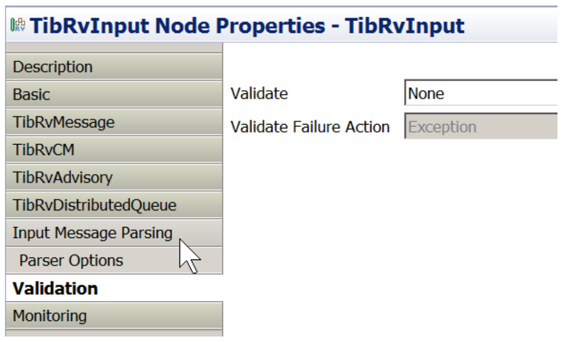
Property	M	C	Default	Description
				and Complete. Parse timing is, by default, set to On Demand, which causes parsing of the message to be delayed. To fully parse the message at the TibRvInput node use Complete.
Use MQRFH2C	N	N	Cleared	This property has no affect as no MQRFH2 header is present.
Build tree using XML schema data types	N	N	Cleared	This property controls whether the XMLNSC parser creates syntax elements in the message tree with data types taken from the XML schema. You can select this property only if you set the Validate property on the Validation tab to Content or Content and Value.
Use XMLNSC For XMLNS Domain	N	N	Cleared	This property controls whether the XMLNSC compact parser is used for messages in the XMLNS domain. If you set this property, the message data is displayed under XMLNSC in nodes that are connected to the output terminal when the input MQRFH2 header or the Input Message Parsing property Message domain is XMLNS
Retain mixed content	N	N	Cleared	This property controls whether the XMLNSC parser creates elements in the message tree when it encounters mixed text in an input message. If you select the check box, elements are created for mixed text. If you clear the check box, mixed text is ignored and no elements are created.
Retain Comments	N	N	Cleared	This property controls whether the XMLNSC parser creates elements in the message tree when it encounters comments in an input message. If you select the check box, elements are created for comments. If you clear the check

Property	M	C	Default	Description
				box, comments are ignored and no elements are created.
Retain Processing Instructions	N	N	Cleared	This property controls whether the XMLNSC parser creates elements in the message tree when it encounters processing instructions in an input message. If you select the check box, elements are created for processing instructions. If you clear the check box, processing instructions are ignored and no elements are created.
Opaque Elements	N	N	Cleared	This property is used to specify a list of elements in the input message that are to be opaquely parsed by the XMLNSC parser. Opaque parsing is performed only if validation is not enabled (that is, if Validate is None); entries that are specified in Opaque Elements are ignored if validation is enabled.

TibRvInput Node Validation Group:

Set these properties if you want the parser to validate the message body against the meta-data specified on the Input Message Parsing property page.

Validation properties only apply when Message Exchange Type is RVBinaryToBLOB, RVStringToBLOB or RVXMLToBLOB. Parsing and validation occur when the TibrvMsg content contained in the TibRv Message Field is process by the underlying connector classes.



Validate None
Validate Failure Action Exception

Property	M	C	Default	Description
Validate	N	Y	None	This property controls whether validation takes place. Valid values are None, Content, and Content and Value. Note: To validate the parsing of message content set “Parser Timing” to Complete and “Validation” to “Content and Value.”
Validate Failure Action	N	N	Exception	This property controls what happens if validation fails. You can set this property only if you set Validate to Content or Content and Value. Valid values are User Trace, Local Error Log, Exception, and Exception List.

TibRvInput Node Monitoring Group:

Use these properties to add one or more monitoring events to a message flow.

Property	M	C	Default	Description
Events	N	N	None	Events that you have defined for the node are displayed on this tab. By default, no monitoring events are defined on any node in a message flow. Use Add, Edit, and Delete to create, change or delete monitoring events for the node. You can enable and disable events that are shown here by selecting or clearing the Enabled check box. See WMB InfoCenter for details regarding WMB monitoring events.

TibRvInput Node LocalEnvironment Information

The TibRvInput node writes information about the inbound TibrvMsg at the following LocalEnvironment tree location:

- **LocalEnvironment.TIBRV.Input**

LocalEnvironment.TIBRV.Input will contain the following fields:

- **isCM** – "true" if the inbound TibrvMsg is a Certified Message; otherwise, "false".
- **msgByteSize** – byte size of the inbound TibrvMsg.
- **numFields** – number of Fields in the inbound TibrvMsg, this is also the number of Field elements in the RvXML message if RVMsgToXML has been specified.
- **subject** – the Subject of the inbound TibrvMsg.
- **cmSender** -- the name of Sender of the inbound Certified Message. Present only if the inbound TibrvMsg is a Certified Message.
- **msgTimeLimit** – the time limit for the inbound Certified Message in milliseconds. 0 indicates the message time limit is unlimited. Present only if the inbound TibrvMsg is a Certified Message
- **seqNum** – the Sequence Number of the inbound Certified Message. Present only if the inbound TibrvMsg is a Certified Message

NOTE: These fields are of type CHAR.

Using Additional Instances of TibRvInput Node

The TibRvInput node is thread safe and can support additional instances at runtime.

The TibRvInput node does not provide its own additional instance pool; additional instances will be used from the message flow instance pool.

Specifying additional instances for a message flow containing a TibRvInput node does not increase the number of dispatchers processing the underlying Rendezvous event queue; rather, it increases the number of threads that can simultaneously process TibrvMsgs from the broker's internal event queue.

While using additional instances will increase TibRvInput node throughput, use caution when using additional instances with Certified and Distributed Queue delivery as the order the inbound TibrvMsgs are processed is not guaranteed, and messages can be processed and confirmed out of order.

Using the TibRvOutput Node

TibRvOutput Node Description

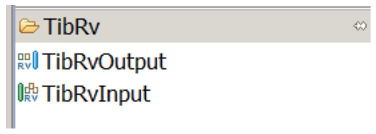
The TibRvOutput node provides connectivity to a TIBCO Rendezvous system allowing the node to publish TibrvMsgs based on a Subject. The Subject and RV Message Type of outbound messages can be dynamically overridden via the LocalEnvironment.

The TibRvOutput node provides two messaging delivery modes:

- Reliable Messaging – Uses a tibrvRvdTransport transport
- Certified Messaging – Uses a tibrvCmTransport transport

TibRv Plug-in category

When supportpac IAMA is installed, the TibRvOutput node will appear in the "TibRv" plug-in category in the Message Brokers Toolkit's Message-Flow editor.



TibRvOutput Node Message Exchange

The TibRvOutput node provides the following message exchange patterns between a message flow and an outbound TibrvMsg :

- **BLOBToRVBinary** – The Message Syntax tree content will be serialized, to a bytestream, using Properties Folder settings and the message content parser. The outbound TibRvMsg contains a single field of type TIBRVMSG_OPAQUE that contains the bytestream. This message exchange pattern could be used to publish a fixed-length (COBOL) message as a single field in a TibrvMsg.
- **BLOBToRVString** -- The Message Syntax tree content will be serialized, to a string, using Properties Folder settings and the message content parser. The outbound TibRvMsg contains a single field of type TIBRVMSG_STRING that contains the string. This message exchange pattern could be used to publish a comma-separated message as a single field in a TibrvMsg. NOTE: The TIBRVMSG_STRING field will use the system default encoding. To change the

encoding set the Execution Group codepage in “property1” of the TIBRV Configurable Service Definition.

- **BLOBToRVXML** – The Message Syntax tree content will be serialized, to a byte array, using Properties Folder settings and the message content parser. The outbound TibrvMsg contains a single field of type TIBRVMSG_XML that contains the byte array. This message exchange pattern could be used to publish an XML message as a single field in a TibrvMsg. The underlying TibrvXml datatype provides data optimization for XML content. NOTE: The XML character content will be encoded using the system default. To change the encoding set the Execution Group codepage in “property1” of the TIBRV Configurable Service Definition.
- **XMLToRVMessage** – The Message Syntax tree contains valid RvXML syntax. The rv.xsd XML schema defines the RvXML syntax for each of the supported TIBRVMSG data types and provides a means of representing a TibrvMsg in a message flow. The TibRvOutput node traverses the simple RvXML syntax tree generating a matching TivrvMsg in the process. The outbound TibrvMsg contains a field for each Field contained in the RvXML syntax tree.

TibRvInput Plug-in node terminals

Terminal	Description
Failure	The output terminal that the message assembly is routed to if a failure occurs in the node. The ExceptionList tree will contain a description of the exception that caused the failure.
Out	The output terminal that emits the inbound message assembly. The LocalEnvironment tree is updated with information about the published message.
In	The input terminal that receives the inbound message assembly. The LocalEnvironment tree can be used to override the Subject and RVMessageType properties.

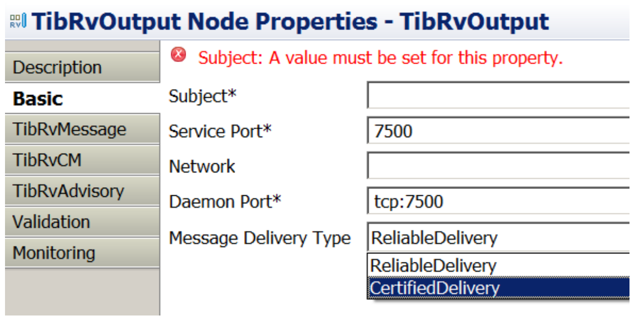
TibRvOutput node properties

The TibRvOutput node properties are displayed on the following Property Pages:

- **Basic** – Transport, Subject and Messaging Mode
- **TibRvMessage** -- Message Exchange Pattern between TibrvMsg and message flow
- **TibRvCM** – Certified Messaging
- **TibRvAdvisory** – Rendezvous Advisory Message Logging
- **Validation** – Validate and Failure Action
- **Monitoring** – Configure WMB Monitoring Events

TibRvOutput Node Basic Group

The Basic property page defines the Rendezvous connectivity parameters, subject and message delivery type.


TibRvOutput Node Properties - TibRvOutput

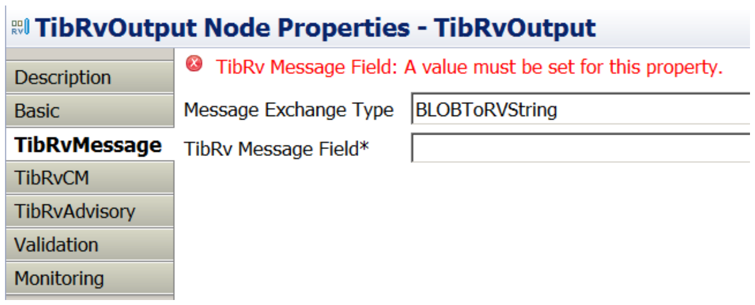
Subject: A value must be set for this property.

Subject*
Service Port* 7500
Network
Daemon Port* tcp:7500
Message Delivery Type ReliableDelivery

Property	M	C	Default	Description
Subject	Y	Y	Cleared	Specifies the Subject for the outbound TibrvMsg. The Subject can be overridden via the LocalEnvironment: OutputLocalEnvironment.Destination.TIBRV.Output.subject See TIBCO Rendezvous Concepts manual for full description of Subject syntax.
Service Port	Y	Y	7500	This is the port number that will be used to listen for Rendezvous messages. See TIBCO Rendezvous Concepts manual for full description of the Service Port property.
Network	N	N	Cleared	The network interface that will be used for outbound messages published by this node. If no value is specified then the primary network interface for the host will be used. See TIBCO Rendezvous Concepts manual for full description of the Network property.
Daemon Port	Y	Y	tcp:7500	This is the port that the node's transport will use to communicate with the RV daemon. See TIBCO Rendezvous Concepts manual for full description of the Daemon Port property.
Message Delivery Type	Y	N	Reliable Delivery	Specifies the messaging delivery mode to be used by the node. Reliable – tibrvRvdTransport is used Certified – tibrvCmTransport is used

TibRvOutput Node TibRvMessage Group:

The TibRvMessage property page defines the message exchange pattern that will be used between the message syntax tree and the outbound TibrvMsg generated by the TibRvOutput node.


TibRvOutput Node Properties - TibRvOutput

TibRv Message Field: A value must be set for this property.

Message Exchange Type: BLOBToRVString

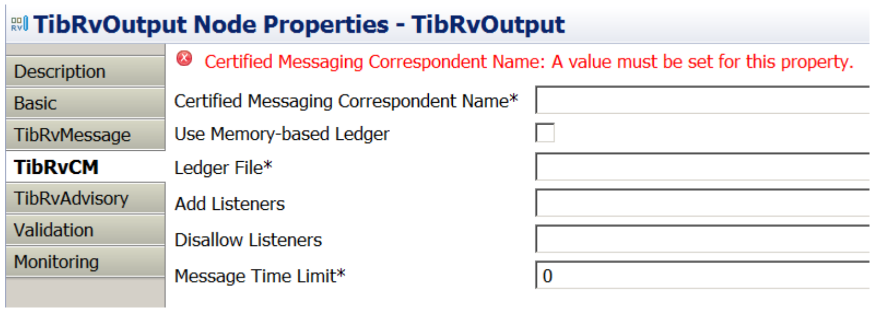
TibRv Message Field*:

Property	M	C	Default	Description
Message Exchange Type	Y	Y	BLOBToRVString	Specifies how the message syntax tree will be exchanged with outbound TibrvMsg (BLOBToRVString, BLOBToRVBinary,BLOBToRVXML, XMLToRVMessage) BLOBToRVString, BLOBToR VBinary and BLOBToRVXML must specify a name for the TibrvMsg Field property to contain the serialized content. This property can be overridden via the LocalEnvironment: OutputLocalEnvironment.Destination.TIBRV. Output.rvMessageType NOTE: RVString and RVXML fields will use the system default codepage. To change the encoding set the Execution Group codepage in “property1” of the TIBRV Configurable Service Definition.
TibRv Message Field	Y	N		Specifies the name of the Field in the outbound TibrvMsg that will contain the serialized content.

				Must be specified if RvMessageType is BLOBToRVString, BLOBToRVBinary or BLOBToRVXML. This property is disabled when RvMessageType is XMLToRVMessage.

TibRvOutput Node TibRvCM Group:

The TibRvCM property page is only active when the Message Delivery Type is Certified Delivery.


TibRvOutput Node Properties - TibRvOutput

Certified Messaging Correspondent Name: A value must be set for this property.

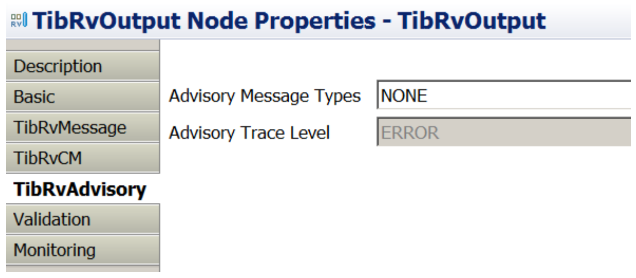
Certified Messaging Correspondent Name*
Use Memory-based Ledger
Ledger File*
Add Listeners
Disallow Listeners
Message Time Limit* 0

Property	M	C	Default	Description
Certified Messaging Correspondent Name	Y	Y		The CM correspondent name must be specified and must be non-null and cannot be an empty string (all spaces). The CM correspondent name must be unique as the CM name is bound to the tibrvCmTransport that is used by the TibRvOutput node. See TIBCO Rendezvous Concepts manual for full description of the Certified Messaging Correspondent Name property.
Use Memory-based Ledger	Y	N	Unchecked	When checked, the node will use a memory-based ledger and not a file-based ledger. The Ledger File property is disabled when this property is checked.
Ledger File	Y	Y	Cleared	Specifies the ledger file to be used by the TibRvOutput node. The ledger file must exist on a local file system. The broker must

				have permission to create/read/write files in the named directory. If the ledger file already exists for the same CM name, it will be reused. If the ledger file does not exist it will be created. This property is disabled when Use Memory-based Ledger is checked.
Add Listeners	N	Y	Cleared	This property is a semi-colon-separated list of CM Correspondent Name|Subject pairs (CM1|Subject1;CM2|Subject2;CM1|Subject3) These listeners will be preregistered and certified messages will be saved in the ledger even if the listener is not active.
Disallow Listeners	N	N	Cleared	This property is a comma-separated list of CM Correspondent names (CM1,CM2,CM3). These listeners will be disallowed before the message flow is started and will not be able to register as a CM listener with this TibRvOutput node.
Message Time Limit	Y	N	0 (unlimited)	Specifies, in seconds, the interval Rendezvous will wait for a confirmation before making the TibrvMsg as expired.

TibRvOutput Node TibRvAdvisory Group:

The TibRvAdvisory property page allows the TibRvOutput node to log Rendezvous Advisory messages associated with the underlying Rendezvous transport. Advisory messages are written to the TIBRV Activity Log under the Execution Group the node is deployed to.



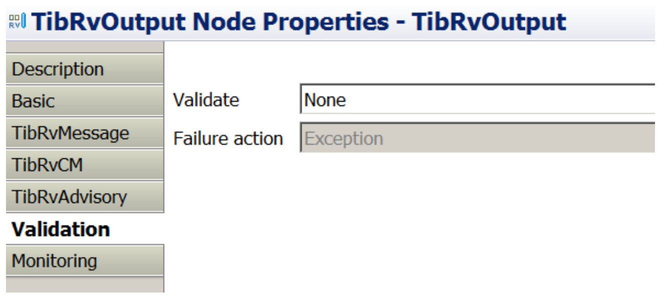
Advisory Message Types NONE
Advisory Trace Level ERROR

Property	M	C	Default	Description
Advisory Message Types	Y	N	NONE	• NONE – Advisory Messages are not logged. • SYSTEM – SYSTEM Advisory Messages are logged. • CM – Certified Messaging Advisory Messages are logged. • ALL – SYSTEM and CM Advisory Messages are logged.
Advisory Trace Level	N	N	ERROR	• ERROR – Only ERROR Advisory Messages are logged. • WARN – Only WARN Advisory Messages are logged. • INFO – Only INFO Advisory Messages are logged. • ALL – ERROR/WARN/INFO Advisory messages are logged. This property is disabled when Advisory Message Types is NONE.

TibRvOutput Node Validation Group:

Set these properties if you want the parser to validate the message body against the meta-data specified by the Properties folder.

Validation properties only apply when Message Exchange Type is BLOBToRVBinary, BLOBToRVString or BLOBToRVXML. Parsing and validation occur when the underlying connector classes serialize the message syntax tree.

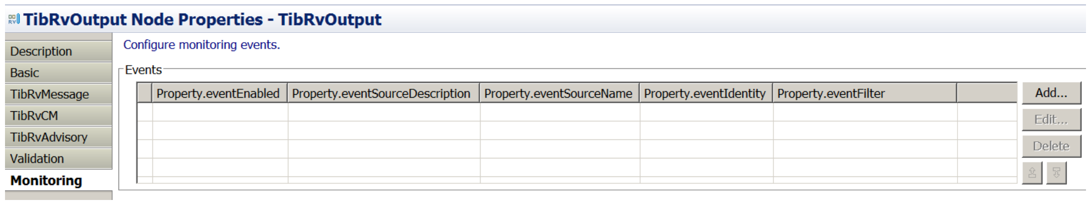

TibRvOutput Node Properties - TibRvOutput

Validate: None

Failure action: Exception

Property	M	C	Default	Description
Validate	N	Y	None	This property controls whether validation takes place. Valid values are None, Content, and Content and Value.
Validate Failure Action	N	N	Exception	This property controls what happens if validation fails. You can set this property only if you set Validate to Content or Content and Value. Valid values are User Trace, Local Error Log, Exception, and Exception List.

TibRvOutput Node Monitoring Group:


TibRvOutput Node Properties - TibRvOutput

Configure monitoring events.

Events

Property.eventEnabled	Property.eventSourceDescription	Property.eventSourceName	Property.eventIdentity	Property.eventFilter

Property	M	C	Default	Description
Event	N	N	None	Events that you have defined for the node are displayed on this tab. By default, no monitoring events are defined on any node in a message flow. Use Add, Edit, and Delete to create, change or delete monitoring events for the node. You can enable and disable events that are shown here by selecting or clearing the Enabled check box. See WMB InfoCenter for WMB monitoring event details.

TibRvOutput Node LocalEnvironment Override Properties

The following TibRvOutput node properties can be overridden in the LocalEnvironment syntax tree:

- OutputLocalEnvironment.Destination.TIBRV.Output.subject – must specify a legal Rendezvous Subject

- OutputLocalEnvironment.Destination.TIBRV.Output.rvMessageType – must be one of (BLOBToRVString, BLOBToRVBinary, BLOBToRVXML or XMLToRVMessage)

TibRvOutput Node WrittenDestination Information

The TibRvOutput node provides the following information in the LocalEnvironment syntax tree after it has successfully published the TibrvMsg

- LocalEnvironment.WrittenDestination.TIBRV.Output.byteSize – Byte size of the TibrvMsg
- LocalEnvironment.Destination.TIBRV.Output.numFields – Number of fields in the TibrvMsg
- LocalEnvironment.Destination.TIBRV.Output.subject – The send Subject of the TibrvMsg

Using Additional Instances of the TibRvOutput Node

The TibRvOutput node is thread safe and can support additional instances at runtime.

The TibRvOutput node does not provide its own additional instance pool; additional instances will be used from the message flow instance pool.

All instances of a TibRvOutput node share the same underlying Tibrv transports.

While using additional instances will increase TibRvOutput node throughput, use caution when using additional instances with Certified delivery as the order the outbound TibrvMsgs are published is not guaranteed, and messages can be published out of sequence.

TibRvInput/TibRvOutput Node Administration

The status of deployed/running TibRv Nodes can be reported by using the mqsireportproperties command. In addition, a certified messaging correspondent's ledger can be reviewed and modified using the CMP API to execute administration commands on a deployed/running TibRvOutput node. The CMP API can also be used to modify a TibRv node's Advisory Message Logging configuration.

TibRv administration commands are executed via the CMP API. The ExecIAMAAdminCommands.jar file, provided with the IAMA supportpac installation, provides command-line access to all TibRv node administration commands.

Using mqsireportproperties with TibRv Nodes

Both the TibRvInput and TibRvOutput nodes report node information and status via the mqsireportproperties command:

```
mqsireportproperties Broker –e EG –o TibRvObject –r
```

Where:

TibRvObject is the object path for the TibRv nodes being queried:

- "TIBRV" – All TibRv Input and Output nodes in the Execution Group
- "TIBRV/Input" – All TibRv Input nodes in the Execution Group
- "TIBRV/Output" – All TibRv Output nodes in the Execution Group
- "TIBRV/Input/TibRvNode" – A specific TibRvInput node in the Execution Group
- "TIBRV/Output/TibRvNode – A specific TibRvOutput node in the Execution Group
- NOTE: TibRvNode syntax is -- BrokerSchema.MessageFlowName_NodeName. If the "default" Broker schema is being used, then the syntax is – MessageFlowName_NodeName

The following mqsireportpropeties command will display the status of all TibRvInput and TibRvOutput nodes deployed to an Execution Group on the broker:

- mqsireportproperties BKName –e EG –o TIBRV –r

The following mqsireportpropeties command will display all TibRvInput nodes deployed to the EG Execution Group on the broker:

- mqsireportproperties BKName –e EG –o TIBRV/Input –r

The following mqsireportpropeties command will display all TibRvOutput nodes deployed to the EG Execution Group on the broker:

- mqsireportproperties BKName –e EG –o TIBRV/Output –r

The following mqsireportpropeties command will display information about a specific TibRvOutput node (TibRvOutputNodeName in message flow MsgFlowName) deployed to the EG Execution Group on the broker:

- mqsireportproperties BKName –e EG –o TIBRV/Output/MsgFlowName_TibRvOutputNodeName –r

NOTE: the path for a specific TibRvInput or TibRvOutput node is a concatenation of BrokerSchema.MessageFlowName_TibRvNodeName. If the "default" broker schema is used then the path is a concatenation of MessageFlowName_TibRvNodeName.

TibRvInput mqsireportproperties Information

The mqsireportproperties command displays the following information for a TibRvInput node:

- Trasnport
  - daemon – The node's "daemon" property
  - network – The node's "network" property
  - service – The node's "service" property
- Queue
  - name – The node's internal event queue
  - count – Number of events in the queue
  - discardcount – Number of discarded events
  - limitpolicy – The event limit
  - priority – The queue's priority
  - valid – True is queue is valid
- Listener
  - type – The node's "Message Delivery Type" property (reliable,certified,distributedQueue)
  - subject – The node's "subject" property
- Messages
  - messagesReceived – Total number of messages (reliable and certified) received by node since startup
  - cmMessagesReceived – Total number of certified messages received by the node since startup
  - cmMessagesConfirmedOnSuccess – Total number of certified messages confirmed on success since startup
  - cmMessagesConfirmedOnFailure – Total number of certified messages confirmed on failure since startup
  - cmMessagesAutoConfirmed – Total number of certified messages that were auto confirmed since startup

- CertifiedMessageDelivery
  - cmName – The node’s “Certified Messaging Correspondent Name” property
  - ledgerFile – The node’s “Ledger File” property (“Memory-Based Ledger” or path to file-based ledger file)
  - requestOld – The node’s “Request Old” property (true or false)
  - confirm – The node’s “Certified Messaging Confirmation” property (“Auto”, “OnSuccess”, “OnSuccessAndFailure”)
- Advisory
  - type – The node’s “Advisory Message Types” property (NONE,SYSTEM,CM,ALL)
  - level – The node’s “Advisory Trace Level” property (ERROR,INFO,WARN,ALL)

TibRvOutput mqsireportproperties Information

The mqsireportproperties command displays the following information for a TibRvInput node:

- Trasnport
  - daemon – The node’s “daemon” property
  - network – The node’s “network” property
  - service – The node’s “service” property
- Messages
  - Subject – The node’s “Subject” property
  - messagesSent – Total number of messages sent by node since startup
- CertifiedMessageDelivery
  - cmName – The node’s “Certified Messaging Correspondent Name” property
  - ledgerFile – The node’s “Ledger File” property (“Memory-Based Ledger” or path to file-based ledger file)
  - messageTimeLimit – The node’s “Message Time Limit” property
  - listeners – The node’s “Allow Listeners” property
  - disallow – The node’s “Disallow Listeners” property
- Advisory
  - type – The node’s “Advisory Message Types” property (NONE,SYSTEM,CM,ALL)
  - level – The node’s “Advisory Trace Level” property (ERROR,INFO,WARN,ALL)

TibRv Node Administration Commands

TibRv node administration commands are executed by using the ExecIAMAAdminCommands utility that is part of the IAMA SupportPac. Use the following command-line format to execute admin commands using ExecIAMAAdminCommands.jar:

- java –jar ExecIAMAAdminCommands.jar –i hostname –q QMgrName – p Port# -e ExecutionGroup –o NodePath –f Function [Function Parameters]

  Where:
    - i = hostname of Broker system
    - p = Broker's Queue Manager's listener port number
    - q = Broker's Queue Manager's name
    - e – Execution Group the TibRv node is deployed to
    - o – Path for TibRv node object.
    - f – The TibRv command to execute
    - [Function Parameters] – the additional parameters required by the specified function

- NOTE: NodePath is "Output/TibRvNodeName" for TibRvOutput nodes and "Input/TibRvNodeName" for TibRvInput nodes. TibRvNodeName syntax is BrokerSchema.MessageFlow_NodeName. If "default" BrokerSchema is used then no BrokerSchema prefix is used.

Output from the administration commands is written to the TIBRV Activity Log of the Execution Group the TibRv node is deployed to. The WMB Explorer (MBX) can be used to view and filter the TIBRV Activity Log. All commands write a message to the TIBRV Activity Log indicating if they were successful or if they failed.

The reviewLedger command will write numerous messages to the TIBRV Activity Log. The first reviewLedger message will indicate that the reviewLedger command has started. There will be an additional message written to the TIBRV Activity Log for each "subject" that is part of the reviewLedger query. The final reviewLedger message indicates that the reviewLedger command has completed.

The TibRv node "advisory" administration command can be used to modify the "Advisory Message Logging" state of a TibRvInput or TibRvOutput node:

- **advisory**
  java –jar ExecIAMAAdminCommands.jar –i hostname –q Qmgr –p Port# -e EG -f advisory –t AdvisoryType –l AdvisoryLevel

  f – Function, execute the "advisory" command
  t – AdvisoryType, one of NONE, SYSTEM, CM or ALL. Note: NONE turns Advisory Logging off.
  l – AdvisoryLevel, one of INFO, WARN, ERROR or ALL

The TibRvOutput node, when using certified messaging, provides the following administration commands to review and modify the ledger file associated with the TibRvOutput node:

- **reviewLedger** – Executes the reviewLedger command. Output from this command is written to the TIBRV resource manager's Activity Log and can be view and filtered using MBX. A Subject parameter must be provided and can contain wildcards.

  ```
  java –jar ExecIAMAAdminCommands.jar –i hostname –q Qmgr –p Port# -e EG
  -f reviewLedger –s Subject
  ```

  f – Function, execute the reviewLedger command
  s – Subject, the ledger will be queried with this subject (wildcards are permitted)

- **removeListener** – Executes the removeListener command on the persistent CM correspondent's ledger file. A CM listener name and Subject must be provided.

  ```
  java –jar ExecIAMAAdminCommands.jar –i hostname –q Qmgr –p Port# -e EG
  -f removeListener –c CMName –s Subject
  ```

  f – Function, execute the removeListener command
  c – Certified Messaging Correspondent's Name, cancel certified delivery of the specified subject to this correspondent's listeners
  s – Subject, certified delivery of the subject will be canceled to the specified listener (wildcards are not permitted)

- **disallowListener** – Executes the disallowListener command on the persistent CM correspondent's ledger file. A CM listener name must be provided.

  ```
  java –jar ExecIAMAAdminCommands.jar –i hostname –q Qmgr –p Port# -e EG
  -f dissallowListener –c CMName
  ```

  f – Function, execute the dissallowListener command
  c – Certified Messaging Correspondent's Name, certified delivery to the correspondent's listeners will be canceled.

- **allowListener** – Executes the allowListener command on the persistent CM correspondent's ledger file. A CM listener name must be provided.

  ```
  java –jar ExecIAMAAdminCommands.jar –i hostname –q Qmgr –p Port# -e EG
  -f allowListener –c CMName
  ```

  f – Function, execute the allowListener command
  c – Certified Messaging Correspondent's Name, requests for certified delivery from the correspondent's listeners will be accepted

- **expireMessages** – Executes the expireMessages command on the persistent CM correspondent's ledger file. A Subject and SequenceNumber must be provided.

  java –jar ExecIAMAAdminCommands.jar –i hostname –q Qmgr –p Port# -e EG -f expireMessages –s Subject –n SequenceNumber

  f – expireMessages, execute the expireMessages command
  s – Subject, mark message with this subject
  n – Sequence Number, mark messages with sequence number less than or equal this value

- **removeSendState** – Executes the removeSendState command on the persistent CM correspondent's ledger file. A Subject must be specified.

  java –jar ExecIAMAAdminCommands.jar –i hostname –q Qmgr –p Port# -e EG -f removeSendState –s Subject

  f – removeSendState, execute the removeSendState command
  s – Subject, remove the send state for this subject

- NOTE: These admin commands must be executed against a deployed and running TibRvOutput node

TibrvMsg Message Exchange

With WMB we typically think of messages as fixed-length (COBOL,C), tag-delimited, comma-separated, XML or other type of structure. With Rendezvous there is only the TibrvMsg structure – a simple structure that contains a sequence of fields, where each field has a “name”, “type” and “value.”

The TibRv nodes integrate the TIBCO Rendezvous messaging system with a WMB message flow. Messages in a WMB message flow context are contained in a Message Syntax Tree, whereas, messages in Rendezvous are contained in the TibrvMsg structure/object.

Both the TibRvInput and TibRvOutput nodes exchange message content with the TibrvMsg structure. The TibRv nodes integrate the message flow context (or message syntax tree) with the TibrvMsg structure. The TibRvOutput node integrates the input message syntax tree with an outbound TibrvMsg message, and the TibRvInput node integrates an inbound TibrvMsg message with an output message syntax tree. Both nodes support four message exchange modes that allow any type of message structure to be passed between WMB and Rendezvous messaging systems.

The TibRv Nodes support two types of message exchange:

- **Message Stream** – Message content is a stream of data that is contained within a single field of the TibrvMsg. Any WMB parser can be used to serialize the Message Stream from the Message Syntax tree (outbound), or to generate a Message Syntax Tree from a Message Stream (inbound). This message exchange pattern provides the message flow direct access to a stream of data in the TibrvMsg structure. The TibRvInput node’s “Input Message Paring” property page allows any parser/meta-data to be used to parse the message stream and generate the message syntax tree. For the TibRvOutput node, the underlying connector classes use the message syntax tree parser and meta-data defined in the Properties folder to generate the message stream.
- **RvXML** – With the RvXML message exchange pattern a TibrvMsg structure is translated to RvXML syntax on input and on output a RvXML syntax tree is translated to a TibrvMsg structure. The rv.xsd schema defines the RvXML syntax that can describe any TibrvMsg structure. RvXML is a simple XML syntax that supports the TIBRVMSG data types defined in Rendezvous 8.0 or higher. Complex types such as TIBRVMSG Arrays and nested TIBRVMSG_MSG types are also supported. This message exchange pattern provides the message flow direct access to all fields in the TibrvMsg structure.

The following output message exchange patterns are support by the TibRvOutput node:

- BLOBToRVString – Message Syntax Tree ➔ Single TIBRVMSG_STRING field in the TibrvMsg. The name of the field that contains the message string is a property of the node.

- BLOBToRVBinary – Message Syntax Tree ➔ Single TIBRVMSG_OPAQUE field in the TibrvMsg. The name of the field that contains the message byte array is a property of the node.
- BLOBToRVXML – XML Message Syntax Tree ➔ Single TIBRVMSG_XML field in the TibrvMsg. The name of the field that contains the XML message byte array is a property of the node.
- XMLToRVMessage – RvXML Message Syntax Tree ➔ TibrvMsg. Translates the RvXML Message Syntax Tree to a TibrvMsg based on the rv.xsd schema. The XMLNSC parser is not used to serialize the RvXML Message Syntax Tree; rather, the TibRvOutput node traverses the RvXML Syntax Tree generating the associated TibrvMsg structure. Validation of the RvXML Message Syntax Tree occurs during the translation and the node will throw an exception if the RvXML Syntax Tree is not valid.
- Output Note: For BLOBToRVString/BLOBToRVBinary/BLOBToRVXML, the message stream that is generated is dependant on the parser being used and Properties folder settings, just like other WMB output nodes. Validation is inherited for this process.

  While the XMLNSC parser must be used to create the RvXML Message Syntax Tree, it is not used to serialize the syntax tree and generate the message byte stream. The TibRvOutput node is highly optimized to process/validate the RvXML Message Syntax Tree and provides excellent performance.

The following message exchange patterns are supported by the TibRvInput node:

- RVStringToBLOB – Single TIBRVMSG_STRING field in TibrvMsg ➔ Output Message Syntax Tree. The name of the field that contains the message string is a property of the node.
- RVBinaryToBLOB – Single TIBRVMSG_BINARY field in TibrvMsg ➔ Output Message Syntax Tree. The name of the field that contains the message byte array is a property of the node.
- RVXMLToBLOB – Single TIBRVMSG_XML field in TibrvMsg ➔ Output Message Syntax Tree. The name of the field that contains the XML message byte array is a property of the node.
- RVMessageToXML – TibrvMsg ➔ XMLNSC Message Syntax Tree. Translates a TibrvMsg structure to a XMLNSC Message Syntax Tree based on rv.xsd schema. No XML content is created for the XMLNSC parser to consume; rather, the TibRvInput node sequentially processes the TibrvMsg structure generating the output XMLNSC Message Syntax Tree.
- Input Note: For RVStringToBLOB/RVBinaryToBLOB/RVXMLToBLOB, the output Message Syntax Tree generated by the TibRvInput node is dependant on the meta-data properties on the “Input Message Parsing” property page. The message byte stream that is extracted from the TibrvMsg will be parsed, and possibly validated, using the meta-data (parser) defined on the node. If no meta-data is defined for the node the BLOB parser will be used.

For RVMessageToXML no XML content is created for the XMLNSC parser to consume; rather, the TibRvInput node constructs the RvXML message syntax tree as it makes a sequential pass of the TibrvMsg. The TibRvInput node is highly optimized to construct the RvXML message syntax tree and offers excellent performance.

Exchanging Message Streams

WMB provides numerous parsers that consume a stream of bytes and produce a message syntax tree (input); likewise, the parsers can consume a message syntax tree and produce a stream of bytes (output).

Working with messages as streams allows complex messages, which may contain hundreds or thousands of fields, to be contained within a single field in the TibrvMsg structure. Applications working with the TibrvMsg must know how to extract and parse the byte stream, just like WMB, to access fields within the message.

The BLOBToRVString/RVStringToBLOB, BLOBToRVBinary/RVBinaryToBLOB and BLOBToRVXML/RVXMLToBLOB message exchange types allow the TibRv nodes to exchange message byte streams with Rendezvous applications. The message stream is contained within a single field in the TibrvMsg. The name (and type) of the single field is a mandatory property of the TibRvInput/TibRvOutput node.

The serialization of the Message Syntax Tree is performed by the underlying connector classes and follows the “normal” Message Syntax Tree serialization process (just like an MQOutput node). Any exceptions that occur during the serialization process are handled by the underlying connector classes and control is routed to the Failure terminal.

BLOBToRVBinary/RVBinaryToBLOB message exchange types treat the message as a byte array, and would typically be used for messages that contain binary content (COBOL Copybook or C structure). With these modes the message byte array is contained in a field of type TIBRVMSG_OPAQUE within the TibrvMsg.

BLOBToRVString/RVStringToBLOB message exchange types treat the message as a string, and would typically be used for messages that contain only character-based data or where codepage translation of string data is desired (HL7,EDI). With these modes the message stream (a string) is contained in a field of type TIBRVMSG_STRING within the TibrvMsg.

BLOBToRVXML/RVXMLToBLOB message exchange types treat the message as a byte array via the TibrvXml object. The TibrvXml object provides optimizations for transporting XML documents in Rendezvous. These message exchange modes would be used to optimize the integration of XML messages with Rendezvous applications. With these modes the message stream (a byte array) is contained in a field of type

TIBRVMSG_XML within the TibrvMsg. The BLOBToRVXML/RVXMLToBLOB message exchange types should only be used for XML messages.

The name of the field, in the TibrvMsg, that contains the message stream is a property of the TibRvInput or TibRvOutput node. If the named field does not exist in the inbound TibrvMsg, or is not the correct TIBRVMSG type, then the TibRvInput node will throw an exception, with the TibrvMsg bytes contained under the BLOB parser; the exception is described under the ExceptionList.

With BLOBToRVString and BLOBToRVXML, the single TIBRVMSG_STRING/TIBRVMSG_XML field, containing the message content, will be encoded using the system default codepage. This codepage can be overridden at the Execution Group level in the TIBRV Configurable Service – property1 is a comma-separated list of "ExcutionGroup=Codepage" pairs (EG1=1250,EG2=1252). This codepage setting is execution-group-wide and affects TibRvInput nodes as well.

With RVStringToBLOB and RVXMLToBLOB the bytes for the single TIBRVMSG_STRING/TIBRVMSG_XML field are encoded using the system default The "default" codepage is overridden by "property1" in the TIBRV Configurable Service.

Because the Rendezvous string encoding setting is process-wide, all TibRvInput and TibRvOutput nodes within an Execution Group must use the same string encoding for fields of type TIBRVMSG_STRING and TIBRVMSG_XML.

If a TibRvInput node uses RVXMLToBLOB, RVStringToBLOB or RVBinaryToBLOB message exchange pattern and "Input Message Parsing" properties are set to parse the TibrvMsg content, the Broker's default CCSID and Encoding will be used. If the character content of the TibrvMsg content uses a different CCSID, then that CCSID must be specified for the "Coded Character Set ID" property on the "TibRvMessage" property page. If the binary content of the TibrRvMsg uses a different encoding, then that encoding must be specified for the Encoding property on the "TibRvMessage" property page. . NOTE: These properties are not implemented in release 1.0.0. A work-around is to specify the CCSID and/or Encoding in the Parse clause of a Create statement.

Exchanging RvXML Messages

RvXML is a simple XML syntax that uses the XMLNSC parser to describe any TibrvMsg structure and provides a powerful, easy-to-use means for a WMB message flow to have direct access to a TibrvMsg structure. RvXML is based on the rv.xsd schema that is provided with the IAMA supportpac.

A TibRvOutput node, using the XMLToRVMessage message exchange type, will traverse an RvXML message syntax tree generating the outbound TibrvMsg in the process. The RvXML syntax is validated during this process.

A TibRvInput node, using the RVMessageToXML message exchange type, will sequentially process a TibrvMsg generating the output RvXML message syntax tree in the process.

Any of the WMB transformation techniques can be used to produce RvXML that will be input to a TibRvOutput node – ESQL, JAVA, .net, PHP, GUI. Likewise, any of these WMB transformation techniques can use an RvXML message syntax tree, generated by a TibRvInput, as a message source.

NOTE: TibrvMsg.String and TibrvMsg.XML fields are encoded/decoded using the system default codepage. To use a different codepage set the Execution Group codepage in “property1” of the TIBRV Configurable Service Definition.

TIBRVMSG Data Types

The rv.xsd schema defines the TIBRVMSG data types that are supported by the RvXML syntax for integration with the TibrvMsg structure.

The following TIBRVMSG Data Types are supported by the RvXML syntax:

TIBRVMSG Type	TibrvMsg Type	ESQL Type	JAVA Type	Comment
TIBRVMSG_BOOL	TibrvMsg.BOOL	BOOLEAN	Bool	
TIBRVMSG_I8	TibrvMsg.I8	INTEGER	Long	
TIBRVMSG_I16	TibrvMsg.I16	INTEGER	Long	
TIBRVMSG_I13	TibrvMsg.I32	INTEGER	Long	
TIBRVMSG_I64	TibrvMsg.I64	INTEGER	Long	
TIBRVMSG_U8	TibrvMsg.U8	INTEGER	Long	
TIBRVMSG_U16	TibrvMsg.U16	INTEGER	Long	
TIBRVMSG_U32	TibrvMsg.U32	INTEGER	Long	
TIBRVMSG_U64	TibrvMsg.U64	INTEGER	Long	
TIBRVMSG_F32	TibrvMsg.F32	FLOAT	Double	
TIBRVMSG_F64	TibrvMsg.F64	FLOAT	Double	
TIBRVMSG_IPADDR32	TibrvMsg.IPADDR32	CHAR	String	
TIBRVMSG_IPPORT16	TibrvMsg.IPPORT16			
TIBRVMSG_I8ARRAY	TibrvMsg.I8ARRAY	Elements of type INTEGER	Elements of type LONG	
TIBRVMSG_I16ARRAY	TibrvMsg.I16ARRAY	Elements of type INTEGER	Elements of type LONG	
TIBRVMSG_I32ARRAY	TibrvMsg.I32ARRAY	Elements of type INTEGER	Elements of type LONG	
TIBRVMSG_I64ARRAY	TibrvMsg.I64ARRAY	Elements of type INTEGER	Elements of type LONG	
TIBRVMSG_U8ARRAY	TibrvMsg.U8ARRAY	Elements of type INTEGER	Elements of type LONG	Not supported for input

				by TibRvInput node
TIBRVMSG_U16ARRAY	TibrvMsg.U16ARRAY	Elements of type INTEGER	Elements of type LONG	Not supported for input by TibRvInput node
TIBRVMSG_U32ARRAY	TibrvMsg.U32ARRAY	Elements of type INTEGER	Elements of type LONG	Not supported for input by TibRvInput node
TIBRVMSG_U64ARRAY	TibrvMsg.U64ARRAY	Elements of type INTEGER	Elements of type INTEGER	Not supported for input by TibRvInput node
TIBRVMSG_F32ARRAY	TibrvMsg.F32ARRAY	Elements of type FLOAT	Elements of type DOUBLE	
TIBRVMSG_F64ARRAY	TibrvMsg.F64ARRAY	Elements of type FLOAT	Elements of type DOUBLE	
TIBRVMSG_OPAQUE	TibrvMsg.OPAQUE	BLOB	Byte[]	
TIBRVMSG_DATETIME	TibrvMsg.DATETIME	TIMESTAMP	com.ibm.broker. plugin.MbTimestamp	
TIBRVMSG_MSG	TibrvMsg.MSG			
TIBRVMSG_XML	TibrvMsg.XML	CHAR	String	
TIBRVMSG_STRING	TibrvMsg.STRING	CHAR	String	
TIBRVMSG_MSGARRAY	TibrvMsg.MSGARRAY			
TIBRVMSG_STRINGARRAY	TibrvMsg.STRINGARRAY	Elements of type CHAR	Elements of type String	

RvXML Syntax

The RvXML syntax is described by the rv.xsd schema that is provided with the IAMA supportpac.

The RvXML syntax must use the XMLNSC parser. Failure to build the RvXML Message Syntax tree using the XMLNSC parser will cause exceptions with the TibRvOutput node. The TibRvInput node will always use the XMLNSC parser when the "Message Exchange Type" is RVMessageToXML.

The root tag of an RvXML XML message is always <Message>. <Message> may contain 0 to unbounded <Field> children. Each <Field> element requires a "name" and "type" attribute. The "name" attribute can be any valid TibrvMsg Field name. The "type" attribute must be one of the supported TIBRVMSG types.

For ARRAY types, the <Field> element contains a single child element named <Array> that contains 0 to unbounded <Element> elements. The <Element> elements are of the base type of the ARRAY (example: for TIBRVMSG_STRINGARRAY, each <Element> element is of type TIBRVMSG_STRING).

A TIBRVMSG_MSG <Field> contains a single child tag <Message> that contains 0 to unbounded <Field> children. A TIBRVMSG_MSG <Field> may be empty, that is, it contains no <Field> elements.

See the IAMA Sample message flows for examples of working with RvXML.

RvXML Message Examples

Following are XML examples of RvXML syntax supported by the TibRv Nodes. In these examples, a TibRvOutput node could consume a message syntax tree (built with the XMLNSC parser) that represents the XML message and generate an associated TibrvMsg structure. Likewise, a TibRvInput node could build an RvXML message syntax tree (using the XMLNSC parser) that represents the associated TibrvMsg structure.

Following is a TibrvMsg containing 5 Fields

```
<Message>
  <Field name=”CRec”  type=”TIBRVMSG_STRING” >C10</Field>
  <Field name=”CName” type=”TIBRVMSG_STRING”>Tom Tinz</Field>
  <Field name=”CAge”  type=”TIBRVMSG_I8”>38</Field>
  <Field name= “CId” type=”TIBRVMSG_I32”>1376549</Field>
  <Field name=”CDate”  type=”TIBRVMSG_DATETIME”>2012-06-21T10:09:08</Field>
</Message>
```

Following is a TibrvMsg containing 3 Fields. Two of the Fields are ARRAY data types.

```
<Message>
  <Field name=”CRec”  type=”TIBRVMSG_STRING” >C11</Field>
  <Field name=”CIdArray” type=”TIBRVMSG_I32ARRAY”>
    <Array>
      <Element>12344987</Element>
      <Element>12399874</Element>
      <Element>12388754</Element>
      <Element>12357774</Element>
    </Array>
  </Field>
  <Field name=”CNameArray” type=”TIBRVMSG_STRINGARRAY”>
    <Array>
      <Element>Tom Tinz</Element>
      <Element>Jill Kurtz</Element>
      <Element>Roy Dole</Element>
      <Element>Gill Hoyle</Element>
    </Array>
  </Field>
</Message>
```

Following is a TibrvMsg containing a single Field of type TIBRVMSG_MSG. The nested message contains 3 Fields, where the last Field is another nested TIBRVMSG_MSG. The third nested message contains a single Field of type TIBRVMSG_STRINGARRAY:

```
<Message>
  <Field name=”Envelope” type=”TIBRVMSG_MSG”>
    <Message>
      <Field name=”EnvId” type=”TIBRVMSG_STRING”>UI7645T8</Field>
      <Field name=”EnvDate” type=”TIBRVMSG_DATETIME”>2012-09-15T01:10.33</Field>
      <Field name=”SubEnv” type=”TIBRVMSG_MSG”>
        <Message>
          <Field name=”SubIdArray” type=”TIBRVMSG_STRINGARRAY”>
            <Array>
              <Element>UI87TH9</Element>
              <Element>UI90TJ7</Element>
              <Element>UI99TT9</Element>
            </Array>
           </Field>
         </Message>
        </Field>
      </Message>
    </Field>
  </Message>
```

Mapping RvXML

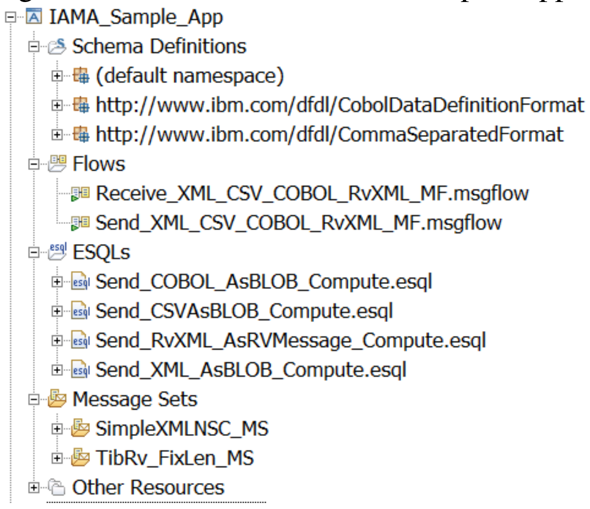
The rv.xsd schema defines the RvXML syntax supported by the TibRv nodes. Any of the WMB interfaces for working with message syntax trees (ESQL, Java, PHP, .net) can be used to construct or access an RvXML message syntax tree.

If you intend to use RvXML messages with a GUI mapping node then you must create a message model from the rv.xsd schema in order to use RvXML messages as a Source or Target message in the GUI mapping node.

IAMA Sample Message Flows

The IAMA Sample message flows are contained in the IAMA_Sample_PI.zip Project Interchange zip file. Import the IAMA_Sample_PI Project Interchange into your workspace to access the IAMA_Sample_App Application Project. NOTE: The IAMA plug-in nodes must be installed on the WMB Toolkit in order to work with the IAMA Sample message flows.

Following is the content of the IAMA_Sample_App Application Project:



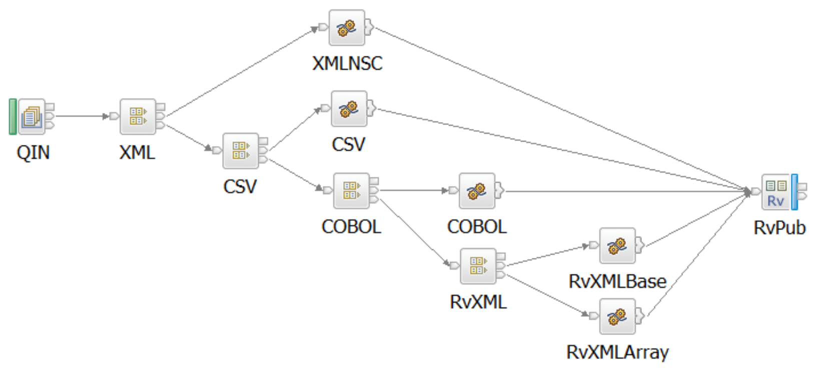
```
IAMA_Sample_App
  Schema Definitions
    (default namespace)
    http://www.ibm.com/dfdl/CobolDataDefinitionFormat
    http://www.ibm.com/dfdl/CommaSeparatedFormat
  Flows
    Receive_XML_CSV_COBOL_RvXML_MF.msgflow
    Send_XML_CSV_COBOL_RvXML_MF.msgflow
  ESQLs
    Send_COBOL_AsBLOB_Compute.esql
    Send_CSVAsBLOB_Compute.esql
    Send_RvXML_AsRVMessage_Compute.esql
    Send_XML_AsBLOB_Compute.esql
  Message Sets
    SimpleXMLNSC_MS
    TibRv_FixLen_MS
  Other Resources
```

The IAMA sample message flows demonstrate all Message Exchange Types being published by an "output" flow, and all Message Exchange Types being received by an "input" flow. The flows show passing simple XML, CSV and COBOL messages between Rendezvous and WMB but can easily be extended to handle more complex messages. As well, RvXML messages, containing all TIBRVMSG data types demonstrate how to work with any TibrvMsg structure.

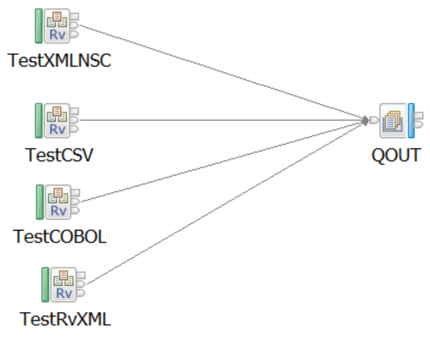
The TibRv nodes in the IAMA sample are configured to use Reliable messaging and Advisory Message Logging is not enabled for any of the TibRv nodes.

The IAMA supportpac includes the following sample message flows:

- Send_XML_CSV_COBOL_RvXML_MF -- Illustrates single TibRvOutput node publishing multiple subjects, using all Message Exchange Types. BLOBToRVXML uses an XMLNSC message-set, BLOBToRVString uses a DFDL CSV schema, BLOBToRVBinary uses a COBOL record and the MRM parser and XMLToRVMessage publishes all "base" and "array" TIBRVMSG data types. A total of 5 messages are published. Send any message to QIN to publish the 5 messages.



- Receive_XML_CSV_COBOL_RvXML_MF – Illustrates multiple TibRvInput nodes using all Message Exchange Types. RVBinaryToBLOB uses a COBOL record and the MRM parser, RVStringToBLOB uses a CSV message and DFDL, RVXMLToBLOB uses an XMLNSC message-set and RVMessageToXML uses all TIBRVMSG data types. A total of 5 messages are received and put to QOUT.



The IAMA Sample message-flows use the following MQ queues:

- QIN
- QOUT
- NOTE: Create these queues on your broker prior to deploying the IAMA_Sample_App Application.

To run the IAMA Sample message flows, deploy the IAMA _Sample_App Application to a broker of your choice after the broker has been property configured to run the TibRv nodes.

Send any message to QIN to run the Send_XML_CSV_COBOL_RvXML_MF flow. The flow publishes 5 different messages using 5 different subjects:

- XML – Simple XML format in XMLNSC message-set, Subject = TestRVXML
- CSV – Simple CSV format in DFDL, Subject = TestRVString
- COBOL – Simple fixed-length COBOL record using MRM parser, Subject = TestRVBinary

- RvXML – First message contains all “base” TIBRVMSG types, Subject = TestRvXML.Base. Second message contains all “array” TIBRVMSG types, subject = TextRvXML.Array. The ‘RvXMLBase” and “RvXMLArray” compute nodes demonstrate creation of all TIBRVMSG types using ESQL.

All output messages are built using ESQL. Each compute node overrides the Subject and rvMessageType properties for its specific configuration, allowing a single TibRvOutput node to be used for all publications. The ‘RvXMLBase” and “RvXMLArray” compute nodes demonstrate creation of all TIBRVMSG types using ESQL.

The Receive_XML_CSC_COBOL_RvXML_MF contains 4 TibRvInput nodes:

- TestXMLNSC – Subject=TextRVXML, Domain=XMLNSC, Set=SimpleXMLNSC_MS
- TestCSV – Subject=TestRVString, Domain=DFDL, Type=SimpleCSV
- TestCOBOL – Subject = TestRVBinary, Domain=MRM, Set=TibRv_FixLen_MS, Type=msg, Format=Binary1
- TestRvXML – Subject=TestRvXML.> (TestRvXML.Base, TestRvXML.Array)

Each node demonstrates a different Message Exchange Type using a different message domain – RVXMLToBLOB/XMLNSC, RVBinaryToBLOB/MRM, RVStringToBLOB/DFDL. As well, all TIBRVMSG data types are demonstrated for the RVMessageToXML message exchange type.

Each input message sent to the Send_XML_CSV_COBOL_RvXML_MF flow will publish 5 messages that are received by the 4 TibRvInput nodes. Each TibRvInput node receives a Message Syntax Tree built with the correct parser and meta-data and simply puts the message to the QOUT queue. There will be 5 messages in QOUT – 1 CSV, 1 Fixed-Length, 1 XML (<Msg>) and 2 RvXML (one contains all “base” types and the other contains all “array” types).

String and Binary Data Encoding Considerations

The TibRvOutput node uses the system default codepage when creating TibrvMsg fields of type TIBRVMSG_STRING and TIBRVMSG_XML; this applies to BLOBToRVString, BLOBToRVXML and XMLToRVMessage modes. Fields can be created using a different codepage by specifying a codepage for the Execution Group using “property1” of the TIBRV configurable service. The Execution Group codepage setting applies to TibRvOutput and TibRvInput nodes.

The TibRvInput node, when in RVStringToBLOB or RVXMLToBLOB mode, uses the system default codepage to generate the content bytes. The content bytes can be created using a different codepage by specifying a codepage for the Execution Group using “property1” of the TIBRV configurable service. This also applies to RVMessageToXML mode; fields of type TIBRVMSG_STRING and TIBRVMSG_XML will be decoded using the system default code page, or the Execution Group codepage setting from "property1" of the TIBRV Configurable Service Definition.

When using "Message Input Parsing" properties to parse content from a TibrvMsg you must consider the CCSID and binary Encoding of the content and the default CCSID and binary Encoding used by the Broker. By default the Broker's CCSID/Encoding will be used (these values are obtained from the Broker's queue manager). If the content to be parsed uses a different CCSID or Encoding then you must specify that CCSID and/or Encoding on the TibRvInput node's "TibRvMessage" property page. . NOTE: These properties are not implemented in release 1.0.0. A work-around is to specify the CCSID and/or Encoding in the Parse clause of a Create statement.

TIBRV Configurable Service Definition

The TIBRV Configurable Service definition must be created on each broker the TibRv nodes will be deployed to. The TIBRV Configurable Service definition describes how to load the Rendezvous jar and native library files, and the IAMA implementation jar. The TIBRV Configurable Service also defines the TibrvMsg String Encoding codepage, per Execution Group, that will be used for converting TibrvMsg String fields to Unicode Strings and for converting Unicode Strings to TibrvMsg String fields.

The TIBRV configurable service definition is based on the ConnectorProviders Configurable Service definition provided with WMB v8.0.0.1.

The TIBRV configurable service definition contains the following properties:

- **Name** = TIBRV
- **Type** = ConnectorProviders
- **connectorClassName** = com.ibm.broker.connector.tibrv.TibRvConnectorFactory
- **jarsURL** = {Path to directory containing connectortibrv_1.0.0.jar};{Path to tibrvnative.jar under the Revdezvous installation}

  NOTE: On Windows systems these paths are separated by a ";", and on Linux/Unix systems these paths are separated by a ":". The directory containing connectortibrv_1.0.0.jar was created as part of the manual IAMA installation process.

  Example:

  Windows: C:\IBM\IAMA;C:\TIBCO\tibrv\8.2.2\lib\tibrvnative.jar
  Linux/Unix: /opt/ibm/iama:/opt/tibcorv/8.2.2/lib/tibrvnative.jar

- **nativeLibs** = Path to directory under the Rendezvous installation that contains the tibrv .dll files (Windows) or tibrv .so files (Unix/Linux).

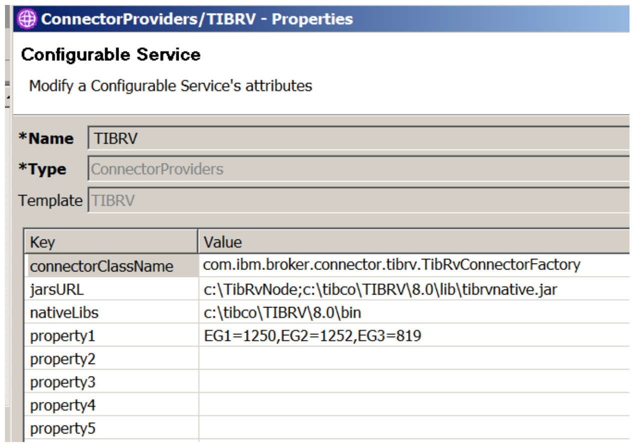
  NOTE: This directory contains the Rendezvous native libraries (.dll on Windows or .so on Linux/Unix) that are required by the TIBRV provider. Typically, on Windows this is the "bin" directory under the Rendezvous installation, and on Unix/Linux this is the "lib" directory under the Rendezvous installation.

- property1 -- Override the "default" TibrvMsg String Encoding codepage for an Execution Group using "property1". "property1" is a comma-separated list of EGName=Codepage pairs. (Example: EG1=1250,EG2=1252,EG3=819). The string encoding codepage specified in the TIBRV configurable service affects both TibRvInput and TibRvOutput nodes. See the TIBCO Rendezvous Concepts manual for details regarding the TibrvMsg String encoding codepage setting.
- property2 – Leave blank

- property3 – Leave blank
- property4 – Leave blank
- property5 – leave blank

The TIBRV ConnectorProviders configurable service definition can be created using the mqsicreateconfigurableservice command or by using MBX, and can be modified using the mqsichangeproperties command. You must recycle the broker for changes to the TIBRV ConnectorProviders configurable service to take effect.

```
mqsicreateconfigurableservice BROKER -c ConnectorProviders -o TIBRV -n "
\"connectorClassName\", \"jarsURL\", \"nativeLibs\", \"property1\", \"property2\",
\"property3\", \"property4\", \"property5\" " -v "
\"com.ibm.broker.connector.tibrv.TibRvConnectorFactory\",
\"c:\TibRvNode;c:\tibco\TIBRV\8.0\lib\tibrvnative.jar\",
\"c:\tibco\TIBRV\8.0\bin\", \"EG1=1250,EG2=1252,EG3=819\", \"\", \"\", \"\",
\"\" "
```



Configurable Service

Modify a Configurable Service's attributes

*Name: TIBRV

*Type: ConnectorProviders

Template: TIBRV

Key	Value
connectorClassName	com.ibm.broker.connector.tibrv.TibRvConnectorFactory
jarsURL	c:\TibRvNode;c:\tibco\TIBRV\8.0\lib\tibrvnative.jar
nativeLibs	c:\tibco\TIBRV\8.0\bin
property1	EG1=1250,EG2=1252,EG3=819
property2	
property3	
property4	
property5	

The TIBRV Activity Log

Each Execution Group with deployed TibRv nodes contains a TIBRV Activity Log that deployed TibRv nodes use to log activity information
.
The TibRv nodes write information to the TIBRV Activity Log for the following reasons:

- TibRvInput node is accessed
- TibRvOutput node is accessed
- Rendezvous Advisory Message Logging
- TibRv Administration Command output

Each time a TibRvInput or TibRvOutput is accessed a message is written to the TIBRV Activity Log under the Execution Group the node is deployed to.

The TibRvInput and TibRvOutput nodes may be configured to log Rendezvous advisory messages associated with their underlying transport. Each Advisory Message that is received will be written to the TIBRV Activity Log under the Execution Group the node is deployed to. The "advisory" administration command allows a node's "Advisory Message Logging" to be modified while it is deployed and running. This allows "Advisory Message Logging" to be used only when it is required.

"Advisory Message Logging" creates additional columns to display the following information:

- Advisory Source – SYSTEM or CM
- Advisory Class -- INFO, WARN or ERROR
- Advisory Name – The "Name" component of the Advisory Message
- Task – AdvisoryMessage
- NOTE: You must select these columns for display

The TibRvOutput node, when using Certified Message Delivery, supports numerous commands for querying and modifying the node's ledger. Each administration command writes a completion message (success or failure) to the TIBRV Activity Log under the Execution Group the node is deployed to. The "reviewLedger" command will actually write several messages to the TIBRV Activity Log. The first message indicates that the "reviewLedger" command has started. There will be one additional message for each "subject" that is returned as part of the query. And the final message will indicate that the "reviewLedger" command has completed.

TibRv Administration Commnads create an additional column to display the following information:

- Task – reviewLedger:CMName:[start|onLedgerMsg|finish]
- NOTE: CMName is the node's Certified Messaging Correspondent's Name property. "start" indicates that the reviewLedger command has started. "onLedgerMsg" indicates a "subject" messages, which contains one or more Listeners that have received certified messages for the "subject."

Use the Message Broker Explorer (MBX) to view and filter the TIBRV Activity Log by expanding Brokers➔Broker➔ExecutionGroup➔ResourceManagers, under the "Navigator" view. Display the contents of the TIBRV Activity Log by right-clicking on the TIBRV resource manager and selecting "Open Activity Log" from the context-menu .

TibRv Node User and Service Trace

The TibRv Nodes support both WMB Service and User tracing.

User Trace is one of two types of optional tracing available in WMB and provides more information than that provided by the entries that are written to the Administration log. User trace is inactive by default; you must activate it explicitly by using a command, or by selecting options in the WebSphere Message Broker Toolkit.

Service Trace is the other of two types of optional tracing available in WMB and provides more information than that provided by the entries that are written to the Event Log or to User trace. Service trace is inactive by default; you must activate it explicitly by using a command.

The TibRv nodes support the “normal” User trace level and will display information regarding the creation of TibRvInput/TibRvOutput nodes, TibrvMsgs that are published and TibrvMsgs that are received.

See WMB InfoCenter for information regarding starting/stopping/formatting User and Service traces.

NOTE: You may be instructed by IBM support to perform a Service trace for problems encountered with the IAMA supportpac. The TibRv node Service trace information is very detailed and will not be described here.

Troubleshooting

In the case that a problem is encountered using the TibRv nodes using the following techniques to resolve or investigate the problem:

- Check the Broker Log file for exception messages related to the TibRv nodes. The TibRv nodes catch and log most errors/exceptions, so this a good source for the cause of most problems.
- Check the Execution Group console file. The TibRv nodes print a stack trace for all exceptions, so checking this file is a good source if the Broker log did not reveal the problem. The console file is located at {MQSI_WORKPATH}\components\BrokerName\ExecutionGroupUUID\console.txt for Windows; /var/mqsi/components/BrokerName/ExecutionGroupUUID/stdout and stderr for Unix/Linux

  Check the console.txt/stdout file for tibrv exceptions. tibrv exceptions can be traced back to the tibrv called made by the IAMA supportpac. This will include the source file and line number.
- If the exception occurs during deployment, check that the tibrv daemon is running. If the daemon is not running, try starting it manually and try deploying the flows containing the TibRv nodes again.
- If the exception occurs when first trying to start the broker after creating the TIBRV configurable service, check the Execution Group console.txt/stdout file for errors that are causing the TIBRV provider from loading correctly. Check for typos or invalid paths in the TIBRV configurable service. Also, check that the broker has the proper permissions to access all directories and files.
- If you cannot resolve the problem contact IBM support.