BEA WebLogic® Integration

TIBCO Rendezvous Control and Event Generator User Guide

Version 10.2
Document Revised: March 2008

Contents

CHAPTER 1

TIBCO Rendezvous Event Generator

TIBCO® Rendezvous™ (a product from TIBCO: http://www.tibco.com) enables exchange of data across applications running on distributed platforms. TIBCO Rendezvous (TIBCO RV) Event Generator is one of the WebLogic Integration™ event generators that you can create from the WebLogic Integration Administration Console. The TIBCO RV event generator listens for messages on a subject and raises events to the message broker on receiving the desired message.

Disclaimer

Use of the TIBCO RV control and event generator with BEA WebLogic Integration in no manner confers or grants the right to use TIBCO Rendezvous including "dynamic libraries". In order to use such TIBCO products, the user of the TIBCO RV control and event generator must obtain a valid license from TIBCO. For more information about the licensed copy of Rendevous, see `http://www.tibco.com`.

Topics Included in This Section

Overview: TIBCO RV Event Generator

Describes the function of the TIBCO RV event generator within WebLogic Integration.

Creating and Using the TIBCO RV Event Generator

Describes how to create an event generator and to use it for receiving messages.

Overview: TIBCO RV Event Generator

The TIBCO RV event generator enables WebLogic Integration generate events to message broker channels. The messages are received in most formats supported by Rendezvous, converted to binary and then published to the WebLogic Integration message broker.

Using TIBCO RV event generator, you can receive messages over the base Rendezvous and certified messaging (CM) transports. Certified messages can be received in either single or distributed queues, with support for clustering in the distributed queues processing. Receipt of each message can be acknowledged by implicit or explicit confirmation.

To learn more, see Event Generators in *Using the WebLogic Integration Administration Console*.

Clusters

TIBCO RV event generators can be deployed on a cluster with load balancing, high availability and failover features. They use distributed queues to support these clustering features. All event generators are automatically deployed on all managed servers from the WebLogic Integration Administration Console.

TIBCO RV event generators on a cluster subscribe to a subject with a single RV daemon machine, a distributed queue option and a distinct CM name entered at the time of event generator creation. Distributed queues work with the concept of scheduler and workers. One of the queues will act as a scheduler and others as workers.A scheduler distributes the messages to workers on a round robin basis, making sure the message is received by one and only one worker. This is also referred to as the “once and only once delivery”.

Load balancing

The scheduler node sends messages to the worker nodes on a round robin basis. This is done by checking for pending tasks at individual workers end. Depending on the which worker is relatively free, the scheduler will assign the task. This is termed as load balancing.

High Availability

This implies that at any given instance, a worker and a scheduler node is always available. If a worker node goes down with a managed server, another worker will be available. If a scheduler node goes down with a managed server, a worker will take over the role of the scheduler node.

Failover

This indicates that even if a worker node goes down before acknowledging receipt of message, the scheduler node will re-assign the task to the next available worker node.

Prerequisites to Using the TIBCO RV Event Generator

Before adding the TIBCO RV Event Generator to the WebLogic Platform, perform the following:

After installation of Tibco, ensure following before starting the server.

1. Install and configure TibcoRV on your machine. Ensure that the installation directory should be in server PATH.
2. The jar file tibjrv.jar must be there in CLASSPATH,

   For Solaris:

   ```
   export LD_LIBRARY_PATH = $HOME/local/jars/tibjrv.jar
   ```

   For Windows:

   ```
   EXT_PRE_CLASSPATH=D:\Installs\TIBCO\TIBRV\lib\tibrvj.jar;%EXT_PRE_CLASS
   PATH%
   ```

3. TIBCO installation bin and lib folder should be there in PATH and LD_LIBRARY_PATH respectively.

   ```
   export LD_LIBRARY_PATH=/opt/tibco/tibrv/lib:$LD_LIBRARY_PATH
   ```

   ```
   export PATH=/opt/tibco/tibrv/bin:$PATH
   ```

After successful deployment, you will be able to create event generators from the WebLogic Integration console.

Creating and Using the TIBCO RV Event Generator

Perform the following steps to create a TIBCO RV event generator.

1. Enter the following URL in your html browser:

`http://localhost:`**`port`**`/wliconsole`

   Replace **`port`** with the specific port id, for example `7001`.

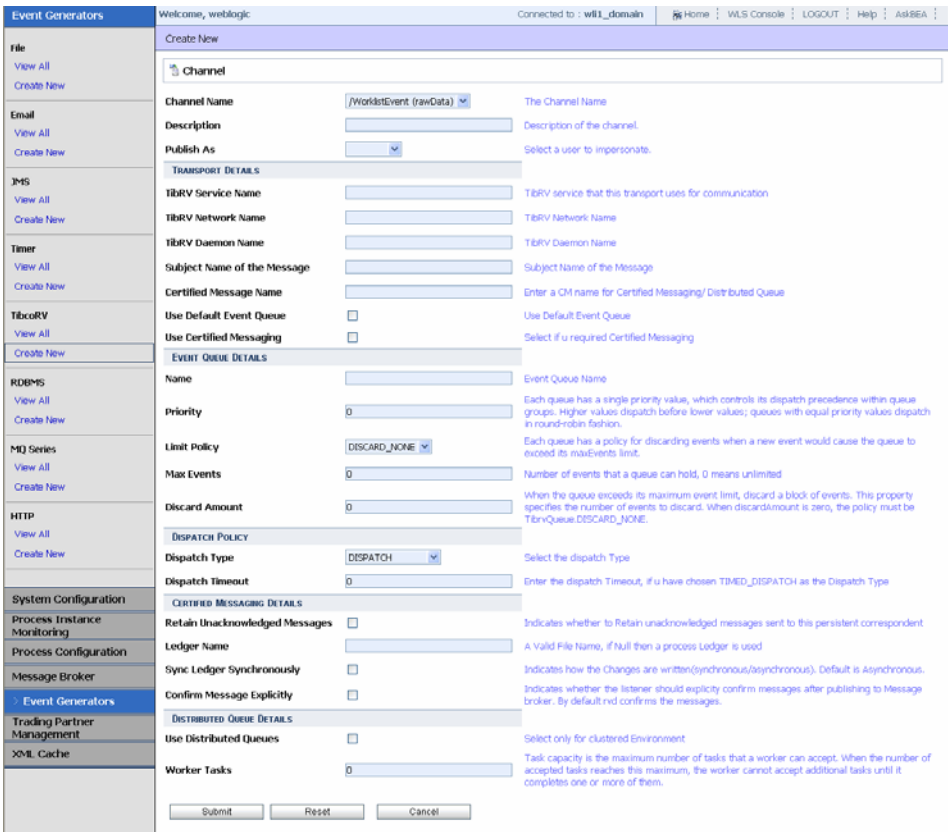
2. In the WebLogic Integration Administration Console home page, click **Event Generators** to display the Event Generators home page.

3. Click **TIBCO RV** option in the left frame and select **Create New** option that appears below it, to create an event generator.

4. In the **Generator Name** field, enter a unique name for the new event generator and click **Submit**.

5. In the next frame, click **Define a New Channel Rule** to display a form (see Figure 1-1)

Figure 1-1 .Creating TIBCO RV Event Generator: Channel Rule Definition



6. Enter the desired information in the fields; a brief description of each is displayed adjacent to the field. Refer TIBCO Rendezvous documentation for information on Rendezvous transport, event queues and certified messaging parameters.

   Note: Select **Use Distributed Queues** option for the event generator to work on a cluster. Doing this would also make it mandatory to specify a CM name.

7. Click **Submit** to effect creation of the channel rule and the event generator.

Note: While creating an event generator, if incorrect Rendezvous related values are entered (like network, daemon and so on), the event generator is created but, a runtime exception error will be displayed.

These Rendezvous properties are not verified at the form submission stage. These values are used only when the application attempts to connect to the specified Rendezvous daemon; hence, the runtime error.

On successful creation of an event generator, a `WLI_TIBRV_`**`event-gen`**`.jar` file is created in the WebLogic server domain folder. Here, **`event-gen`** is the unique name of the event generator as specified in step 4 above. This file connects to the Rendezvous daemon, as specified in the Channel Rule Definition form, and creates a listener on the subject.

Note: Always create a single rule definition for each unique event generator. Whenever an event generator is created with multiple channel rule definitions, only the first channel rule definition is recorded and used.

Explicit Confirmation

Explicit confirmation directs the listener to explicitly confirm delivery of message after publishing to the message broker. To employ the explicit confirmation feature, select the **Confirm Message** option while defining the rules for an event generator.

If **explicit confirmation** is selected in the OAM console, the TIBCO RV event generator will confirm the message only on successful completion of the subscribed JPDs.

Notes: Process.java files need to have a synchronous subscription to confirm explicitly.

If any of the subscribed `Process.java` files throw an exception error, the TIBCO RV event generator will not confirm the message.

Retrieving Messages

Once the event generator has been created, it will start publishing messages on the subscribed subject to the WebLogic Integration message broker channel. Applications subscribed to that channel will receive the messages in raw data format. To retrieve the content of the message, insert a perform node with the following code:

```
TibrvMsg RecvdMsg = new TibrvMsg(receivedData.byteValue());
```

Where **`receivedData`** is of type `com.bea.data.RawData` and contains the message published by the TIBCO RV event generator.

In addition, you need to edit the following annotation in the JPD file:

```
@com.bea.wli.control.broker.MessageBroker.StaticSubscription(channelName =
"/TimeOut/CMrequest/TibcoRawDataChannel",messageBody = "{x0}",
/
```

to read as follows:

```
@com.bea.wli.control.broker.MessageBroker.StaticSubscription(channelName =
"/TimeOut/CMrequest/TibcoRawDataChannel",messageBody = "{x0}",
messageMetaData = "{x1}")
```

Important: The parameter used in the annotation also needs to be added to the method definition. For an example, refer to Retrieving Information From a TIBCO RV Event Generator Header.

TIBCO RV Event Generator Header

This section provides the schema of a TIBCO RV event generator header and an example code to retrieve information from the header.

Schema of a TIBCO RV Event Generator Header

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://www.bea.com/wli/control/TibHeaders"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.bea.com/wli/control/TibHeaders"
elementFormDefault="qualified">
   <xs:element name="TibHeaders">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="ReplySubject" type="xs:string" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="SendSubject" type="xs:string" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="TibrvTransport" type="Transport"
minOccurs="0" maxOccurs="1"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
      <xs:complexType name="Transport">
         <xs:sequence>
         <xs:element name="Service" type="xs:string" minOccurs="0"
```

```
maxOccurs="1"/>
         <xs:element name="Network" type="xs:string" minOccurs="0"
maxOccurs="1"/>
         <xs:element name="Daemon" type="xs:string" minOccurs="0"
maxOccurs="1"/>
         </xs:sequence>
      </xs:complexType>
</xs:schema>
```

Retrieving Information From a TIBCO RV Event Generator Header

Following is a code example for retrieving **`replySubject`** from the TIBCO RV event generator header.

```
@com.bea.wli.control.broker.MessageBroker.StaticSubscription(channelName
="/Soak/reply/TibcoDataChannel"messageBody = "{x0}", messageMetaData =
"{x1}")

public void subscription(com.bea.data.RawData x0, com.bea.xml.XmlObject x1)
{
   //#START: CODE GENERATED - PROTECTED SECTION - you can safely add code
above this comment in this method. #//
   // input transform
   // parameter assignment
   this.recievedBytes = x0;
   this.Header = x1;
   //#END  : CODE GENERATED - PROTECTED SECTION - you can safely add code
below this comment in this method. #//
   TibHeaders
tibHeader=TibHeadersDocument.Factory.newInstance().addNewTibHeaders();
      tibHeader.set(this.Header);
   String replySubject=tibHeader.getReplySubject();
}
```

Index