Application Notes for Message Queue Interface (MQI) Integration Between the Avaya Event Processor and IBM WebSphere MQ - Issue 1.0

Abstract

These Application Notes describe a sample configuration for Message Queue Interface (MQI) integration between the Avaya Event Processor (EP) and IBM WebSphere MQ (WMQ). IBM WMQ is a messaging middleware platform that allows heterogeneous enterprise applications and systems to exchange data reliably and asynchronously. Enterprise applications and systems can use either the MQI or Java Message Service (JMS) API to access messaging resources on IBM WMQ. In these Application Notes, the Avaya EP is configured to use the MQI API.
1. Introduction

These Application Notes describe a sample configuration for Message Queue Interface (MQI) integration between the Avaya Event Processor (EP) and IBM WebSphere MQ (WMQ). IBM WMQ is a messaging middleware platform that allows heterogeneous enterprise applications and systems to exchange data reliably and asynchronously. Enterprise applications and systems can use either the MQI or Java Message Service (JMS) API to access messaging resources on IBM WMQ. In these Application Notes, the Avaya EP is configured to use the MQI API.

Multiple MQ queue managers may be configured and run on an IBM WMQ server, and each MQ queue manager may be configured with multiple MQ queues. In the context of the MQI architecture, the Avaya EP is an MQI-capable application that connects to one or more MQ queue managers and accesses MQ queues on the MQ queue managers to exchange data with other MQI-capable applications. Upon reception of an MQI message from an MQ queue, the Avaya EP can extract relevant data from the message and insert the data into one or more Avaya EP “Streams”. Similarly, the Avaya EP can extract data from one or more Avaya EP Streams and send the data in an MQI message to another MQ queue.

2. Configuration

The sample configuration used in these Application Notes is shown in Figure 1.

![Figure 1: Configuration](image)

In the sample configuration, IBM WMQ is installed on a Microsoft Windows 2003 Server, and the Avaya EP is installed on a Red Hat Enterprise Linux ES 4.0 server. MQI-capable applications and the Avaya EP exchange data by sending and receiving MQI messages to and from one or more specified MQ queues configured on IBM WMQ. Consult the IBM WMQ documentation [3] for further details on WMQ configuration.
3. Equipment and Software Validated

The following equipment and software were used for the sample configuration:

<table>
<thead>
<tr>
<th>Equipment &amp; Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avaya Event Processor running on Red Hat Enterprise Linux ES Release 4.0 (Nahant Update 5) Server</td>
<td>2.0.110_GA_26</td>
</tr>
<tr>
<td>IBM WebSphere MQ</td>
<td>6.0.2.2</td>
</tr>
</tbody>
</table>

Table 1: Equipment/Software List

4. Avaya Event Processor

4.1. Event Processing Language

This section describes the Event Processing Language (EPL) statements for configuring the Avaya EP to receive and send MQ messages from and to MQ queues on an IBM WMQ server. The EPL statements may be executed dynamically on a running Avaya EP server instance, or saved in EPL files. EPL files are saved in the [Avaya EP installation directory]/eplserver/[Avaya EP server instance]/engine/epl directory, and are automatically executed when the Avaya EP server instance is started. Consult the Avaya EP documentation [1][2] for further details on EPL structure and syntax, and EPL files.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Create a Stream in the Avaya EP into which data extracted from received MQ messages can be inserted. The sample EPL statement below creates a Stream named “MyMQReceiveStream”, where each data record in the Stream will consist of two fields, “ITEM” and “PRICE”. Note that each field name must match a name-value pair in the received MQ message.</td>
</tr>
</tbody>
</table>

```epl
CREATE STREAM system:MyMQReceiveStream
TYPE {
    ITEM String,
    PRICE Double
}
```
Step | Description
--- | ---
2. | Create a Stream Provider in the Avaya EP that receives MQ messages from an MQ queue on the IBM WMQ server, and inserts the data in the MQ messages into the Stream configured in Step 1. Configure the Stream Provider with the built-in type “MQ_PROVIDER” and the following parameters:
  * **queueManagerName** – name of MQ queue manager configured on the IBM WMQ server in Section 5 Steps 1 - 6.
  * **queueName** – name of MQ queue configured on the IBM WMQ server in Section 5 Steps 7 - 10 from which the Avaya EP will receive MQ messages.
  * **host** – IP address, hostname, or FQDN of the IBM WMQ server.
  * **channel** – name of MQ channel configured on the IBM WMQ server in Section 5 Steps 12 - 15.
  * **port** – port number configured on the IBM WMQ server in Section 5 Step 5.

The sample EPL statement below creates a Stream Provider of type “MQ_PROVIDER” named “MyMQReceiveStreamProvider” that receives MQ messages from the MQ queue “Queue1” on the MQ queue manager “MQQueueManager1” residing at 135.8.139.234, and inserts the data in the MQ messages into the “MyMQReceiveStream” Stream.

```
CREATE STREAMPROVIDER system:MyMQReceiveStreamProvider MQ_PROVIDER ON STREAM system:MyMQReceiveStream {
  queueManagerName 'MQQueueManager1',
  queueName 'Queue1',
  host '135.8.139.234',
  channel 'Channel1',
  port 1415
}
```

3. | Create a Stream in the Avaya EP from which data can be extracted and sent in MQ messages. The sample EPL statement below creates a Stream named “MyMQSendStream”, where each data record in the Stream will consist of two fields, “ITEM” and “PRICE”.

```
CREATE STREAM system:MyMQSendStream TYPE {
  ITEM String,
  PRICE Double
}
```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 4.   | Create a Stream Listener in the Avaya EP that listens on the Stream configured in Step 3 and sends the data in the Stream to another MQ queue on the IBM WMQ server. Configure the Stream Listener with the built-in type “MQ_LISTENER” and the following parameters:  
  - `queueManagerName` – name of MQ queue manager configured on the IBM WMQ server in Section 5 Steps 1 - 6.  
  - `queueName` – name of MQ queue configured on the IBM WMQ server in Section 5 Step 11 to which the Avaya EP will send MQ messages.  
  - `host` – IP address, hostname, or FQDN of the IBM WMQ server.  
  - `channel` – name of MQ channel configured on the IBM WMQ server in Section 5 Steps 12 - 15.  
  - `port` – port number configured on the IBM WMQ server in Section 5 Step 5.  
  
  The sample EPL statement below creates a Stream Listener of type “MQ_LISTENER” named “MyMQSendStreamListener” that extracts data from the “MyMQSendStream” Stream and sends the data in MQ messages to the MQ queue “Queue2” on the MQ queue manager “MQQueueManager1” residing at 135.8.139.234.  
  
  ```java  
  CREATE STREAMLISTENER system:MyMQSendStreamListener MQ_LISTENER ON STREAM system:MyMQSendStream  
  {  
    queueManagerName 'MQQueueManager1',  
    queueName 'Queue2',  
    host 135.8.139.234,  
    channel 'Channel1',  
    port 1415  
  }  
  ```  

4.2. JAR Files  

This section lists the JAR files that must be included in the Avaya EP server classpath at runtime. The JAR files should follow the “ep-classpath.jar” JAR file in the runtime classpath.  

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1.   | Add the following IBM WMQ JAR file to the [Avaya EP installation directory]/lib/third_party directory:  
  - `com.ibm.mq.jar`  
  
  **Note:** In these Application Notes, a directory named “IBMMQ” was created in the [Avaya EP installation directory]/lib/third_party directory and the above JAR files were placed in the [Avaya EP installation directory]/lib/third_party/IBMMQ directory.
### 4.3. Startup Script

The [Avaya EP installation directory]/eplserver/[Avaya EP server instance]/bin/eplServer.sh file (or eplServer.bat for the Avaya EP development version running on a Microsoft Windows server) is a script for starting an Avaya EP server instance.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1.   | If a copied and renamed “setEnv.sh” (or “setEnv.bat”) file was used in Section 4.2 Step 2, then modify the default environment setting in the “eplServer.sh” (or “eplServer.bat”) file with the name of the renamed file. For example:  

Linux: . /usr/local/EP/bin/setEnvIBMMQ.sh  
Windows: call “C:\Program Files\Avaya\Event Processor\bin\setEnvIBMMQ.bat”  

| 2.   | If the Avaya EP is unable to resolve the hostname or FQDN of the IBM WMQ server, then add an entry with the IP address, hostname, and FQDN of the IBM WMQ server to the /etc/hosts (Linux) or C:\WINDOWS\system32\drivers\etc\hosts file on the Avaya EP server. |
5. IBM WebSphere MQ

This section describes the steps for configuring MQ queue managers, queues, and channels on IBM WMQ.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>On the IBM WMQ server, launch the IBM WebSphere MQ Explorer. In the left pane of the WebSphere MQ Explorer, right-click on Queue Managers, and select New → Queue Manager.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 2.   | In Step 1 of the **Create Queue Manager** wizard, enter a descriptive **Queue manager name** and click on “Next”.

![Create Queue Manager Wizard](image1.png)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 3.   | In Step 2 of the **Create Queue Manager** wizard, click on “Next”.

![Create Queue Manager Wizard](image2.png)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>In <strong>Step 3</strong> of the <strong>Create Queue Manager</strong> wizard, check the <strong>Create server-connection channel</strong> checkbox and click on “Next”.</td>
</tr>
<tr>
<td><img src="image1" alt="Create Queue Manager" /></td>
<td><img src="image2" alt="Create Queue Manager" /></td>
</tr>
</tbody>
</table>

| 5.   | In **Step 4** of the **Create Queue Manager** wizard, enter an unused TCP port number for **Listen on port number** and click on “Next”. |
| ![Create Queue Manager](image1) | ![Create Queue Manager](image2) |
6. In Step 5 of the Create Queue Manager wizard, click on “Finish”.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 6.   | In Step 5 of the Create Queue Manager wizard, click on “Finish”.

![Create Queue Manager](image_url)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.</td>
<td>In the left pane of the <strong>WebSphere MQ Explorer</strong>, expand the MQ queue manager created in Steps 1 – 6, right-click on <strong>Queues</strong>, and select <strong>New → Local Queue</strong>.</td>
</tr>
</tbody>
</table>

![WebSphere MQ Explorer screenshot](image)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.</td>
<td>In the <strong>Create a Local Queue</strong> step of the <strong>New Local Queue</strong> wizard, enter a descriptive <strong>Name</strong> and click on “Next”.</td>
</tr>
</tbody>
</table>

![Screenshot of Create a Local Queue dialog box](image-url)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.</td>
<td>In the <strong>Change properties</strong> step of the <strong>New Local Queue</strong> wizard, click on “Finish”. In these Application Notes, the Avaya EP is a consumer of MQ messages from the MQ queue created in Steps 7 – 9, i.e., the Avaya EP will receive MQ messages from this MQ queue.</td>
</tr>
<tr>
<td>10.</td>
<td>In the confirmation window, click on “OK”.</td>
</tr>
<tr>
<td>11.</td>
<td>Repeat Steps 7 – 10 to create another MQ queue, using another <strong>Name</strong>, e.g., “Queue2”. In these Application Notes, the Avaya EP is a producer of MQ messages to this MQ queue, i.e., the Avaya EP will send MQ messages to this MQ queue.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>12.</td>
<td>In the left pane of the <strong>WebSphere MQ Explorer</strong>, expand the MQ queue manager created in Steps 1 – 6, expand <strong>Advanced</strong>, right-click on <strong>Channels</strong>, and select <strong>New Æ Server-connection Channel</strong>.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>13.</td>
<td>In the <strong>Create a Server-connection Channel</strong> step of the <strong>New Server-connection Channel</strong> wizard, enter a descriptive “<strong>Name</strong>” and click on “<strong>Next</strong>”.</td>
</tr>
</tbody>
</table>

![Create a Server-connection Channel wizard](image)
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.</td>
<td>In the <strong>Change properties</strong> step of the <strong>New Server-connection Channel</strong> wizard, click on “<strong>Finish</strong>”.</td>
</tr>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>15.</td>
<td>In the confirmation window, click on “<strong>OK</strong>”.</td>
</tr>
</tbody>
</table>
6. Verification Steps

The following steps may be used to verify the configuration:

- From an MQ-capable application, send an MQ message to the receive MQ queue (e.g., “MQQueue1”) on the MQ queue manager (e.g., “MQQueueManager1”) running on the IBM WMQ server and verify that the Avaya EPL application receives the message and inserts the data into the appropriate Stream.
- Insert data into the Avaya EP Stream (e.g., “MyMQSendStream”) and verify that the MQ queue manager (e.g., “MQQueueManager1”) running on the IBM WMQ server receives on the send MQ queue (e.g., “MQQueue2”) an MQ message containing the inserted data.

Notes:
1. The Avaya EP Console web application may be used to view data in Streams and insert data into Streams.
2. If an external MQ-capable application is not available, configure the Avaya EPL application to send and receive MQ messages to and from the same MQ queue (i.e., loopback). For example, set the “queueName” parameters in the Stream Provider and Stream Listener configurations in Section 4.1 to the same MQ queue (e.g., “Queue1”), and create the Event Definitions below in the Avaya EP to verify the configuration.

```sql
CREATE EVENT DEFINITION system:SendEvent
ON
  CLOCK.SCHEDULE(NOW(), TIME(0, 0, 30))
THEN
  MyMQSendStream.INSERT(
    ITEM='Shades',
    PRICE=20.95
  )

CREATE EVENT DEFINITION system:ReceiveEvent
ON
  MyMQReceiveStream
THEN
  PRINT('Got input: ITEM='
    + MyMQReceiveStream.ITEM
    + ' PRICE=' + MyMQReceiveStream.PRICE
  )
```

The “SendEvent” Event Definition inserts an event (data record) into the “MyMQSendStream” Stream every thirty seconds, and the “ReceiveEvent” Event Definition prints the contents of each received event (data record) in the “MyMQReceiveStream” Stream. If the Stream Listener and Stream Provider in Section 4.1 are both configured to access the same MQ queue, then the following should be printed approximately every thirty seconds:

```
Got input: ITEM=Shades PRICE=20.95
```
7. Conclusion
These Application Notes described a sample configuration for Message Queue Interface (MQI) integration between the Avaya Event Processor (EP) and IBM WebSphere MQ (WMQ). IBM WMQ is a messaging middleware platform that allows heterogeneous enterprise applications and systems to exchange data reliably and asynchronously. Enterprise applications and systems can use either the MQI or Java Message Service (JMS) API to access messaging resources on IBM WMQ. In these Application Notes, the Avaya EP is configured to use the MQI API.

8. Additional References
The following document may be found under the [Avaya EP installation directory]/docs directory on the Avaya EP server.

[1] “Getting Started with the Avaya Event Processor 2.0”

The following document may be obtained from http://support.avaya.com.


© 2008 Avaya Inc. All Rights Reserved.
Avaya and the Avaya Logo are trademarks of Avaya Inc. All trademarks identified by ® and ™ are registered trademarks or trademarks, respectively, of Avaya Inc. All other trademarks are the property of their respective owners. The information provided in these Application Notes is subject to change without notice. The configurations, technical data, and recommendations provided in these Application Notes are believed to be accurate and dependable, but are presented without express or implied warranty. Users are responsible for their application of any products specified in these Application Notes.

Please e-mail any questions or comments pertaining to these Application Notes along with the full title and filename, located in the lower right corner, directly to the Avaya Solution & Interoperability Test Lab at interoplabnotes@list.avaya.com