Getting started with record and replay in WebSphere Message Broker V8

Dominic Storey
Anton Piatek
Peter Masters
Steve Haskey

December 05, 2012

WebSphere Message Broker V8 introduces the new record and replay feature, which lets you record and view messages for audit or problem determination purposes, and replay them after problem resolution or other downtime issues. This tutorial shows you how to use event monitoring to emit a message from a flow, how to configure record and replay, and how to use the new web UI to view messages that you have recorded and replay them to a WebSphere MQ queue.

Before you start

About this tutorial

This tutorial shows you how to record, view, and replay messages using IBM® WebSphere® Message Broker V8.0.0.1.

Prerequisites

To benefit from this tutorial, you should have some experience with WebSphere Message Broker and have a basic understanding of message flows and WebSphere MQ messaging. You do not need to be an expert in these areas, since the tutorial provides sample files or detailed instructions.

System requirements

- WebSphere Message Broker V8.0.0.1 Runtime and Toolkit
- A WebSphere Message Broker instance, which you can create using the Default Configuration Wizard in Message Broker Toolkit or Message Broker Explorer.
- Record and Replay requires a database. This article uses IBM® DB2®, which you can either install locally or use remotely. For information on supported levels of DB2, see WebSphere Message Broker requirements. Oracle is also supported but is not covered in this tutorial.

Setting up the Web UI

To start the Web UI, you must create a broker to host it, and then apply some configuration properties against that broker.

Creating a broker

If you've already created a broker, you can skip this step. A quick way to create a broker is via the Default Configuration Wizard in the WebSphere Message Broker Toolkit:

1. Start the Default Configuration Wizard from the WebSphere Message Broker Toolkit Welcome page, which is displayed the first time you start the WebSphere Message Broker Toolkit. If the Welcome page is not displayed, open it in the WebSphere Message Broker Toolkit by clicking **Help => Welcome**.
2. Click **Get Started** on the Welcome page, then click **Create the Default Configuration**.
3. Click **Start the Default Configuration Wizard**.
4. The welcome page of the Wizard describes what is about to happen. Click **Next** to continue. The Wizard checks whether the default configuration has already been created.
5. After the Wizard runs successfully, click **Next** and then **Finish**. The wizard creates default broker named MB8BROKER.

Configuration properties

You can configure the Web UI to use either HTTP or HTTPS on any port. This tutorial assumes that the default broker's name is MB8BROKER and that you want to use HTTP on Port 4414, which is set by default. If you have another broker, then you may need to set the port manually. If you want the Web UI to use a another port, then change the commands accordingly.

1. Set the HTTP port by running the mqsichangeproperties command in the command console:
   ```
   mqsichangeproperties MB8BROKER -b webadmin -o HTTPConnector -n port -v 4414
   ```
2. To confirm that the properties have been set correctly, run the mqsireportproperties command, as shown below:
   ```
   mqsireportproperties MB8BROKER -b webadmin -o HTTPConnector -a
   ```

   This command produces a response similar to this truncated example:
   ```
   HTTPConnector
   uuid='HTTPConnector'
   address=''
   port='4414'
   allowTrace=''
   . . .
   ```
3. By default, the Web Administration Interface is enabled for new brokers. For migrated brokers you must enable the Web Administration Interface yourself. To do so, run the mqsichangeproperties command in the command console:
   ```
   mqsichangeproperties MB8BROKER -b webadmin -o server -n enabled,enableSSL
       -v true,false
   ```
4. To confirm that the properties have been set correctly, run the mqsireportproperties command, as shown below:
   ```
   mqsireportproperties MB8BROKER -b webadmin -o server -a
   ```

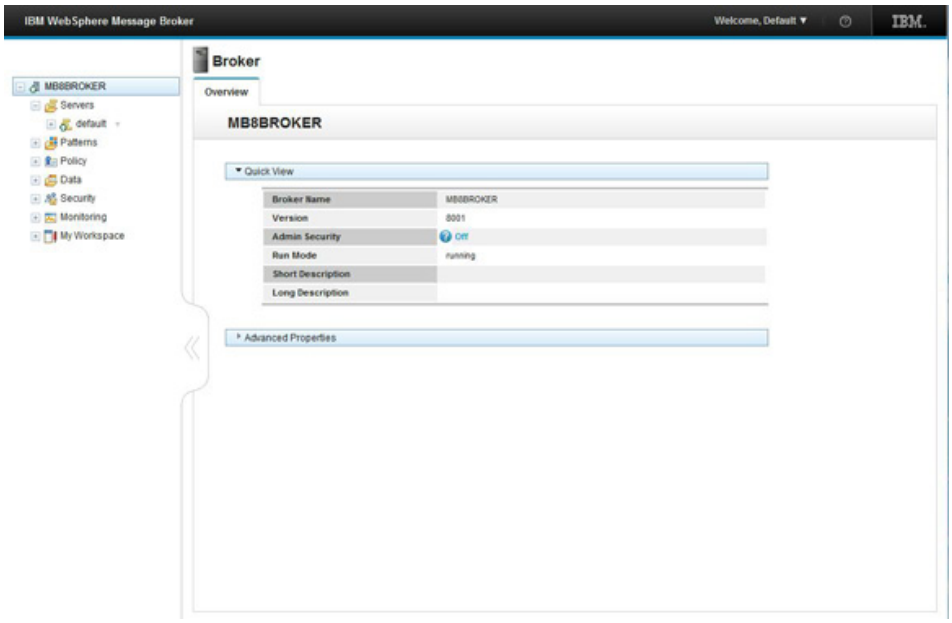
   This command produces a response similar to this example:
   ```
   server=''
   uuid='server'
   enabled='true'
   enableSSL='false'
   ```

5. To ensure that the changes take effect, restart the broker by running the following commands in the command console:

```
mqsistop MB8BROKER
mqsistart MB8BROKER
```

Connecting to the web UI

After a few seconds, the Web UI should be up and running. Using either Mozilla Firefox V3.6 or later, or Microsoft® Internet Explorer V8 or later, you can open the Web UI using the URL `http://serverAddress:4414/`. For example, if your broker is running on your local host, the URL would be `http://localhost:4414`:



Setting up the record and replay database

To store recorded messages, WebSphere Message Broker uses a database. Therefore you need to set a database with the necessary tables, and configure the broker to use them.

This tutorial uses IBM DB2. To define the necessary tables, the broker installation includes an SQL script , located at `{install path}/ddl/db2/DataCaptureSchema.sql`. You will need to customize the script to suit your recording needs and database naming standards. At the top of the file, you will find the database name MBRECORD, and you can optionally use a database schema:

```
. . .
CREATE DATABASE MBRECORD;
CONNECT TO MBRECORD;
-- If you want to use a different schema, edit and uncomment the following lines
--CREATE SCHEMA WMB;
--SET SCHEMA WMB;
. . .
```

Consider the size of messages that you want to record. If you are capturing large message payloads, you may need to increase the amount of space allocated from the default of 5 MB. You can change the size in bytes for the DATA field in the WMB_BINARY_DATA table:

```
. . .
CREATE TABLE WMB_BINARY_DATA
(
   WMB_MSGKEY    VARCHAR(100) NOT NULL,
   DATA_TYPE     INTEGER      NOT NULL,
   ENCODING      VARCHAR(50)  NOT NULL,
   DATA          CLOB(5242880)
 )
 . . .
```

In this case, leave the settings at their default values and create a database MBRECORD that can hold 5 MB. Then run the script using the command `db2 -tvf DataCaptureSchema.sql` from a db2 command shell. Check that the statements executed correctly before continuing.

The next task is to create an ODBC datasource for this database so that the broker can connect to it. This task is platform-dependent -- for more information, see Enabling ODBC connections to databases in the WebSphere Message Broker V8 information center.

Important: If you are using a 32-bit installation of WebSphere Message Broker on Microsoft® Windows® 64-bit, then make sure you are using the 32-bit version of the ODBC configuration panel, because the configurations are separate. For details, see the information center topic above. In this example, use the name MBRECORD for the ODBC definition to match the database name.

The database should now be configured and ready to use, and you can now connect the broker to the database. Since the broker is called MB8BROKER, use the following `mqsisetdbparms` command to store the database credentials:

```
mqsisetdbparms MB8BROKER -n odbc::MBRECORD -u {DBUSER} -p {DBPASSWORD}
```

After this command completes successfully, restart the broker to pick up the new credentials. Then use the `themqsicvp` command to verify that the database connection is correct:

```
mqsicvp MB8BROKER -n MBRECORD -u {DBUSER} -p {DBPASSWORD}
```

If successful, this command returns a list of supported operations against the database.

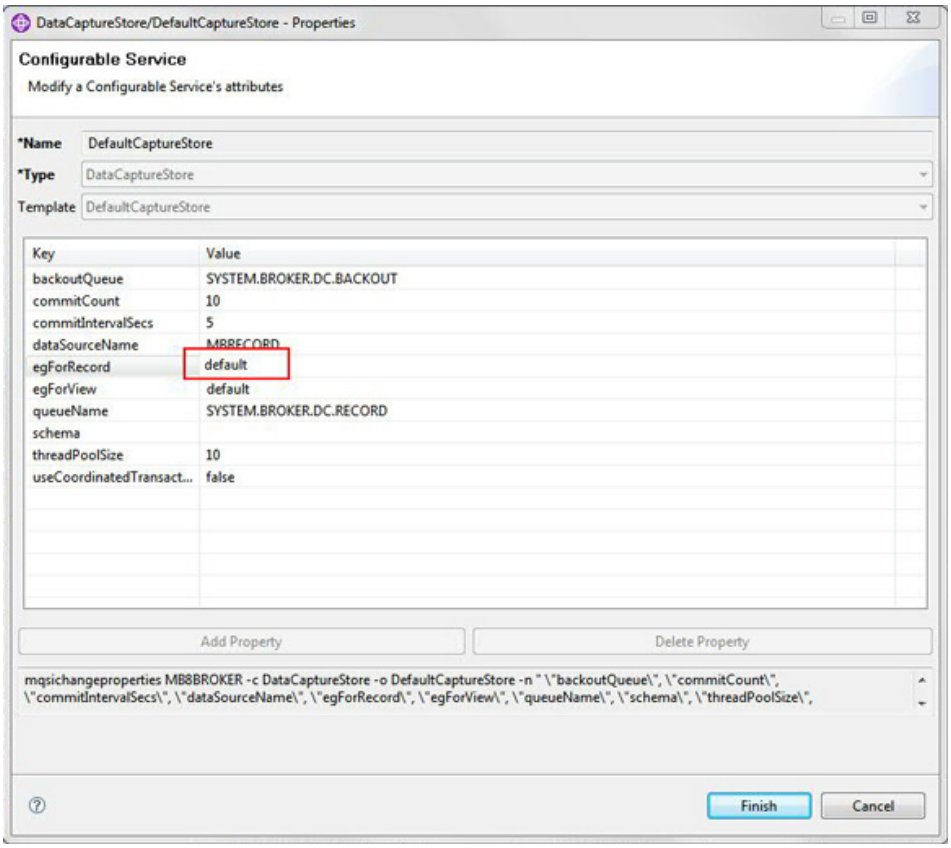
The final task here is to configure the database for record and replay by introducing the first of three new configurable services, the DataCaptureStore. This configurable service definition is important, since it represents access to the recorded messages for a number of purposes, as you will see later. Messages are recorded, viewed, and replayed using this DataCaptureStore. This DataCaptureStore configurable service is dynamic, meaning that you won't need to restart the broker for the changes to be picked up. The other two configurable services described below are also dynamic -- DataCaptureSource, and DataDestination.

You can define a DataCaptureStore either via the command line or in Message Broker Explorer. The image below uses the Explorer, but command-line equivalents are also shown. This example assumes that you have already created and started MB8BROKER. Note that it is the execution groups that will record, view, and replay the message data, and you need to specify which

execution groups have that task. You can have different execution groups (and even different brokers) perform these tasks to a single database.

1. Open up the Message Broker Explorer interface and navigate to the Configurable Services section in the left-hand tree view under MB8BROKER. Right click and select **New =>**

Configurable service:



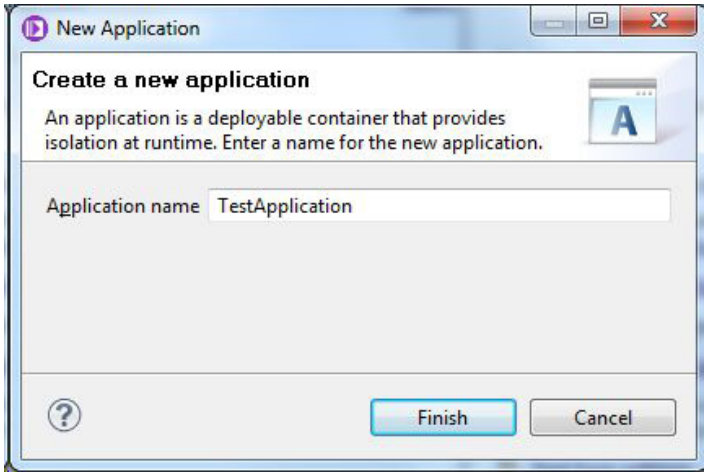
2. Change the configurable service type to be a DataCaptureStore. Initially, you will see values created based on the IBM-provided template definition, which you can then change.
3. For the broker to access the database, you just need to configure the dataSourceName to match the name of your ODBC definition above (MBRECORD). If you defined a database schema in the SQL, it will also need to be put in the schema property.
4. The other properties are used when recording messages. The first important (and mandatory) property to set is `egForRecord`, which denotes which execution group performs the recording. You can leave the remaining properties at their default values. When configured to record messages, the designated execution group creates a number of threads that wait for messages and place them in the database. Tuning this thread pool in order to get the maximum throughput for the message workload is the purpose of the other properties shown here.

Adding monitoring to your flows for record and replay

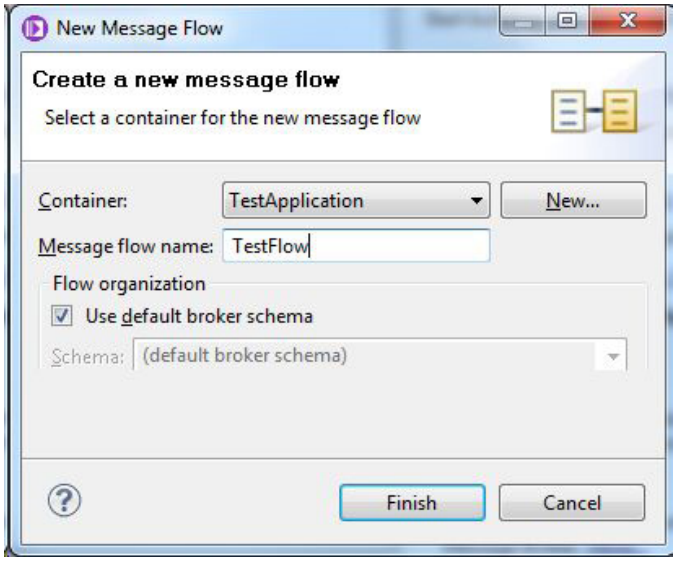
In this section, you create a simple application containing a flow, then add monitoring events to the flow, which are picked up by the default execution group and stored in the `datacapturestore` database that you just defined and connected to MB8BROKER.

You can download the mbtest files and BAR files at the bottom of the article.

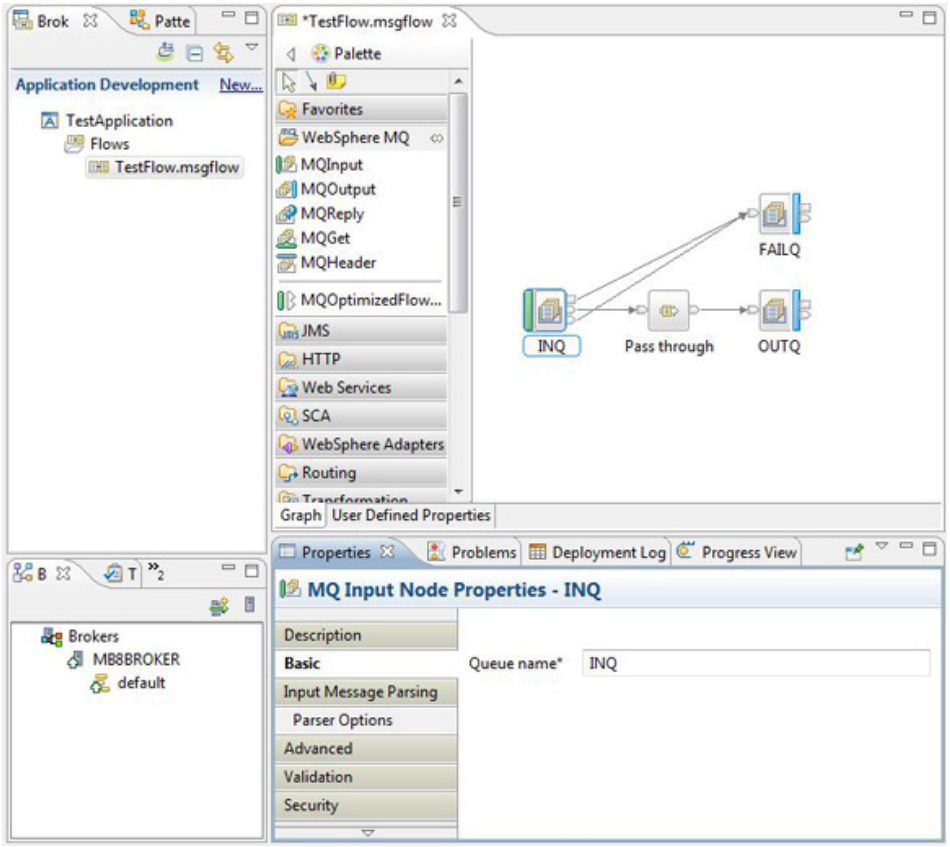
1. Create a new application using the Message Broker Toolkit:



2. Create your TestFlow flow:

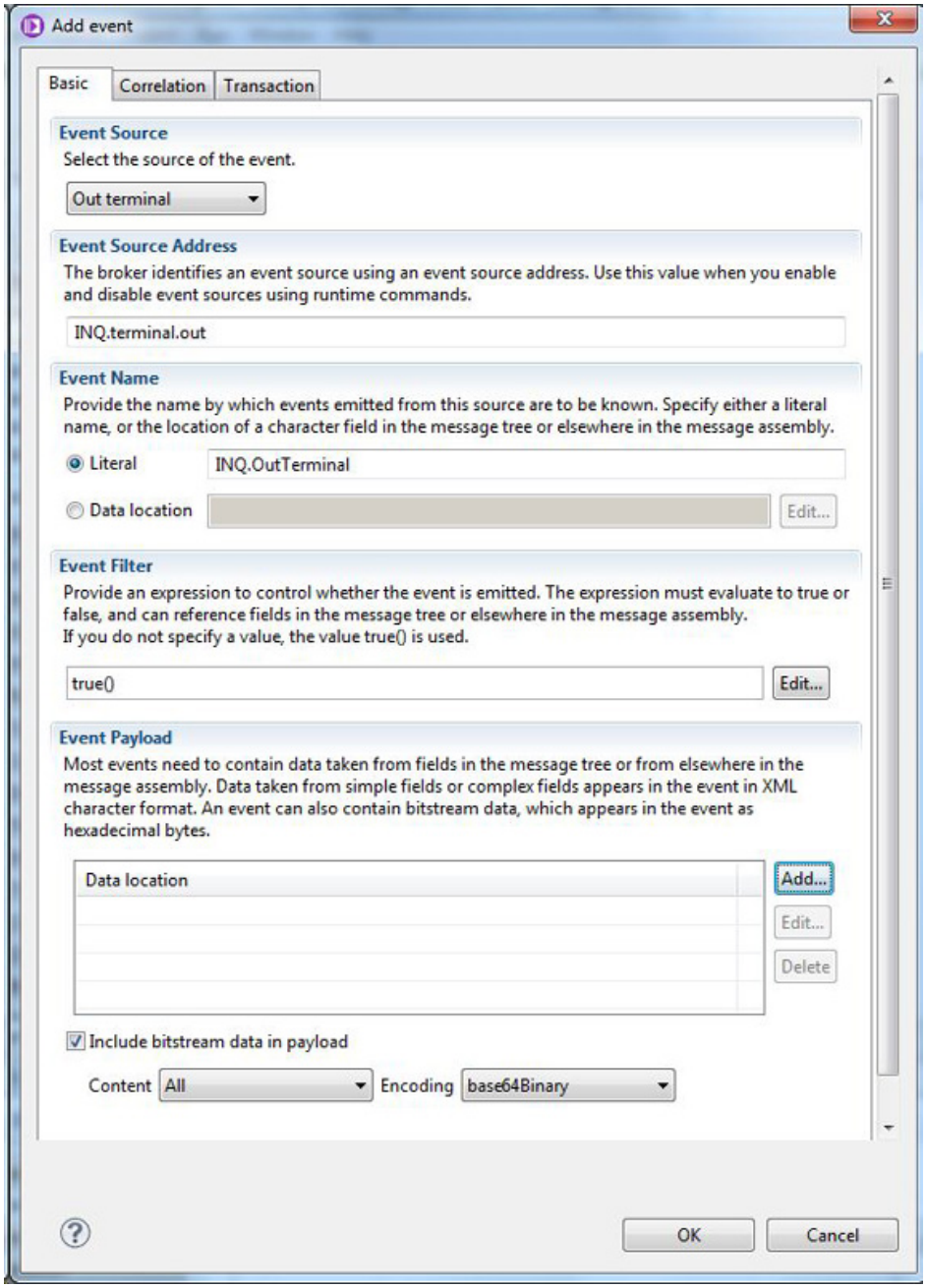


Here is a simple flow: MQInput Node (Queue Name=INQ) => PassThrough Node => MQOutput Node (Queue Name=OUTQ), with the Catch and Failure terminals connected to an MQOutput Node (Queue Name=FAILQ):



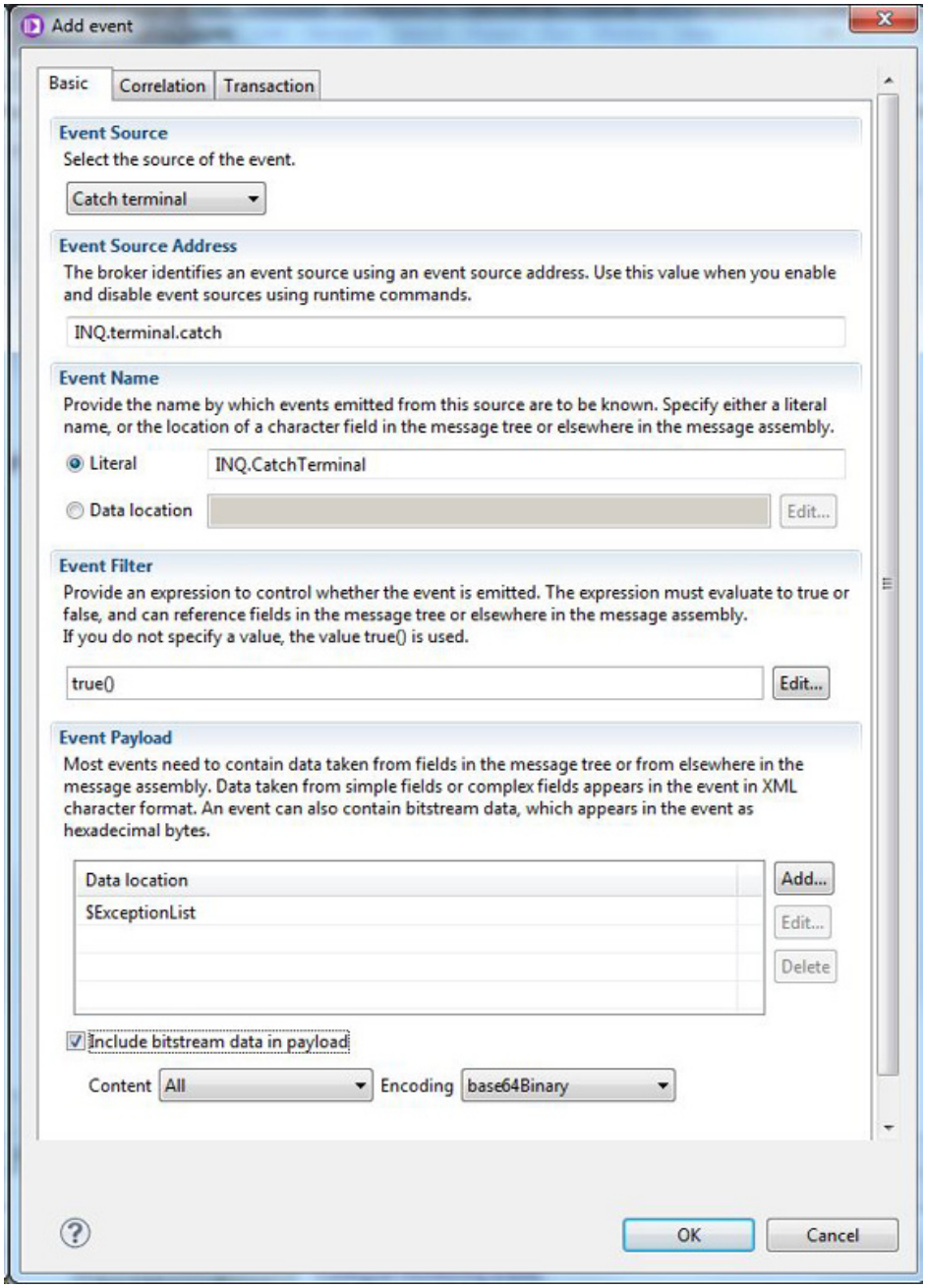
3. Now you have a simple flow and can add monitoring events to it. You want an audit event to fire whenever the message leaves the Out terminal of the INQ, and an exception message to fire whenever the message goes down the Catch terminal to the FAILQ. Click on the **INQ** node to select the Monitoring tab in the Properties and then click **Add**.
4. Select the Event Source as **Out Terminal** and tick **Include bitstream in payload** to include the bitstream in the event message, so that the broker emits an event every time a message goes out of this terminal and attaches the bitstream, which the recording

broker can listen for and then place into the broker's record and replay database:

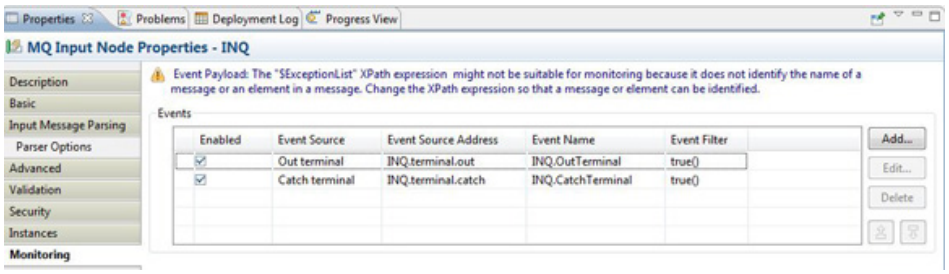


5. For the failure case you want to add another monitoring event, but this time select the source as **Catch Terminal**, add in the data location as **$ExceptionList**, and include the bitstream as before. Any exception information will also be sent to the record and replay database and can be viewed there. At present the broker only records bitstreams

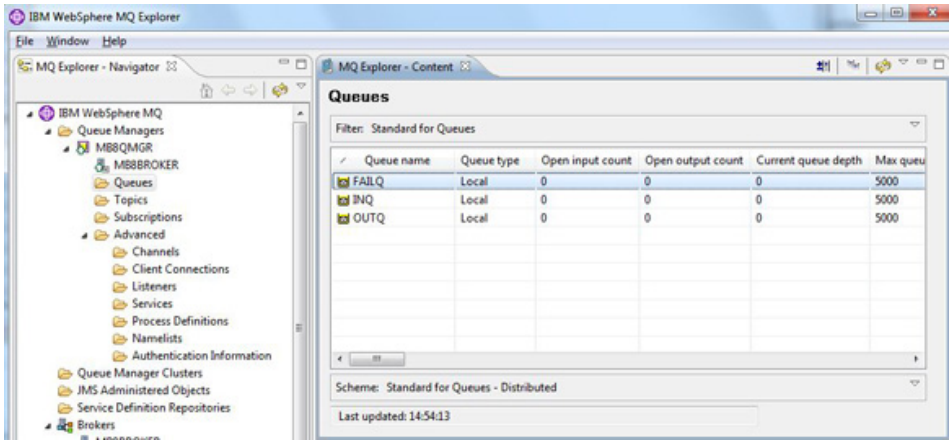
and ExceptionLists, and any other environment properties set there are ignored:



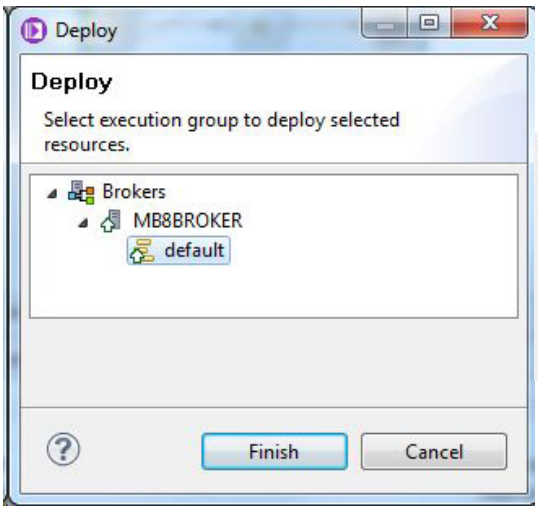
6. Check that your monitoring has been correctly set up by looking at your flow overview. You should have two monitoring events created:

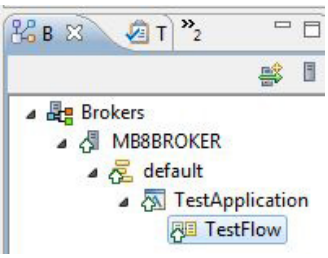


7. Before deploying this flow to the default broker, make sure that the queues are created and available. Using Message Broker Explorer, create the queues: INQ, OUTQ, and FAILQ:



8. You are now ready to deploy the TestApplication. Right-click on the flow, select **Deploy**, and then select the **default** execution group. After a few seconds you should see your deployed application in the broker:





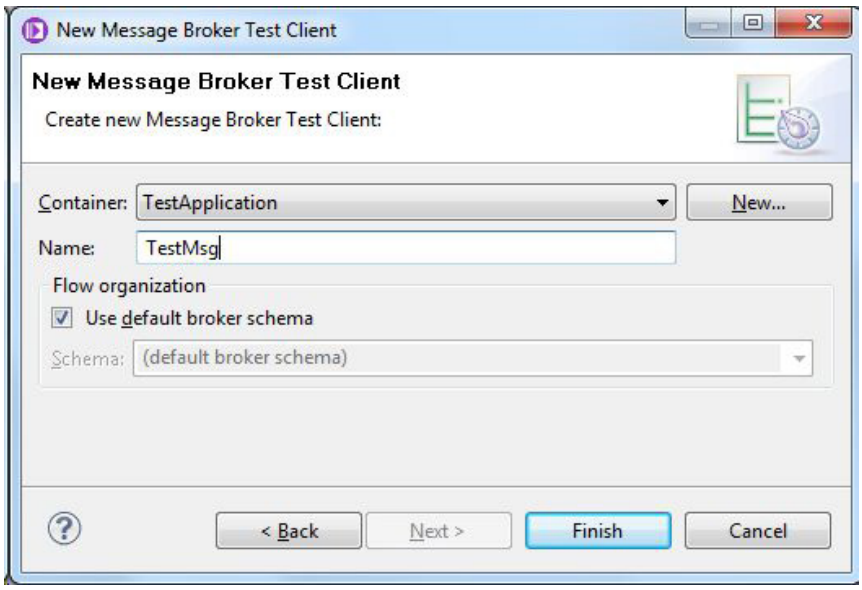
9. Every time you deploy, flow monitoring is turned off by default, so after you have deployed the application, make sure you turn on flow monitoring using the command:

```
mqsichangeflowmonitoring MB8BROKER -e default -k TestApplication
    -f TestFlow -c active
```

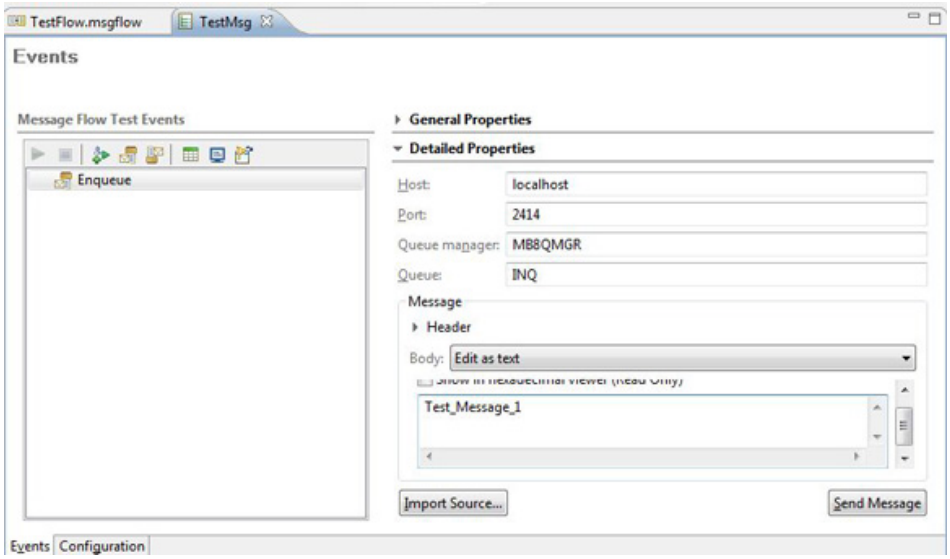
10. Run the command below to verify that monitoring is enabled for your flow:

```
mqsireportflowmonitoring MB8BROKER -e default -k TestApp
```

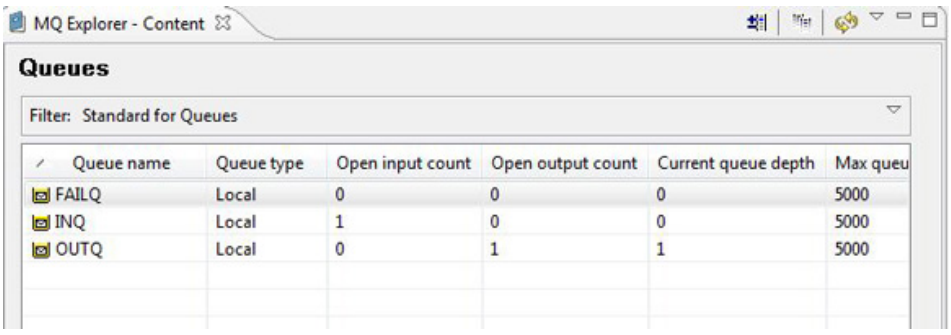
11. Now that you have the application deployed, you need to test that the flow is working by sending some test messages using the Message Broker Toolkit test client. Select **File => New Message Broker Test Client** and create a message named TestMsg:



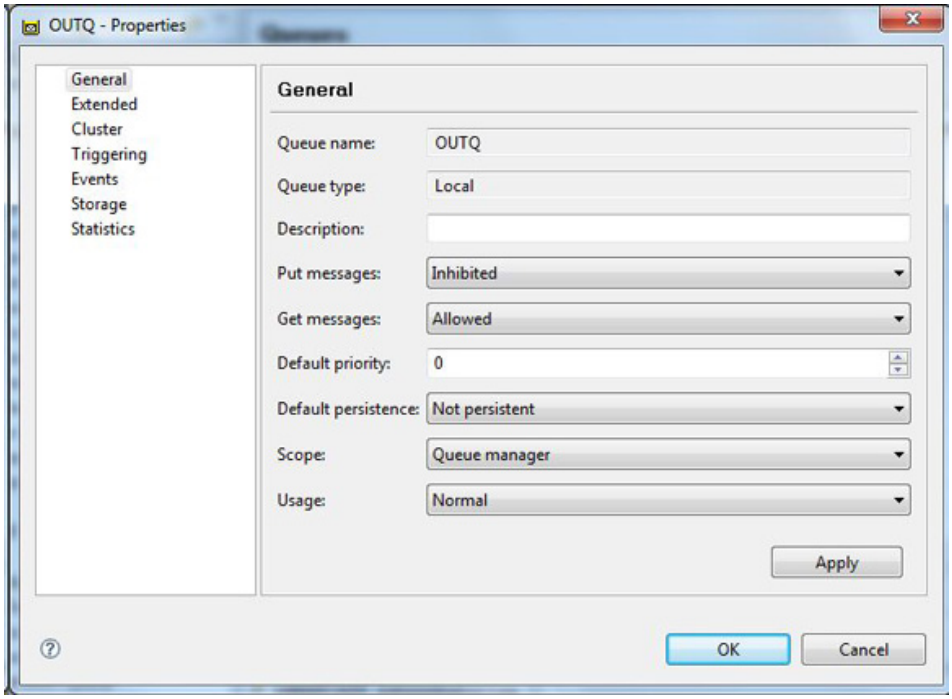
12. You want to enqueue a test message to the MQ queue INQ on the queue manager MB8QMGR, and make sure that it gets to OUTQ. Enter some test message data so that you can view this payload later in the Web Administration:



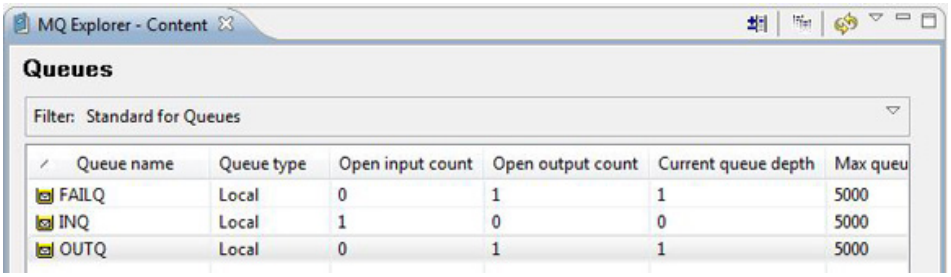
13. After the message has been sent, you can check that it has been sent to OUTQ by looking in Message Broker Explorer and inspecting the queues. You will see a queue depth of 1 on OUTQ:



14. Next, test the behaviour of FAILQ. Change OUTQ so that it is put-inhibited by using Message Broker Explorer:



15. Now when you send a message, the flow will not be able to put the message to OUTQ, and the message will be caught and routed by the INQ MQInput node and routed to FAILQ. On its way there it will fire a monitoring event capturing the ExceptionList and payload of the message for the Record and Replay database. Here you can see that the second test message has been routed to the FAILQ as expected:

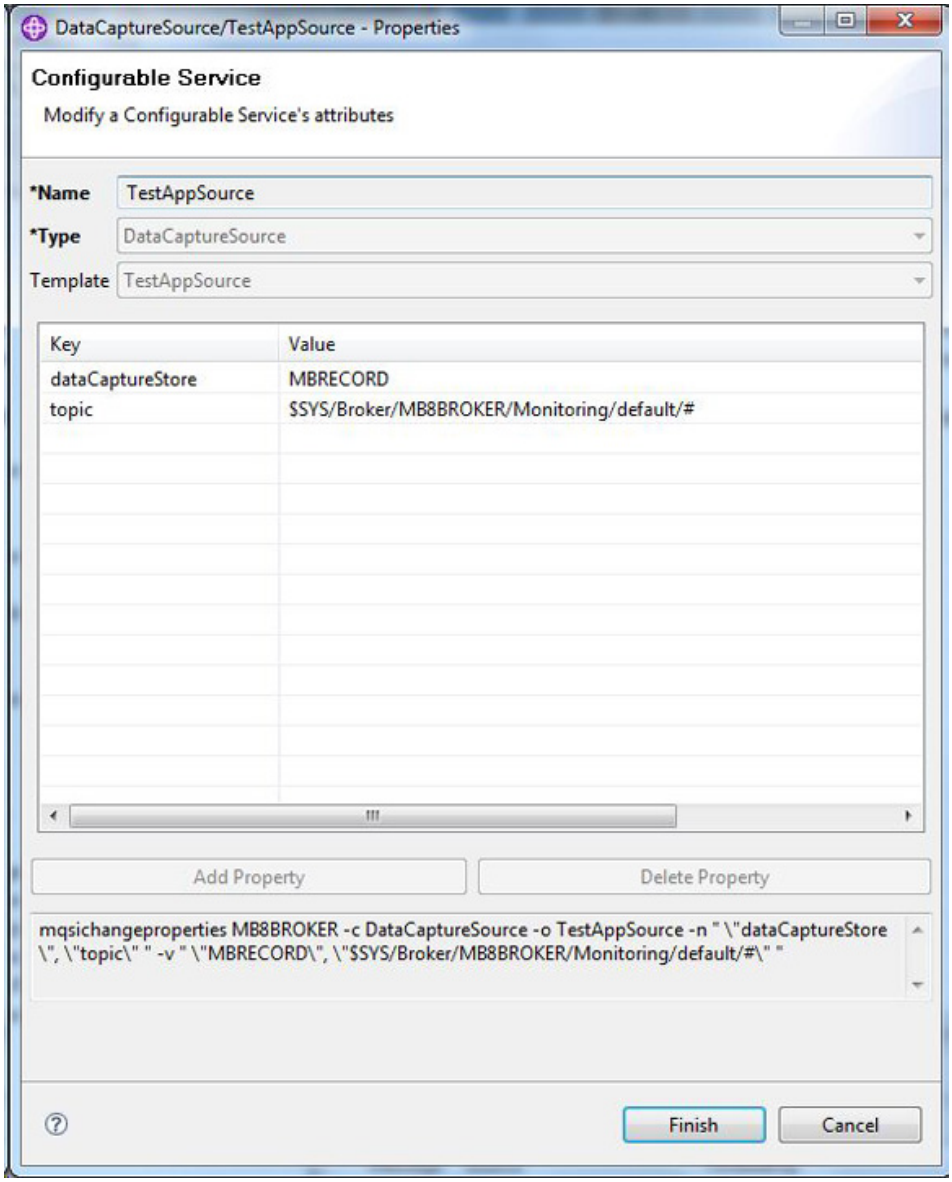


In the next section, you set up the broker to listen for the monitoring events you have set up here, so it can route them to the Record and Replay database.

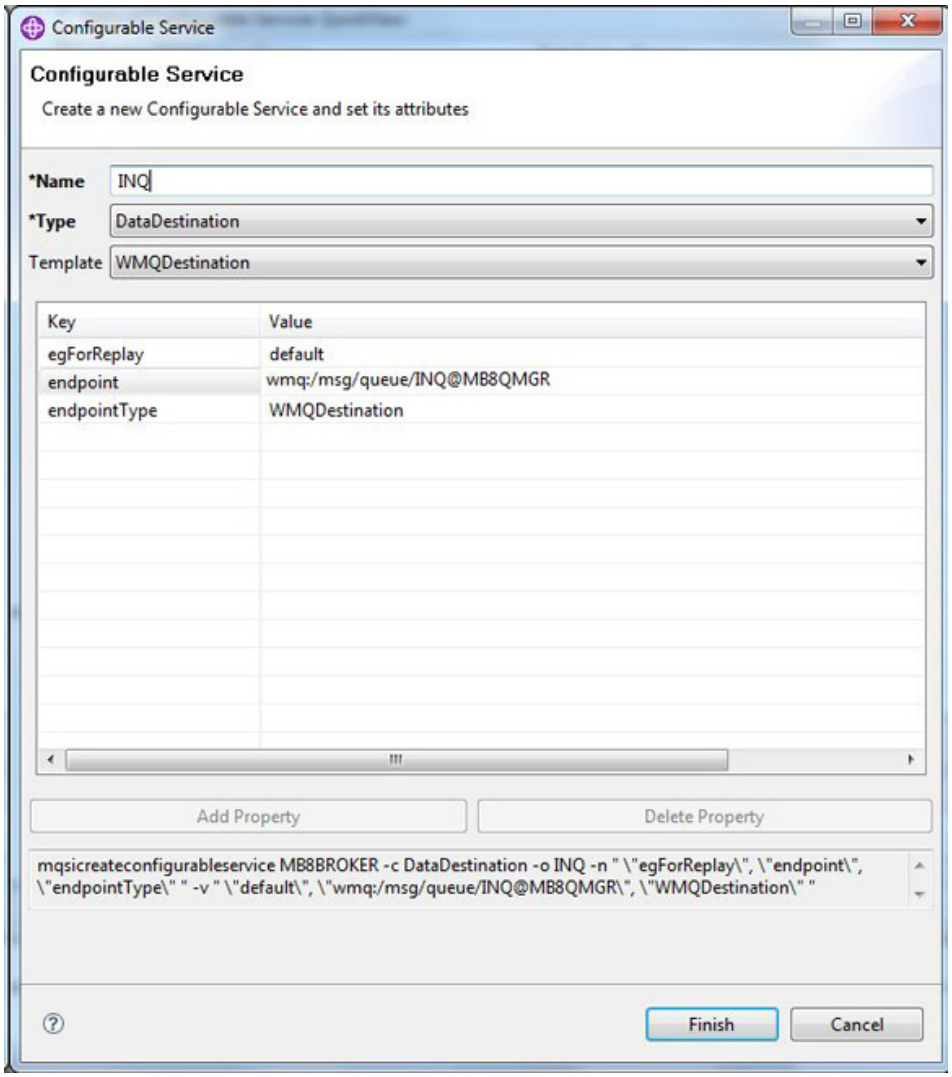
Setting up the DataCaptureSource on your broker

1. You now need to configure the broker to listen for monitoring events being emitted from the flow that you have deployed. The test message you sent has been published, but no one was subscribed to the published topic, so the messages were not received anywhere. So you need to set up a DataCaptureSource to specify a topic to listen to messages from, and a

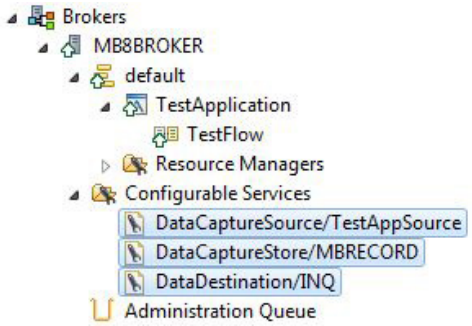
Record and Replay database connection (previously defined as a DataCaptureStore) to send the captured messages to:



2. While you are using Message Broker Explorer, you can also set up your INQ DataDestination, which will then become a queue to which you can replay messages through the Web Administration:



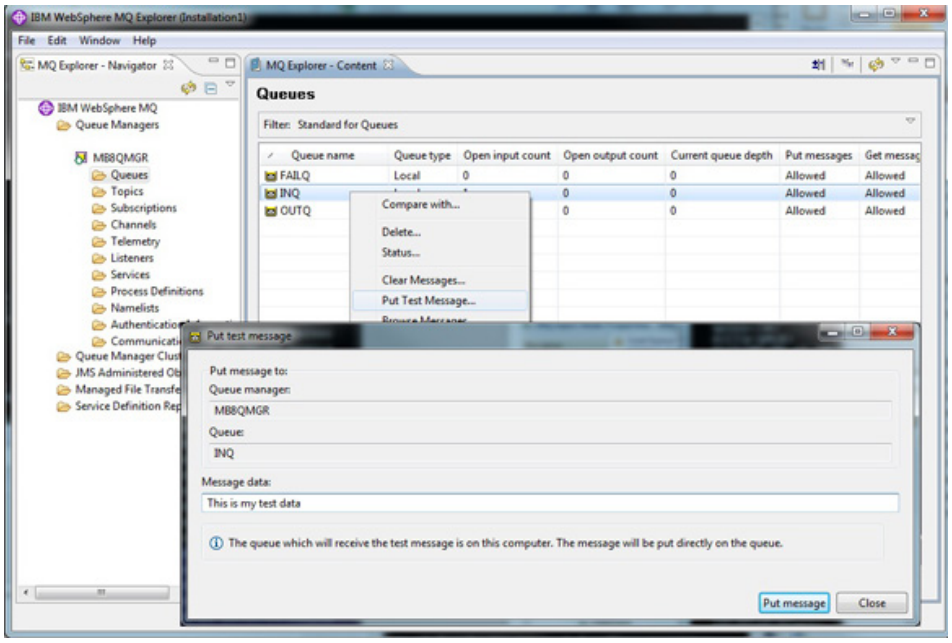
3. Here are the three deployed configurable services in Message Broker Explorer to configure Record and Replay:



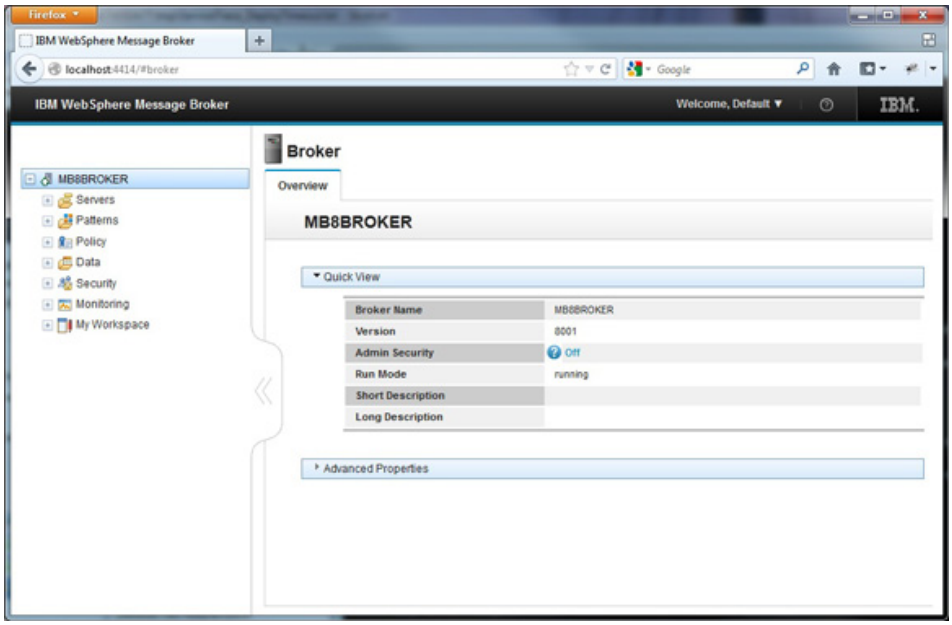
Now you are ready to send some messages through and view them within the Web Administration.

Viewing and replaying messages

1. Before looking at the web interface, you need to put a message on the INQ queue so that you can see that it has been recorded into the database. The simplest way to do this is to use MQ Explorer and select **INQ** from the list of queues. Right-click the queue and click **Put test message**. At the prompt, enter some content data for a message:

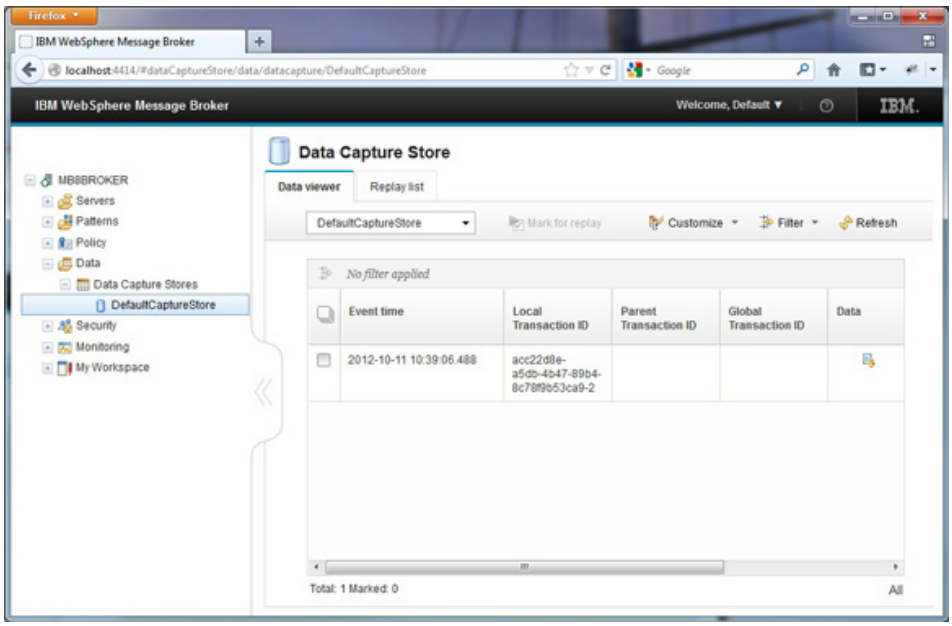


2. You have now recorded your first message to the database. If it is not already open, start up the Web Administration Interface by going to `http://localhost:4414`:



3. In the left-hand navigator, open the **Data** section and select **DefaultCaptureStore**. If it is not visible, check that the egForView property is defined as a valid execution

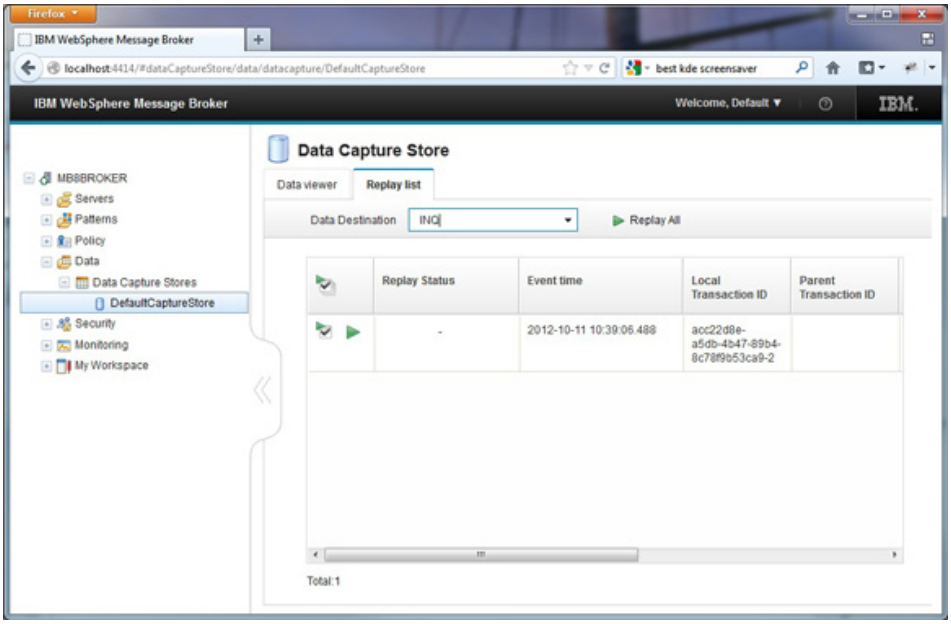
group in the configurable service. You should see that a message has been recorded:



If you do not see a message, verify that the mqsichangeflowmonitoring command has been run, that the DataCaptureSource and DataCaptureStore definitions are correct, and that the database is defined and available in ODBC.

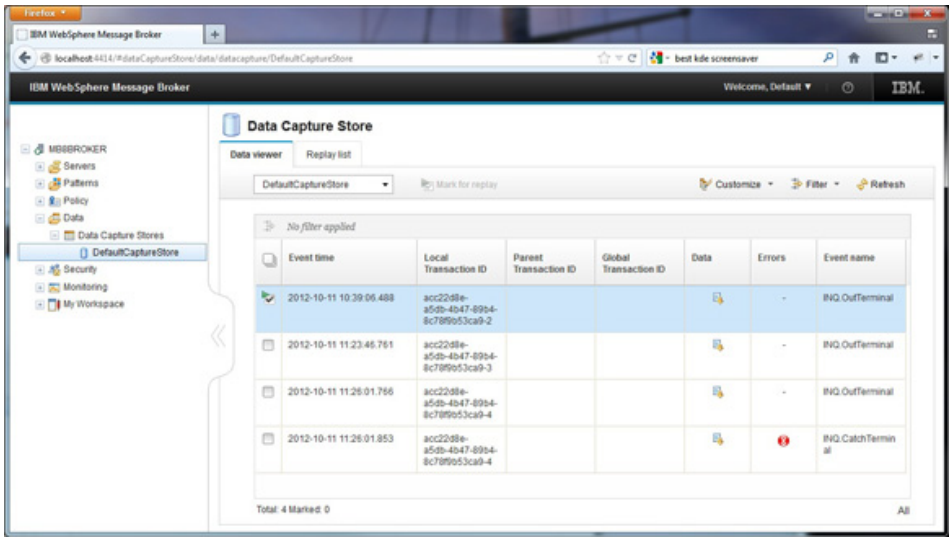
4. You can customize your data viewer by choosing columns, changing the column names and order by clicking **Customize** to customize the view on a per-DataCaptureStore basis, regardless of user.
5. You are now ready to replay a message to a destination, which was defined using the INQ DataDestination configurable service above. When a message is replayed, it will be sent to the defined endpoint (the queue called INQ). Since this queue is the input for the flow, the replayed message will be re-recorded and become visible in the Data Viewer. To replay the message, select the check box to the left of the messages you want to replay and then click **Mark for replay**, which takes you to the second tab, where you can see your chosen messages and select the destination to play them to.

6. Click the green arrow next to a message to replay one, or select **Replay all** at the top of the screen:



7. Having successfully replayed a message, change from the **Replay list** tab to the **Data viewer** tab and click **Refresh** on the viewer. You will now see another message, which is the replayed message having been re-recorded by the flow.
8. Replay the message again, but this time, make the target flow (TestFlow) fail the message and create a message on the catch terminal by put-inhibiting the flow's output queue. You can either use MQ Explorer as described above, or run the following command inside runmqsc:

```
ALTER QLOCAL(OUTQ) PUT(DISABLED)
```



9. You now have four messages in the Data Viewer. Message 1 is the original recorded message, which you replayed to create Message 2. This message was injected into the original flow, emitted from the out terminal, and recorded as Message 3. But it then failed to be put to OUTQ and therefore went to the catch terminal as Message 4. Now that you have multiple messages in the Data Viewer, you can experiment with message filtering, and

look at the captured message payload and exception information. When you have finished, remember to put-enable the OUTQ.

Conclusion

This tutorial showed you how to configure WebSphere Message Broker to record messages from a message flow to a database, and to replay those messages to a WebSphere MQ queue. It also showed you how to configure the Web Administration Interface, which lets you view deployed artifacts and configurable services, and access the Data Viewer.

As a next step, you can add security to the Web Administration Interface by enabling Administration Security, adding web user accounts, and granting them levels of authority. For more information, see the WebSphere Message Broker V8 information center.

Downloadable resources

Description	Name	Size
Code sample[1]	download.zip	6 KB

Note

1. download.zip contains two files: TestApplicationproject.generated.bar and TestMsg.mbtest

Related topic

- **WebSphere Message Broker resources**
  - WebSphere Message Broker V8 documentation in IBM Knowledge Center
  - WebSphere Message Broker documentation library