

**Supported Version:** Oracle BAM 10.1.3

**Objectives:**

Define a plan, and collect data from a message source. Parse collected data, manipulate data, and store in data objects. JMS Topic events are used as incoming message source.

**Prerequisites:**

1. Basic knowledge of BAM and installed BAM software.
2. Previous tutorial steps are completed.

**Notes:**

1. In this section we use the OC4J container and push data into the jms bus using a small java program. This java program pushes data “exactly” as intended by the BPEL OrderBooking process.
2. Enterprise Link Plans are used to collect data from JMS bus, and parse the data and populate the BAM ADC data objects.
3. Details of pushing “actual” BPEL events from OrderBooking BPEL process is given in the end of this lab, as optional exercise.
4. Many other alternate mechanisms are available to push data into BAM. These include direct BAM\_WSDL webservice calls, BPEL BAM sensors, ESB to BAM service, Oracle AQ/OJMS to BAM, Database polling, etc. These design steps are given in other documents.
5. A sample of pushing data into ADC from a J2EE webservice program is provided in this document, for illustration.

**Jndi properties file:**

Change Dir to *C:\OracleBAM\BAM\j2rel.4.1\_01\lib* and verify the jndi.properties file exists (create if needed, this file is also provided with some sample directory). The contents of this file should be:

```
java.naming.security.principal=oc4jadmin  
java.naming.security.credentials=welcome1
```

**The OC4J container (to push data into BAM):**

After the oc4j container is installed, start the oc4j container. (give default password as ‘welcome1’) *cd C:\XXX\OracleBAM\10.1.3\_Folder\Samples\SimulateJMSEvents\1.StartOC4J.bat* (do not close this window)

**Collecting RAW Data in BAM:**

Collecting :raw: data from the incoming JMS source is given in previous tutorial documents. After examining the raw data, steps are taken to parse the data and collect only fields that are required, and populate (insert, update or delete) data in BAM ADC.

**Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC.**  
(Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

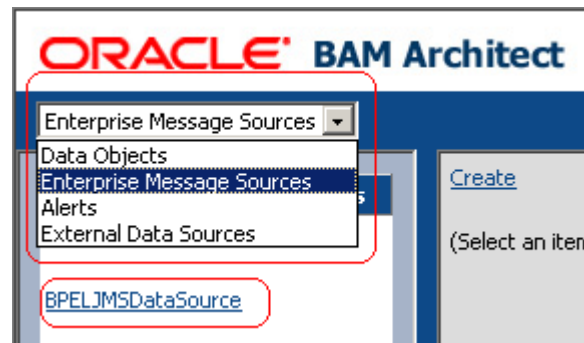
---

**Extending the message source (parse messages and mapping XML nodes to columns)**

The following steps will enhance the plan to “parse” the collected data using XSLT, and store it in the data objects (ADC)

Open Oracle BAM Architect. Select Enterprise Message Sources from the drop-down list.

Click on “BPELJMSDataSource.”



Click the “Edit” link to edit this data source.

In the Formatting tab select “XML: Column values are contained in tags”.

View | **Edit** | Copy | Delete | Create

Name:

Type: Oracle (AS JMS and OJMS)

Initial Context Factory:

JNDI Service Provider URL:

TopicConnectionFactory Name:

Topic Name:

JMS Message Type:

Durable Subscriber Name (Optional):

Message Selector (Optional):

Client ID (Optional):

Name	Dataflow name	Type	Max size	Formatting	
rawData	rawData	String	4000	No Formatting	<a href="#">Add</a>
					<a href="#">Remove</a>

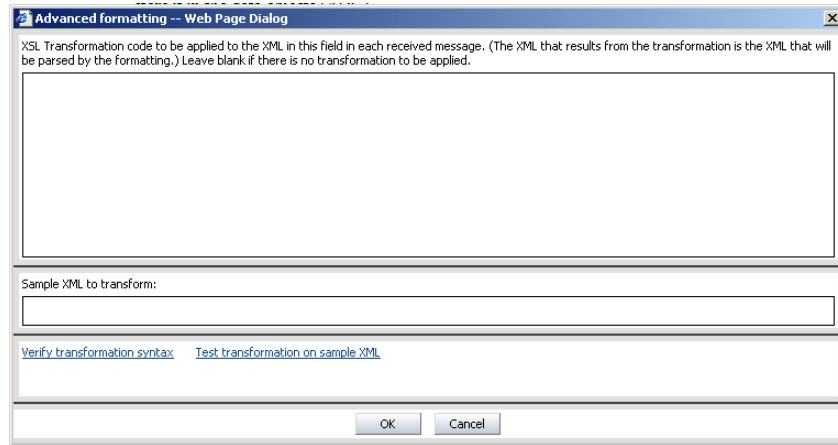
Save Cancel

No formatting  
XML: Column values are contained in tags  
XML: Column values are contained in attributes

## Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC. (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

---

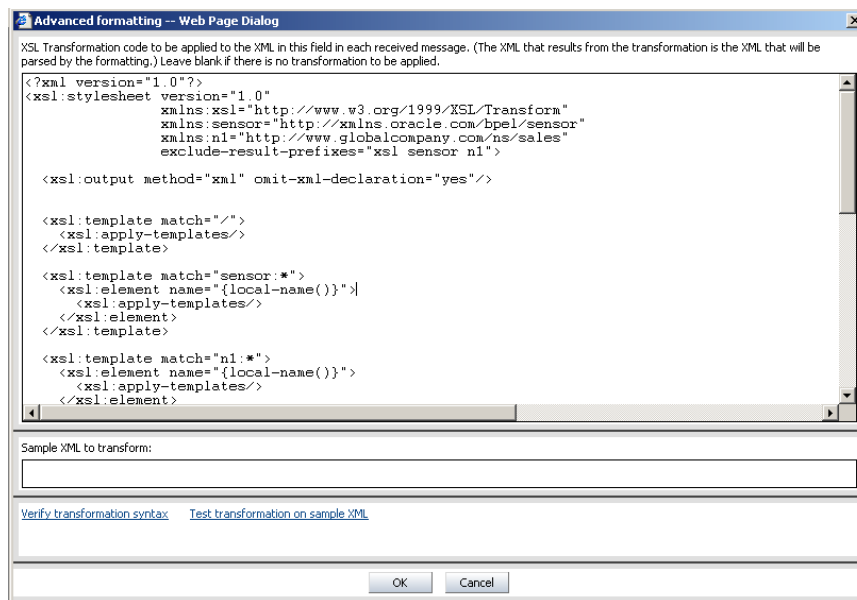
Click on “advanced formation option”. This will open the “Advanced Formatting screen”.



Now open the file in notepad.

*“C:\XXX\OracleBAM\10.1.3\_Folder\Samples\BPEL-JMS-BAM\OrderBookingDemo\LabFiles\StripNameSpacesFlattenOrderBooking.xsl”*

Copy and paste the contents of this file into the XSL window as shown below, and click OK.



**Note:** The above XSLT file translates the incoming xml data structure (including nested nodes, hierarchical xml data) and flattens it so that all the “contents” are available at the first level of the xml structure.

Edit the entries in the formatting tab as follows.

## Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC. (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

**Path:** /actionData (case sensitive)

Now click on the **Add** link (shown below) to specify TagName and other details for each field value. (case sensitive)

Name	Dataflow name	Type	Max size	Formatting	<a href="#">Add</a>																																				
rawData	rawData	String	4000	XML: Column values are contained in tags <a href="#">Advanced formatting options</a> Path: /actionData <table border="1"><thead><tr><th>Tag/Attr name</th><th>Dataflow name</th><th>Max size</th><th><a href="#">Add</a></th></tr></thead><tbody><tr><td>instanceId</td><td>instanceId</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>processName</td><td>processName</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>OrderTimeStamp</td><td>OrderTimeStamp</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>SupplierName</td><td>SupplierName</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>SupplierPrice1</td><td>SupplierPrice1</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>OrderPrice1</td><td>OrderPrice1</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>OrderStatus</td><td>OrderStatus</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>updaterName</td><td>updaterName</td><td>50</td><td><a href="#">Remove</a></td></tr></tbody></table>	Tag/Attr name	Dataflow name	Max size	<a href="#">Add</a>	instanceId	instanceId	50	<a href="#">Remove</a>	processName	processName	50	<a href="#">Remove</a>	OrderTimeStamp	OrderTimeStamp	50	<a href="#">Remove</a>	SupplierName	SupplierName	50	<a href="#">Remove</a>	SupplierPrice1	SupplierPrice1	50	<a href="#">Remove</a>	OrderPrice1	OrderPrice1	50	<a href="#">Remove</a>	OrderStatus	OrderStatus	50	<a href="#">Remove</a>	updaterName	updaterName	50	<a href="#">Remove</a>	<a href="#">Remove</a>
Tag/Attr name	Dataflow name	Max size	<a href="#">Add</a>																																						
instanceId	instanceId	50	<a href="#">Remove</a>																																						
processName	processName	50	<a href="#">Remove</a>																																						
OrderTimeStamp	OrderTimeStamp	50	<a href="#">Remove</a>																																						
SupplierName	SupplierName	50	<a href="#">Remove</a>																																						
SupplierPrice1	SupplierPrice1	50	<a href="#">Remove</a>																																						
OrderPrice1	OrderPrice1	50	<a href="#">Remove</a>																																						
OrderStatus	OrderStatus	50	<a href="#">Remove</a>																																						
updaterName	updaterName	50	<a href="#">Remove</a>																																						

Tag/Attr Name	Dataflow Name	Max size	
instanceId	instanceId	50	Case sensitive
processName	processName	50	Case sensitive
OrderTimeStamp	OrderTimeStamp	50	Case sensitive
SupplierName	SupplierName	50	Case sensitive
SupplierPrice1	SupplierPrice1	50	Case sensitive
OrderPrice1	OrderPrice1	50	Case sensitive
OrderStatus	OrderStatus	50	Case sensitive
updaterName	updaterName	50	Case sensitive

Verify the entries to make sure they appear exactly as shown below. Verify that the names match in case.

Name	Dataflow name	Type	Max size	Formatting	<a href="#">Add</a>																																				
rawData	rawData	String	4000	XML: Column values are contained in tags <a href="#">Advanced formatting options</a> Path: /actionData <table border="1"><thead><tr><th>Tag/Attr name</th><th>Dataflow name</th><th>Max size</th><th><a href="#">Add</a></th></tr></thead><tbody><tr><td>instanceId</td><td>instanceId</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>processName</td><td>processName</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>OrderTimeStamp</td><td>OrderTimeStamp</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>SupplierName</td><td>SupplierName</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>SupplierPrice1</td><td>SupplierPrice1</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>OrderStatus</td><td>OrderStatus</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>OrderPrice1</td><td>OrderPrice1</td><td>50</td><td><a href="#">Remove</a></td></tr><tr><td>updaterName</td><td>updaterName</td><td>50</td><td><a href="#">Remove</a></td></tr></tbody></table>	Tag/Attr name	Dataflow name	Max size	<a href="#">Add</a>	instanceId	instanceId	50	<a href="#">Remove</a>	processName	processName	50	<a href="#">Remove</a>	OrderTimeStamp	OrderTimeStamp	50	<a href="#">Remove</a>	SupplierName	SupplierName	50	<a href="#">Remove</a>	SupplierPrice1	SupplierPrice1	50	<a href="#">Remove</a>	OrderStatus	OrderStatus	50	<a href="#">Remove</a>	OrderPrice1	OrderPrice1	50	<a href="#">Remove</a>	updaterName	updaterName	50	<a href="#">Remove</a>	<a href="#">Remove</a>
Tag/Attr name	Dataflow name	Max size	<a href="#">Add</a>																																						
instanceId	instanceId	50	<a href="#">Remove</a>																																						
processName	processName	50	<a href="#">Remove</a>																																						
OrderTimeStamp	OrderTimeStamp	50	<a href="#">Remove</a>																																						
SupplierName	SupplierName	50	<a href="#">Remove</a>																																						
SupplierPrice1	SupplierPrice1	50	<a href="#">Remove</a>																																						
OrderStatus	OrderStatus	50	<a href="#">Remove</a>																																						
OrderPrice1	OrderPrice1	50	<a href="#">Remove</a>																																						
updaterName	updaterName	50	<a href="#">Remove</a>																																						

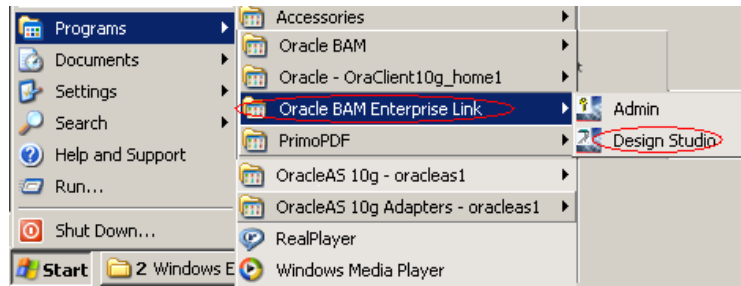
Click “Save” to **Save** the changes.

Click “Continue” and close the Architect Window.

### **Testing the “Plan”** (message processing)

Open the DesignStudio (from windows start programs menu), and login as “BAM”.

## Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC. (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)



”Drag and Drop” the splitter bar to the center of the window.

Click on the “Grid Panel” and “drag and drop” the “myBPELOrderBookingPlan” to the bottom panel. (See picture below with step numbers)

**Note:** DO NOT click or double click the Plan name – whenever you are opening a plan always drag and drop the plan.

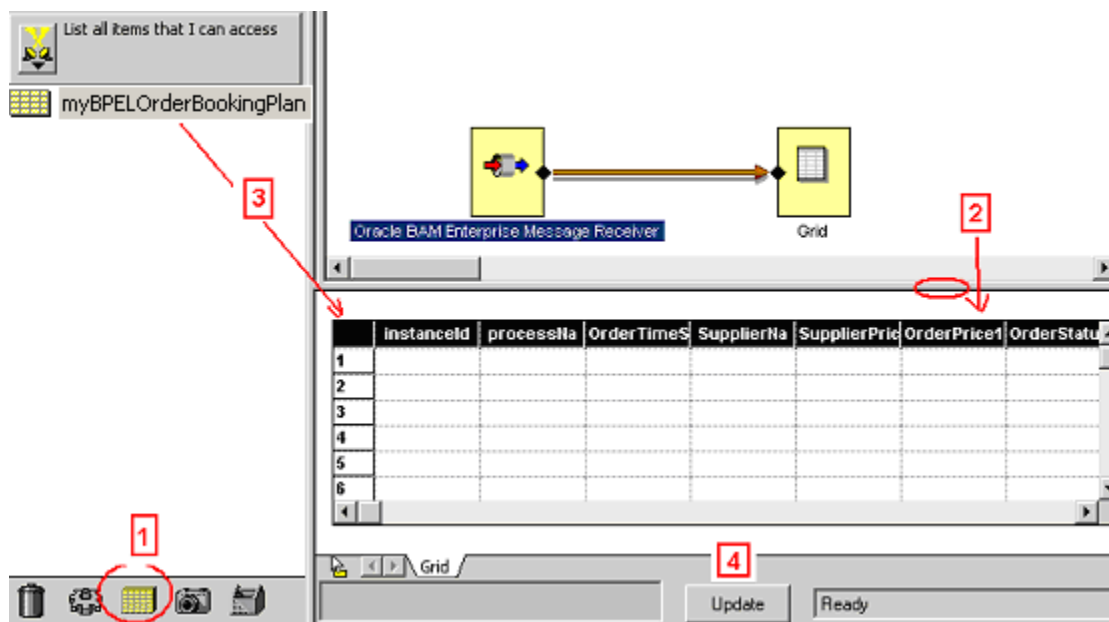
This will open the plan and the data sink grid as shown.

Start pushing data using the small java program given with the lab tutorial.

On the dos prompt, start the given java program to push data to BAM.

*cd C:\XXX\OracleBAM\10.1.3\_Folder\Samples\SimulateJMSEvents  
2.SimulateEvents (Note this batch program takes command line arguments)  
Use the command line arguments to push more data if required.*

Click on “update” button for the plan to connect and collect the messages from the source. (*jms topic bus*)



## Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC. (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

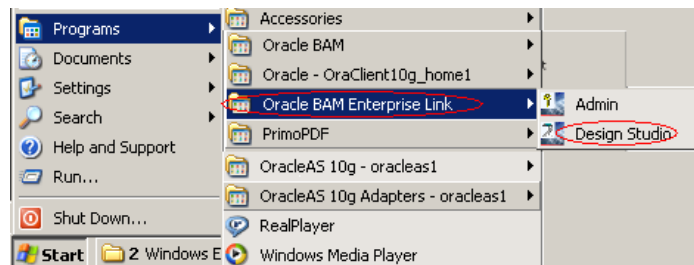
---

Note that the “message source” is now collecting messages, parsing the XSLT and presenting the nodes as data object to the “plan”. The above results validate the XSLT logic and data parsing. Note that all the data columns are in “string” format at this stage. We have not yet defined, designed or populated the data objects (in ADC)

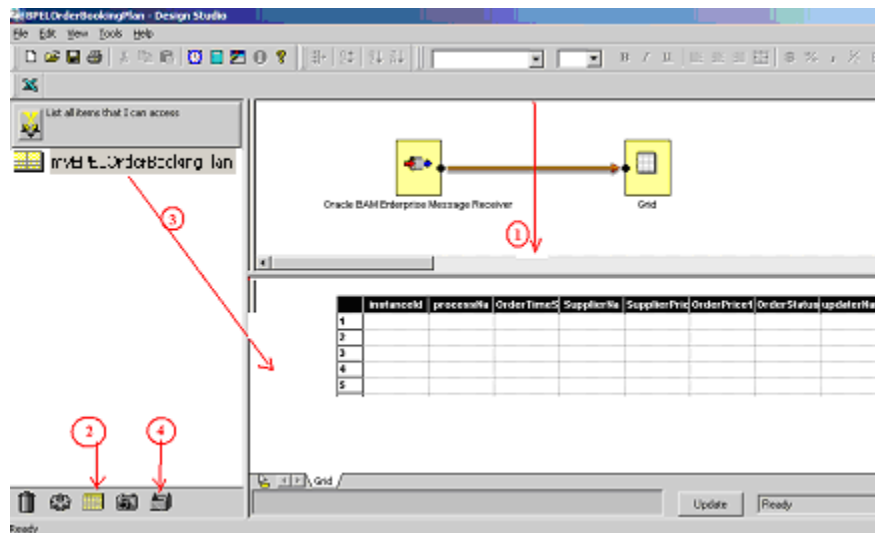
Close the Plan (save & close).

### Extending the plan to parse the collected data and store in ADC

Start the DesignStudio and Log In as default (BAM) with no password.



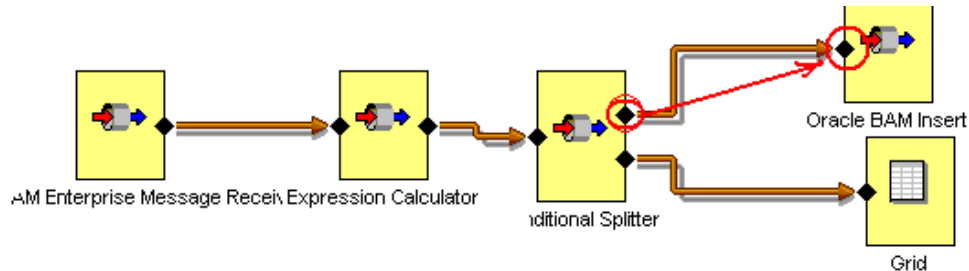
Open the myBPELOrderBookingPlan for editing. (See picture below with step numbers)  
First drag the splitter bar half way down the screen. Next click on the tool bar for plans (grid). Next drag and drop the plan into the lower window pane. (Do not double click on the Plan name to open it). Next click on tool bin palette.



Drag and drop the “**expression calculator**” (found under the Data Manipulation branch) in between the Receiver and Grid (on the line). Drag and drop the “**conditional splitter**” (found under the Data Flow Control branch) between the expression calculator and the grid.

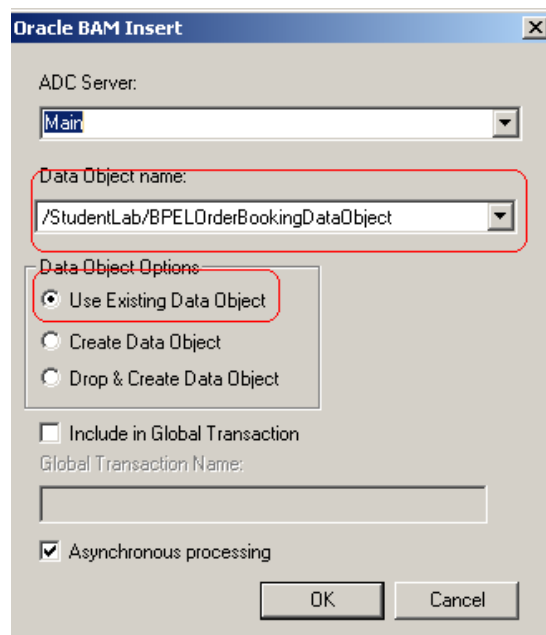
## Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC. (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

Drag and drop the “**Oracle BAM Insert Sink**” (found under Non-display Sinks) into the diagram and connect the splitter with the Oracle BAM Insert icon.(NOTE : The top link should connect to Oracle BAM Insert and the Bottom one to Grid. You have to delete the existing link by right clicking on a link and selecting **delete flow** and recreate these links)



Double click on the Oracle BAM Insert, and select the “**Use existing Data Object**” option.

In the data Object Name drop down list, select the data object: “/StudentLab/BPELOrderBookingDataObject”.



Click OK and close the Oracle BAM Insert Dialog

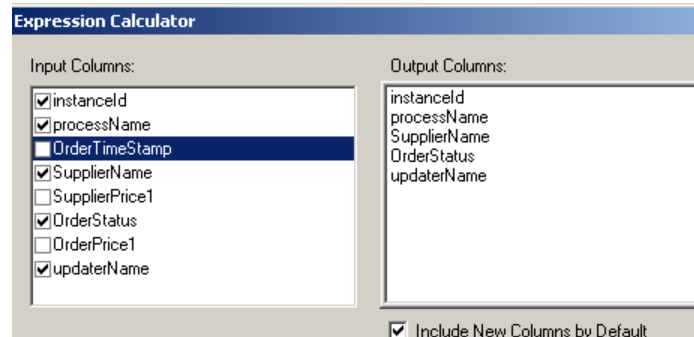
**Note:** Grid is used to collect the data from Receiver and display it online, while Insert block actually collects the data and puts it persistently (insert) into the data object.

## Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC. (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

### Data Conversion

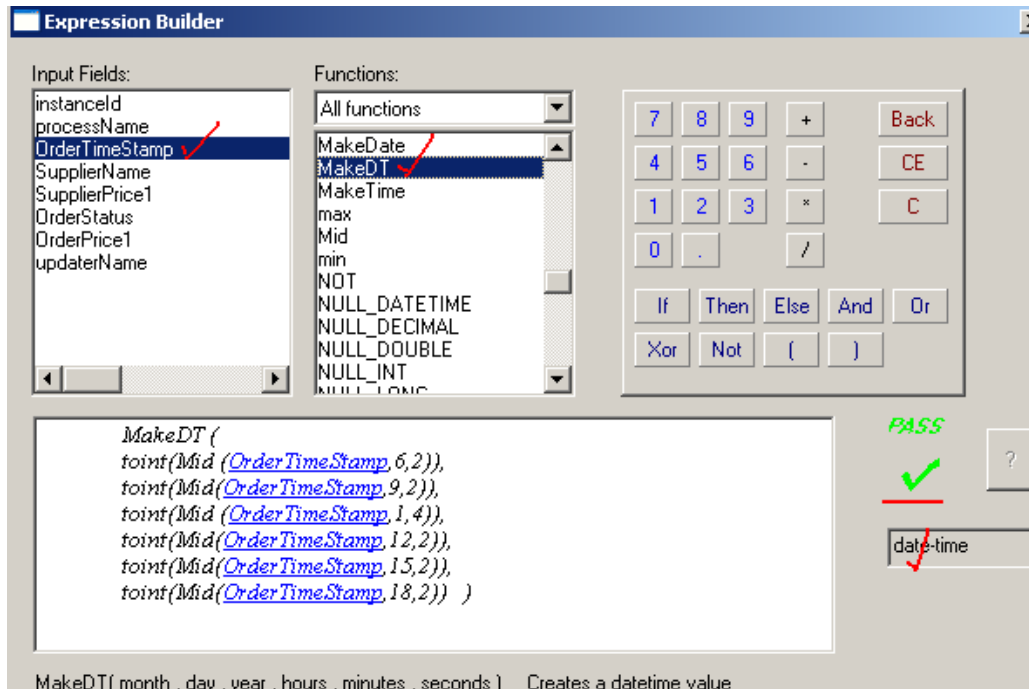
These steps design the converting mechanism to convert the string values into correct data formats. (date time objects, and decimals)

Double click on the Expression Calculator and open it. Uncheck the variables that are not required as below (OrderPrice1, SupplierPrice1, and OrderTimeStamp).



Click on “New” to create a new expression. In the expression builder window, create a new expression to convert the OrderTimeStamp from string to a “DateTime” value as below.

```
MakeDT (  
  toint(Mid (OrderTimeStamp,6,2)),  
  toint(Mid(OrderTimeStamp,9,2)),  
  toint(Mid (OrderTimeStamp,1,4)),  
  toint(Mid(OrderTimeStamp,12,2)),  
  toint(Mid(OrderTimeStamp,15,2)),  
  toint(Mid(OrderTimeStamp,18,2)) )
```

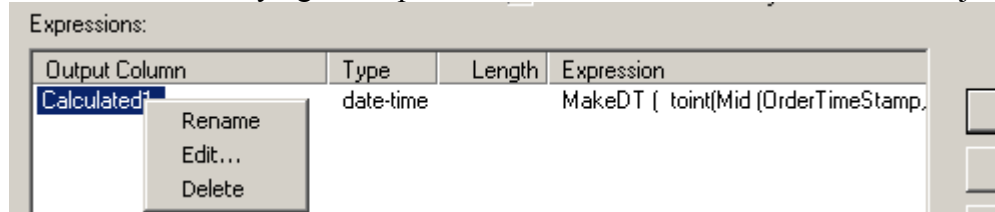




**Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC.**  
(Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

---

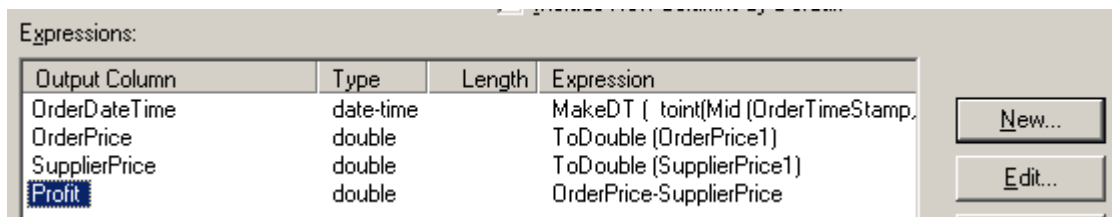
Click on “OK” after verifying the expression is correct and result is date-time object.



Rename the value as “OrderDateTime”.

Repeat similar instructions carefully and slowly to create three new expression to convert & calculate prices. After creating the expressions rename them as shown in the screen below

Expression1 Name: *OrderPrice* Expression Value: *toDouble(OrderPrice1)*  
Expression2 Name: *SupplierPrice* Expression Value: *toDouble(SupplierPrice1)*  
Expression3 Name: *Profit* Expression Value: *OrderPrice - SupplierPrice*



Click OK to Close & Save the expression builder window.

Next double click the conditional splitter, and edit the condition as follows.

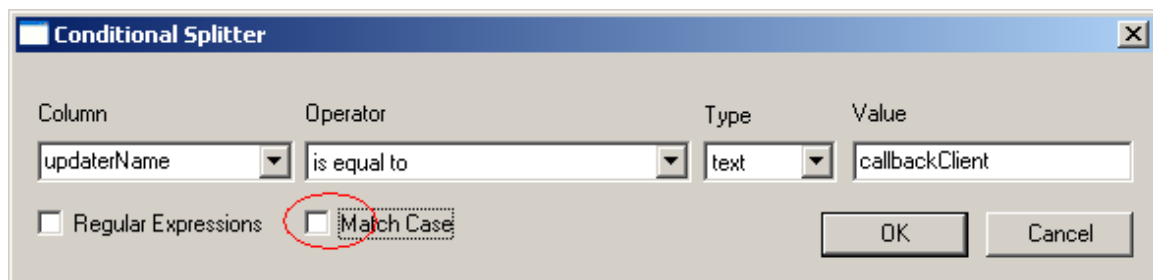
**Column:** “*updaterName*”

**Operator:** “*is equal to*”

**Type:** “*text*”

**Value:** “*callbackClient*”

Clear the “Match Case” selection.



## Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC. (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

This is to save only objects at end of OrderBookingProcess, i.e. only sensor messages of “callbackClient”.

This stage completes defining the plan to :

(i) connect to message source, (ii) the collect & parse message in xslt (iii) convert data to required format (expression builder) and (iv) conditionally store results in data objects.

### Verifying Plan Definition

Start pushing data using the small java program given with the lab tutorial.

On the dos prompt, start the given java program to push data to BAM.

*cd C:\XXX\OracleBAM\10.1.3\_Folder\Samples\SimulateJMSEvents  
2.SimulateEvents (Note this batch program takes command line arguments)  
Use the command line arguments to push more data if required.*

In the plan window, click on Update button. The resulting window will show you the data objects collected and persisted in the ADC. The window results below will display the contents of the grid as shown (grid shows data for condition not satisfied, because the data for conditions satisfied will be populated in the data object (see the condition splitter).

The screenshot shows the Oracle BAM Design Studio interface. At the top, a plan flow diagram is visible with components: Oracle BAM Enterprise Message Receiv Expression Calculator, Conditional Splitter, Oracle BAM Insert, and Grid. A red arrow labeled 'verify' points to the Conditional Splitter. Below the diagram is a data grid with 10 columns: instancelid, processName, SupplierName, OrderStatus, updaterName, OrderDateTi, OrderPrice, SupplierPri, Profit. The grid contains 8 rows of data. At the bottom right, there is an 'Update' button and a status bar showing 'Updated [22.94 seconds, 8 records]'.

	instancelid	processName	SupplierName	OrderStatus	updaterName	OrderDateTi	OrderPrice	SupplierPri	Profit
1	12130	OrderBooking	SelectManufa	Accepted	ReceivePOA	7/17/2005 6:4	5100.00	4000.00	1100.00
2	12203	OrderBooking	SelectManufa	Accepted	receiveInput	7/17/2005 6:4	6100.00	4000.00	2100.00
3	12203	OrderBooking	SelectManufa	Accepted	Invoke_RD	7/17/2005 6:4	6100.00	4000.00	2100.00
4	12203	OrderBooking	SelectManufa	Accepted	Invoke_SM	7/17/2005 6:4	6100.00	4000.00	2100.00
5	12203	OrderBooking			Receive_SM	7/17/2005 6:4	0.00	0.00	0.00
6	12203	OrderBooking			Receive_RD	7/17/2005 6:4	0.00	0.00	0.00
7	12203	OrderBooking	SelectManufa	Accepted	InvokePOASe	7/17/2005 6:4	6100.00	4000.00	2100.00
8	12203	OrderBooking	SelectManufa	Accepted	ReceivePOA	7/17/2005 6:4	6100.00	4000.00	2100.00

Verifying the results in the grid panel. Note – grid displays data for false condition (i.e. all events for which “updaterName” is not equal to callbackClient. (True condition goes to insert block)

(Important – Save before closing). Save the plan and close this DesignStudio application.

## Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC. (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

---

Open Architect window, and click on Data Objects and verify that BPELOrderBookingDataObject is created with correct data format and populated correctly.

**ORACLE<sup>®</sup> BAM Architect**

Data Objects

General | Layout | **Contents** | Security Filters | Permissions | Dimensions | Rename/Move | Indexes | Delete | Clear | Create

Edit Contents

Data Object "/StudentLab/BPELOrderBookingDataObject".  
1003 total rows | [Show row numbers](#)

First | Previous | [Next](#) | Last | [Refresh](#)

Row ID	instanceId	processName	SupplierName	OrderStatus	updaterName	OrderDateTime	Or
1	73580	OrderBooking	RapidDistributors	Rejected	callbackClient	3/14/2003 5:43:45 AM	611
2	78879	OrderBooking	RapidDistributors	Accepted	callbackClient	2/18/2003 5:43:50 AM	611
3	72886	OrderBooking	RapidDistributors	Rejected	callbackClient	1/22/2003 5:48:34 AM	611
4	72887	OrderBooking	RapidDistributors	Rejected	callbackClient	3/23/2003 5:48:40 AM	611

### **Completion:**

This completes the first exercise of defining the message source, connecting to the JMS message source, collecting data, parsing the XSLT, mapping the nodes to object columns (not final data objects). Defining plans to collect data from source, manipulating data per expression builder, format conversion etc, and finally populating data in the ADC.

### **Note:**

For real time continuous execution, replace above “grid” with “terminal grid”. Grid should be used for only display during debugging or development. For production runs or continuous runs, always use “terminal grid”.

### **Summary**

- Connect to the message source and verify incoming data.
- Define a plan, and collect data from a message source.
- Parse collected data, manipulate data, and store in data objects.

### **Complete Plan**

In this part, we will extend the Oracle BAM Plans to correlate sensor data received at various time intervals, and to collect these time differed data into a correlated data set – the correlation id used for matching various time delayed data set is the instance id. The design involves collecting the timestamps with reference to each instance, and designing correlated event sets to collect all info uniquely about each event for each timestamp.

## Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC. (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

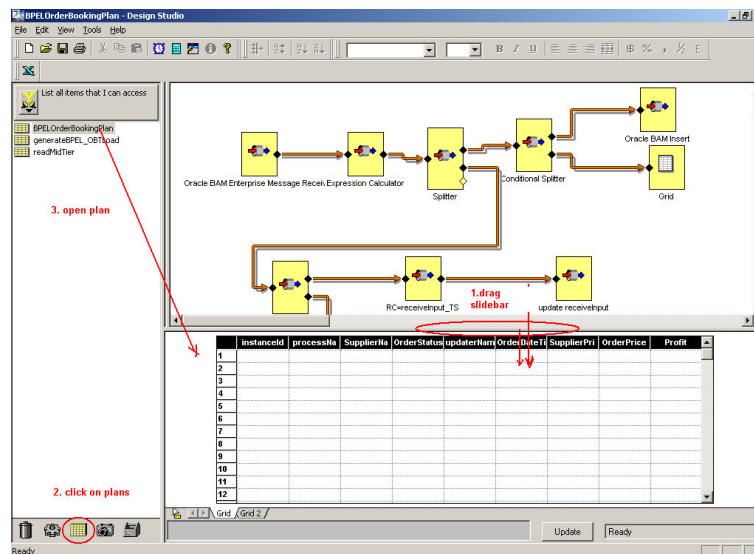
---

### Importing the plan

Import the pre-defined completed plan which demonstrates collecting event data and populating the ADC object, based on correlation ids of various events collected at discrete time intervals. Open dos prompt and import predefined completed plan for correlation events.

*C:\XXX\OracleBAM\10.1.3\_Folder\Samples\BPEL-JMS-BAM\OrderBookingDemo\SampleDir  
iCommand cmd=import file=BPELOrderBookingPlan.plan mode=overwrite*

This should successfully import the completed BPEL plan extended to collect data into BPELOrderBookingDataObject and also correlate events and collect data into the time stamp object BPELOrderBookingTimestamp object. Open the Design Studio application and open the plan BPELOrderBookingPlan. Verify that the plan is imported successfully and appears as below. IMPORTANT: Save the plan and exit the application.



### Optional parts to push data from BPEL process:

#### Prerequisite on BPEL:

If you are doing this optional part, close the OC4J container started at the start of this document. BPEL server installation. Install standalone BPEL server, to be used in later parts for collecting BPEL sensor data.

### **Install 127.OrderBookingTutorial**

1. You should be familiar with BPEL concepts to do this section. Open BPEL Developer “dos” prompt. Deploy OrderBookingTutorial from BPEL samples directory using “obant”. Verify in BPEL console, that the OrderBooking process is deployed correctly. Initiate 1 or 2 (or few) instances and complete the process end to end, with SelectManufacturing GUI and Order Approval work list.

## **Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC.** (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

---

2. To start an instance of 127.OrderBooking process, copy the files OrderBookingPO\_1.xml, OrderBookingPO\_2.xml and OrderBookingPO\_3.xml from the <yourBPELSamplesDir>\127.OrderBookingTutorial\PracticeFiles folder to c:\temp.
3. The OrderBooking process is configured to publish/write sensor data into JMS topics. This is your main sensor data input source.
4. Observe that BPEL process is running in Start BPEL dos window. Complete the process instance using SelectManufacturing GUI and Order Approval work list GUI. This will start pushing BPEL events to the.jms bus (on the BPEL oc4j container)

### **Verify BPEL JMS configuration (Verification Only – no changes needed)**

BPEL server publishes its JMS messages on the JMS topic as given in the.jms.xml configuration file under

<BPELinstallDir>\integration\orabpel\system\appserver\oc4j\j2ee\home\config\jms.xml

This configuration file contains.jmsTopicConnection factory and.jms/demoTopic as shown below.

```

    JNDI path for later retrieval -->
    <queue name="Demo Queue" location="jms/demoQueue">
      <description>A dummy queue</description>
    </queue>

    <!-- Topic bindings, these topic will be bound to their respective
    JNDI path for later retrieval -->
    <topic name="Demo Topic" location="jms/demoTopic">
      <description>A dummy topic</description>
    </topic>

    <!-- path to the log-file where JMS-events/errors are stored -->
    <log>
      <file path="../../log/jms.log"/>
      <!-- Uncomment this if you want to use ODL logging capabilities
      <odl path="../../log/jms/" max-file-size="1000" max-directory-size="10000"/>
      -->
    </log>

    <queue name="jms/OracleUddiReplicationQueue"
      location="jms/OracleUddiReplicationQueue">
      <description>Queue for replication scheduler</description>
    </queue>

    <!--
    <queue-connection-factory
      name="jms/OracleUddiReplicationQueueConnectionFactory"
      location="jms/OracleUddiReplicationQueueConnectionFactory"/>
    -->
    <queue-connection-factory location="jms/OracleUddiReplicationQueueConnectionFactory"/>

    <queue name="jms/OracleWebClippingQueue"
      location="jms/OracleWebClippingQueue">
      <description>Queue for Web Clipping</description>
    </queue>

    <!--
    <queue-connection-factory
      name="jms/OracleWebClippingQueueConnectionFactory"
      location="jms/OracleWebClippingQueueConnectionFactory"/>
    -->
    <queue-connection-factory location="jms/OracleWebClippingQueueConnectionFactory"/>

```

These same parameters are used by the BAM server to connect to the.jms bus. (as given in the configuration dialog box).

## Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC. (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

---

```
<!-- Orabpel -->
<queue name="BPelWorkerQueue" location="jms/collaxa/BPELWorkerQueue"/>
<queue-connection-factory name="BPelWorkerQueueFactory" location="jms/collaxa/BPELWorkerQueueFactory"/>

<queue name="BPelInvokerQueue" location="jms/collaxa/BPELInvokerQueue"/>
<queue-connection-factory name="BPelInvokerQueueFactory" location="jms/collaxa/BPELInvokerQueueFactory"/>

<!-- Queues and Topics for Oracle BPEL Samples -->
<queue name="ExceptionQueue" location="jms/orabpel_samples_ExceptionQueue"/>
<queue name="SellerQueue" location="jms/orabpel_samples_SellerQueue"/>
<queue name="BuyerQueue" location="jms/orabpel_samples_BuyerQueue"/>
<queue-connection-factory name="BPESamplesQueueFactory" location="jms/QueueConnectionFactory"/>

<topic name="SellerTopic" location="jms/orabpel_samples_SellerTopic"/>
<topic name="BuyerTopic" location="jms/orabpel_samples_BuyerTopic"/>
<topic-connection-factory name="BPESamplesTopicFactory" location="jms/TopicConnectionFactory"/>

</jms-server>
```

Close the jms.xml file.

## **Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC.** (Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

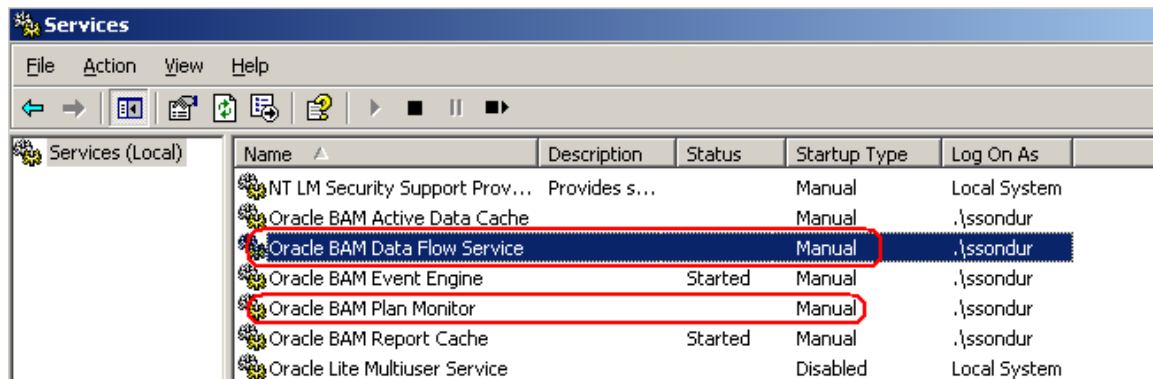
---

### **Optional parts to push data from J2EE program to BAM webservice process:**

**Note :** You need special jar files for this section of the tutorial. See readme.txt file under the samples directory for `**\Samples\J2EE-BAM-WS`.

This section provides a sample program to push data directly to BAM ADC data objects using the BAM web services interface. A sample J2EE application is provided to illustrate this example.

Open Windows services, and turn off Plan Monitoring and Data Flow Service as shown.



Turning off the above services guarantees that the data does not flow thro Enterprise Link Data Flow Services or use the plan monitor service, but data directly goes thro the webservice calls.

Open the sample directory and change directory to: `<urDir>\Samples\J2EE-BAM-WS`.

Run the samples program: `SimulateWSEvents.bat`.

**Note:** this program takes command line arguments such as your machine name, login name, password etc.

Provide the correct command line parameters to this program, and observe this sample Java program push data into ADC through the web service interface.

Open BAM console, Architect, Data Objects and observe new data populated in the

`/StudentLab/BPELOrderBookingDataObject` and

`/StudentLab/BPELOrderBookingTimestamp`.

### **Other examples:**

Other examples of pushing real time data into BAM ADC is given in the sample directory. For example – thro JMS bus, thro. Oracle AQ, etc.

**Tutorial Dependent XSLT file:**

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:sensor="http://xmlns.oracle.com/bpel/sensor"
    xmlns:n1="http://www.globalcompany.com/ns/sales"
    exclude-result-prefixes="xsl sensor n1">
  <xsl:output method="xml" omit-xml-declaration="yes"/>

  <xsl:template match="/">
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="sensor:*">
    <xsl:element name="{local-name()}">
      <xsl:apply-templates/>
    </xsl:element>
  </xsl:template>

  <xsl:template match="n1:*">
    <xsl:element name="{local-name()}">
      <xsl:apply-templates/>
    </xsl:element>
  </xsl:template>

  <xsl:template match="/">
    <actionData>
      <instanceId>
        <xsl:value-of select="/sensor:actionData/sensor:header/sensor:instanceId"/>
      </instanceId>
      <processName>
        <xsl:value-of
select="/sensor:actionData/sensor:header/sensor:processName"/>
        </processName>
      <OrderTimeStamp>
        <xsl:value-of select="/sensor:actionData/sensor:header/sensor:timestamp"/>
      </OrderTimeStamp>
      <SupplierName>
        <xsl:value-of
select="/sensor:actionData/sensor:payload/sensor:variableData/sensor:data/n1:PurchaseOrder/
n1:SupplierInfo/n1:SupplierName"/>
        </SupplierName>
      <SupplierPrice1>
        <xsl:value-of
select="/sensor:actionData/sensor:payload/sensor:variableData/sensor:data/n1:PurchaseOrder/
n1:SupplierInfo/n1:SupplierPrice"/>
        </SupplierPrice1>
      <OrderStatus>
        <xsl:value-of
select="/sensor:actionData/sensor:payload/sensor:variableData/sensor:data/n1:PurchaseOrder/
n1:OrderInfo/n1:OrderStatus"/>
      </OrderStatus>
    </actionData>
  </xsl:template>
</xsl:stylesheet>
```



**Tutorial : Oracle BAM Plans to collect & parse data from JMS and populate ADC.**  
(Tutorial\_6\_BAM\_PopulatingDatainADC.doc)

---

```
</OrderStatus>
  <OrderPrice1>
    <xsl:value-of
select="/sensor:actionData/sensor:payload/sensor:variableData/sensor:data/n1:PurchaseOrder/
n1:OrderInfo/n1:OrderPrice"/>
  </OrderPrice1>
  <updaterName>
    <xsl:value-of
select="/sensor:actionData/sensor:payload/sensor:variableData/sensor:updaterName"/>
  </updaterName>
</actionData>
</xsl:template>

</xsl:stylesheet>
```

**Questions & Clarifications:**

If you have any comments or need additional information, please communicate through the Oracle BAM forum at: <http://forums.oracle.com/forums/forum.jspa?forumID=252>