

Oracle® SOA Suite

New Features

10g (10.1.3.3)

E10381-02

September 2007

This document describes new features available with the Oracle SOA Suite for release 10.1.3.3 patch set.

This document contains the following topics:

- [Deploying a Custom Worklist Application and Enabling SSO](#)
- [Changing Workflow Standard View Definitions](#)
- [New Database Views for Oracle Workflow](#)
- [File/FTP Adapter New Features](#)
- [MQSeries Adapter New Features](#)
- [Database Adapter New Features](#)
- [AQ Adapter New Features](#)
- [JMS Adapter New Features](#)
- [Oracle ESB Singleton Behavior for Inbound Adapter Endpoints](#)
- [Decision Service Support for Ilog JRules](#)
- [Fault Management Framework](#)

See Also: Additional documentation included with this patch set:

- *Oracle Application Server Patch Set Notes Addendum* for details about known issues with the patch set
- *Oracle Application Server Patch Set Notes* for details about applying the patch set
- *Oracle Application Server Fixed Bugs List* for details about bugs fixed with the patch set

Deploying a Custom Worklist Application and Enabling SSO

In 10.1.3.1, deploying a custom Oracle BPEL Worklist Application (Worklist Application) and enabling single sign-on (SSO) for the Worklist Application required a number of workarounds. Bug fixes introduced in 10.1.3.3 simplify the process considerably. This section describes the revised processes for deploying a custom Worklist Application and enabling SSO in the preinstalled Worklist Application.

Deploying the Custom Worklist Application

The top-level directory of the sample Worklist Application contains an ant script, `build.xml`, for building and deploying the Worklist Application. This ant script uses the `orabpel.properties` file that exists in the same directory.

The following steps describe how to build and deploy the custom Worklist Application:

1. Ensure that all the properties in `orabpel.properties` have been updated to reflect your environment.
2. Build and deploy the customized Worklist Application from the operating system command prompt:

```
ant deploy.oc4j
```

Or run `ant deployssso.oc4j`, if you want the custom Worklist Application to be Java single sign-on (JSSO)-enabled.

3. Access the customized Worklist Application at the following URL:

```
http://host:port/integration/customapp/
```

4. Log in to the Worklist Application.

The task list page appears.

Enabling the Worklist Application for Single Sign-On

By default, the 10.1.3.3 Worklist Application uses a custom authentication mechanism through its own login page. The preinstalled Worklist Application does not run under OC4J container security and is not JSSO-enabled.

When deploying a custom Worklist Application, you can deploy it to be either JSSO-enabled or to use the Worklist Application login page (see ["Deploying the Custom Worklist Application"](#) on page 2). It is also possible to reconfigure the preinstalled Worklist Application to be JSSO-enabled. This section describes how to perform this task.

Note that JSSO does not provide multirealm support. If you are using an identity provider configured with multiple realms, Oracle recommends that you not use a JSSO-enabled Worklist Application.

This section contains the following topics:

- [Task 1: Updating web.xml for the Worklist Application](#)
- [Task 2: Enabling JSSO for the Worklist Application in Oracle Enterprise Manager 10g Application Server Control Console](#)

Task 1: Updating web.xml for the Worklist Application

1. Go to the directory `SOA_ORACLE_HOME\j2ee\home\applications\hw_services\worklistapp\WEB-INF`.
2. Copy the file `web.xml` to `webnonsso.xml` (so you can revert the changes if necessary).
3. Copy the file `websso.xml` to `web.xml`. This enables container-managed security for the Worklist Application.

Task 2: Enabling JSSO for the Worklist Application in Oracle Enterprise Manager 10g Application Server Control Console

1. Start the Oracle Enterprise Manager 10g Application Server Control Console for the OC4J server that is hosting your SOA instance.
2. Click the **Administration** tab.
3. Click the **SSO Configuration** task.
4. Click on the **Participating Applications** tab.
5. Click the check box next to **hw_services**.
6. Click **Apply**.
7. Restart the server.

When you access the preinstalled Worklist Application, the JSSO login page appears, rather than the Worklist Application login page.

Changing Workflow Standard View Definitions

The workflow service includes a number of standard view definitions. These views define standard queries against a user's list of tasks. The views are available to all users. They are displayed in the Worklist Application, and can be queried through the user metadata service.

While these views are useful in themselves, you often want to modify the existing standard view definitions, or create your own standard views. The Worklist Application and the user metadata service do not provide a way of doing this.

The standard views included with the workflow service are defined in an XML file named `StandardTaskViews.xml` that gets loaded from the workflow service's class path. The contents of this file are shown in "[Contents of StandardTaskViews.xml](#)" on page 5.

You can modify the contents of this file by editing existing views or adding new views. Place the edited version of this file in the following directory:

```
SOA_ORACLE_  
HOME\bpel\system\classes\oracle\bpel\services\workflow\user\config\
```

When the server is restarted, it loads the new standard view definitions.

When defining new standard views, the existing standard views can be used as a guideline.

The following XML elements define a view:

- `<id>` — standard view IDs must be unique, and must begin with `ORCL_WF_STD_VIEW`.
- `<name>` — the name you give to the standard view.
- `<owner>` — this is not used for standard views; copy the dummy owner element from another standard view.
- `<hidden>` — this should be `false`.
- `<description>` — a short description of the view.

- `<viewColumns>` — a list of the columns for the view to display. The element should contain one or more `<column>` elements. Each `<column>` element should contain a `<columnName>` element that specifies the name of the column, together with an optional `<displayName>` element. If a value is specified for `<displayName>`, it overrides the default column header used by the Worklist Application.
- `<viewOptionalInfo>` — if the view is to display a column containing drop-down lists of permitted actions for each task, this element should contain a `<taskOptionalInfo>` element with a value of Actions (as specified in the views that are already listed in the file). Otherwise, the `<viewOptionalInfo>` element can be empty.
- `<viewPredicate>` — Defines the query predicate for the view. It can contain the following elements:
 - `assignmentFilter` — This element is mandatory. It must contain one of the following values:
 - * My
 - * My+Group
 - * Group
 - * Owner
 - * Previous
 - * Reportees
 - `<keywords>` — A view can optionally specify keywords. A view filters tasks if any of these task attributes contain the specified string: task title, identification key, all text attribute columns in the task, and task number (only if the keyword is a number).
 - `<clause>` — A view can optionally specify one or more predicate clauses to filter tasks. A joinOperator of OR or AND can be specified for each `<clause>`. Each `<clause>` element contains the following:
 - * `<column>` — the name of the task column.
 - * `<operator>` — the operator to use for the predicate. The allowable operators depend on the type of the column:

For string columns: eq (equals), neq (not equals), begins, not_begins, ends, not_ends, like, not_like, contains, not_contains, is_null, is_not_null, in, and not_in.

For number columns: eq (equals), neq (not equals), lt (less than), lte (less than or equal to), gt (greater than), gte (greater than or equal to), is_null, is_not_null, in, and not_in.

For date columns: eq (equals), neq (not equals), lt (less than), lte (less than or equal to), gt (greater than), gte (greater than or equal to), is_null, is_not_null, next_n_days, and last_n_days.
 - * `<value>` — The value associated with the column and the operator. Note the following exceptions:

No value needs to be specified for the is_null and is_not_null operators.

If using the `in` or `not_in` operator, use a `<valueList>` element instead. The `<valueList>` element must contain one or more `<value>` elements containing the values to specify for the `in` or `not_in` join.

If using a date column, and not using the `next_n_days` or `last_n_days` operator, use a `<dateValue>` element instead. This should contain the required date for the clause, as specified in `xsd:dateTime`.

- `<viewOrdering>` — the view can specify how to order tasks, using one or more ordering `<clause>` elements. Each `<clause>` element contains:
 - `<column>` — name of column on which to order
 - `<sortOrder>` — ascending or descending order
 - `<nullFirst>` — should null values appear first or last in the order (true or false).
- `<chart>` — this is not used by the Worklist Application.

When defining a view that displays filters on flex fields, use the flex field column name (for example, `TextAttribute1`), rather than the attribute label name. To ensure that the Worklist Application can determine the appropriate attribute label name when displaying the flex field column in the view, also specify a clause in the view for the `taskDefinitionId` of the task type of the flex field mapping. For example:

```
<clause>
  <column>taskDefinitionId</column>
  <operator>eq</operator>
  <value>[the task definition id]</value>
</clause>
```

Alternatively, you can use the `<displayName>` element to define a display name for the column.

Contents of StandardTaskViews.xml

This section shows the contents of the `StandardTaskViews.xml` file. Copy the contents into your own `StandardTaskViews.xml` file, and edit it to make changes to the standard views.

```
<StandardTaskViews>
  <!--High Priority Tasks -->
  <userViewDetail viewType="VIEW"
    xmlns="http://xmlns.oracle.com/bpel/workflow/userMetadata">
    <id>ORCL_WF_STD_VIEW_HIGH_PRIORITY_TASKS</id>
    <name>STD_VIEW_HIGH_PRIORITY_TASKS</name>
    <owner type="USER">
      <realm
        xmlns="http://xmlns.oracle.com/bpel/workflow/common">iPlanetRealm</realm>
        <name
          xmlns="http://xmlns.oracle.com/bpel/workflow/common">oc4jadmin</name>
        </owner>
        <hidden>false</hidden>
        <description>High Priority Tasks (Priority IsEq 5)</description>
        <viewColumns>
          <column>
            <columnName>taskNumber</columnName>
```

```

        <displayName></displayName>
    </column>
    <column>
        <columnName>title</columnName>
        <displayName></displayName>
    </column>
    <column>
        <columnName>priority</columnName>
        <displayName></displayName>
    </column>
    <column>
        <columnName>State</columnName>
        <displayName></displayName>
    </column>
    <column>
        <columnName>createdDate</columnName>
        <displayName></displayName>
    </column>
    <column>
        <columnName>expirationDate</columnName>
        <displayName></displayName>
    </column>
</viewColumns>
<viewOptionalInfo>
    <taskOptionalInfo xmlns="http://xmlns.oracle.com/bpel/workflow/
        taskQuery">Actions</taskOptionalInfo>
</viewOptionalInfo>
<viewPredicate>
    <assignmentFilter xmlns="http://xmlns.oracle.com/bpel/workflow/
        taskQuery">My+Group</assignmentFilter>
    <keywords
xmlns="http://xmlns.oracle.com/bpel/workflow/taskQuery"></keywords>
    <clause joinOperator="AND"
xmlns="http://xmlns.oracle.com/bpel/workflow/taskQuery">
        <column>priority</column>
        <operator>lte</operator>
        <value>2</value>
    </clause>
    <clause joinOperator="AND"
xmlns="http://xmlns.oracle.com/bpel/workflow/taskQuery">
        <column>state</column>
        <operator>eq</operator>
        <value>ASSIGNED</value>
    </clause>
</viewPredicate>
<viewOrdering>
    <clause xmlns="http://xmlns.oracle.com/bpel/workflow/taskQuery">
        <column>expirationDate</column>
        <sortOrder>ascending</sortOrder>
        <nullFirst>false</nullFirst>
    </clause>
</viewOrdering>

    <chart>
        <groupByColumn>state</groupByColumn>
    </chart>
</userViewDetail>

<!-- Due Soon -->

```

```

<userViewDetail viewType="VIEW"
xmlns="http://xmlns.oracle.com/bpel/workflow/userMetadata">
  <id>ORCL_WF_STD_VIEW_TASKS_DUE_SOON</id>
  <name>STD_VIEW_TASKS_DUE_SOON</name>
  <owner type="USER">
    <realm
xmlns="http://xmlns.oracle.com/bpel/workflow/common">iPlanetRealm</realm>
      <name
xmlns="http://xmlns.oracle.com/bpel/workflow/common">oc4jadmin</name>
    </owner>
    <hidden>false</hidden>
    <description>Due Soon (expires within next 24 hours)</description>
    <viewColumns>
      <column>
        <columnName>taskNumber</columnName>
        <displayName></displayName>
      </column>
      <column>
        <columnName>title</columnName>
        <displayName></displayName>
      </column>
      <column>
        <columnName>priority</columnName>
        <displayName></displayName>
      </column>
      <column>
        <columnName>State</columnName>
        <displayName></displayName>
      </column>
      <column>
        <columnName>createdDate</columnName>
        <displayName></displayName>
      </column>
      <column>
        <columnName>expirationDate</columnName>
        <displayName></displayName>
      </column>
    </viewColumns>
    <viewOptionalInfo>
      <taskOptionalInfo xmlns="http://xmlns.oracle.com/bpel/workflow/
        taskQuery">Actions</taskOptionalInfo>
    </viewOptionalInfo>
    <viewPredicate>
      <assignmentFilterxmlns="http://xmlns.oracle.com/bpel
        /workflow/taskQuery">My+Group</assignmentFilter>
      <keywords
xmlns="http://xmlns.oracle.com/bpel/workflow/taskQuery"></keywords>
      <clause joinOperator="AND"
xmlns="http://xmlns.oracle.com/bpel/workflow/taskQuery">
        <column>expirationDate</column>
        <operator>next_n_days</operator>
        <value>1</value>
      </clause>
      <clause joinOperator="AND"
xmlns="http://xmlns.oracle.com/bpel/workflow/taskQuery">
        <column>state</column>
        <operator>eq</operator>
        <value>ASSIGNED</value>
      </clause>

```

```

</viewPredicate>
<viewOrdering>
  <clause xmlns="http://xmlns.oracle.com/bpel/workflow/taskQuery">
    <column>priority</column>
    <sortOrder>descending</sortOrder>
    <nullFirst>>false</nullFirst>
  </clause>
</viewOrdering>
<chart>
  <groupByColumn>priority</groupByColumn>
</chart>
</userViewDetail>

<!--New Tasks -->
<userViewDetailViewType="VIEW"
xmlns="http://xmlns.oracle.com/bpel/workflow/userMetadata">
  <id>ORCL_WF_STD_VIEW_NEW_TASKS</id>
  <name>STD_VIEW_NEW_TASKS</name>
  <owner type="USER">
    <realm
xmlns="http://xmlns.oracle.com/bpel/workflow/common">iPlanetRealm</realm>
    <name
xmlns="http://xmlns.oracle.com/bpel/workflow/common">oc4jadmin</name>
  </owner>
  <hidden>>false</hidden>
  <description>New Tasks (created within past 24 hours)</description>
  <viewColumns>
    <column>
      <columnName>taskNumber</columnName>
      <displayName></displayName>
    </column>
    <column>
      <columnName>title</columnName>
      <displayName></displayName>
    </column>
    <column>
      <columnName>priority</columnName>
      <displayName></displayName>
    </column>
    <column>
      <columnName>State</columnName>
      <displayName></displayName>
    </column>
    <column>
      <columnName>createdDate</columnName>
      <displayName></displayName>
    </column>
    <column>
      <columnName>expirationDate</columnName>
      <displayName></displayName>
    </column>
  </viewColumns>
  <viewOptionalInfo>
    <taskOptionalInfo xmlns="http://xmlns.oracle.com/bpel/workflow/
      taskQuery">Actions</taskOptionalInfo>
  </viewOptionalInfo>
  <viewPredicate>
    <assignmentFilter xmlns="http://xmlns.oracle.com/bpel/workflow/
      taskQuery">My+Group</assignmentFilter>

```



```

        <keywords xmlns="http://xmlns.oracle.com/bpel/
        workflow/taskQuery"></keywords>
        <clause joinOperator="AND"
xmlns="http://xmlns.oracle.com/bpel/workflow/taskQuery">
        <column>createdDate</column>
        <operator>last_n_days</operator>
        <value>1</value>
        </clause>
        <clause joinOperator="AND"
xmlns="http://xmlns.oracle.com/bpel/workflow/taskQuery">
        <column>state</column>
        <operator>eq</operator>
        <value>ASSIGNED</value>
        </clause>
    </viewPredicate>
    <viewOrdering>
        <clause xmlns="http://xmlns.oracle.com/bpel/workflow/taskQuery">
        <column>priority</column>
        <sortOrder>descending</sortOrder>
        <nullFirst>false</nullFirst>
        </clause>
    </viewOrdering>
    <chart>
        <groupByColumn>priority</groupByColumn>
    </chart>
</userViewDetail>
</StandardTaskViews>

```

New Database Views for Oracle Workflow

This section describes database views that were added to allow queries against the Oracle workflow services schema to get reports. The following table lists the reports exposed in the worklist application and the database views corresponding to these reports.

Existing Worklist Report	Corresponding Database View
Unattended Tasks report	WFUNATTENDEDTASKS_VIEW
Task Cycle Time report	WFTASKCYCLETIME_VIEW
Task Productivity report	WFPRODUCTIVITY_VIEW
Task Priority Report	WFTASKPRIORITY_VIEW

Note: Refer to the Worklist Application documentation in *Oracle BPEL Process Manager Developer's Guide* for details about these reports

The following sections describe each database view with samples:

- [Unattended Tasks Report View](#)
- [Task Cycle Time Report View](#)
- [Task Productivity Report View](#)
- [Task Priority Report View](#)

Unattended Tasks Report View

View Name: WFUNATTENDEDTASKS_VIEW

View Description:

Name	Type
TASKID ¹	VARCHAR2 (64)
TASKNAME	VARCHAR2 (200)
TASKNUMBER	NUMBER
CREATEDDATE	DATE
EXPIRATIONDATE	DATE
STATE	VARCHAR2 (100)
PRIORITY	NUMBER
ASSIGNEEGROUPS	VARCHAR2 (2000)

¹ NOT NULL column

Samples:

- Query unattended tasks that have an expiration date of next week:

```
SELECT tasknumber, taskname, assigneegroups FROM WFUNATTENDEDTASKS_VIEW
WHERE expirationdate > current_date AND expirationdate < current_date +
7;
```

- Query unattended tasks for mygroup:

```
SELECT tasknumber, taskname, assigneegroups FROM WFUNATTENDEDTASKS_VIEW
WHERE 'mygroup' IN assigneegroups;
```

- Query unattended tasks created in the last 30 days:

```
SELECT tasknumber, taskname, assigneegroups FROM WFUNATTENDEDTASKS_VIEW
WHERE createddate > current_date -30;
```

Task Cycle Time Report View

View Name: WFTASKCYCLETIME_VIEW

View Description:

Name	Type
TASKID ¹	VARCHAR2 (64)
TASKNAME	VARCHAR2 (200)
TASKNUMBER	NUMBER
CREATEDDATE	DATE
ENDDATE	DATE
CYCLETIME	NUMBER (38)

¹ NOT NULL column

Samples:

- Compute the average cycle time (task completion time) for completed tasks that were created in the last 30 days:

```
SELECT avg(cycletime) FROM WFTASKCYCLETIME_VIEW WHERE createddate >
(current_date - 30);
```

- Query the average cycle time for all completed tasks created in the last 30 days and group them by task name:

```
SELECT taskname, avg(cycletime) FROM WFTASKCYCLETIME_VIEW WHERE
createddate > (current_date - 30) GROUP BY taskname;
```

- Query the least and most time taken by each task:

```
SELECT taskname, min(cycletime), max(cycletime) FROM WFTASKCYCLETIME_VIEW
GROUP BY taskname;
```

- Compute the average cycle time for tasks completed in the last seven days:

```
SELECT avg(cycletime) FROM WFTASKCYCLETIME_VIEW WHERE enddate >
(current_date - 7);
```

- Query tasks that took more than seven days to complete:

```
SELECT taskname, avg(cycletime) FROM WFTASKCYCLETIME_VIEW WHERE cycletime
> ((current_date +7) - current_date) GROUP BY taskname;
```

Task Productivity Report View

View Name: WFPRODUCTIVITY_VIEW

View Description:

Name	Type
TASKNAME	VARCHAR2(200)
TASKID	VARCHAR2(200)
TASKNUMBER	NUMBER
USERNAME	VARCHAR2(200)
STATE ¹	VARCHAR2(100)
LASTUPDATEDDATE	DATE

¹ For completed tasks, the state is null. Use decode(outcome, '', 'COMPLETED', outcome) in queries.

Samples:

- Count the number of unique tasks that the user has updated in the last 30 days:

```
SELECT username, count(distinct(taskid)) FROM WFPRODUCTIVITY_VIEW WHERE
lastupdateddate > (current_date -30) GROUP BY username;
```

- Count the number of tasks that the user has updated (one task may have been updated multiple times) in the last seven days:

```
SELECT username, count(taskid) FROM WFPRODUCTIVITY_VIEW WHERE
```

```
lastupdateddate > (current_date -7) GROUP BY username;
```

- Count the number of tasks of each task type on which the user has worked:

```
SELECT username, taskname, count(taskid) FROM WFPRODUCTIVITY_VIEW GROUP  
BY username, taskname;
```

- Count the number of tasks of each task type that the user has worked on in the last 100 days:

```
SELECT username, taskname, count(taskid) FROM WFPRODUCTIVITY_VIEW WHERE  
lastupdateddate > (current_date -100) GROUP BY username, taskname;
```

Task Priority Report View

View Name: WFTASKPRIORITY_VIEW

View Description:

Name	Type
TASKID ¹	VARCHAR2 (64)
TASKNAME	VARCHAR2 (200)
TASKNUMBER	NUMBER
PRIORITY	NUMBER
OUTCOME	VARCHAR2 (100)
ASSIGNEDDATE	DATE
UPDATEDDATE	DATE
UPDATEDBY	VARCHAR2 (64)

¹ NOT NULL column

Samples:

- Query the number of tasks updated by each user in each task priority:

```
SELECT updatedby, priority, count(taskid) FROM WFTASKPRIORITY_VIEW GROUP  
BY updatedby, priority;
```

- Query task-to-outcome distribution:

```
SELECT taskname, decode(outcome, '', 'COMPLETED', outcome), count  
(taskid) FROM WFTASKPRIORITY_VIEW GROUP BY taskname, outcome;
```

- Query the number of tasks updated by the given user in each priority:

```
SELECT priority, count(taskid) FROM WFTASKPRIORITY_VIEW WHERE  
updatedby='jstein' GROUP BY priority;
```

File/FTP Adapter New Features

This section describes new features for the File/FTP adapter.

This section contains the following topics:

- [Appending Files to an Existing File](#)

- [Transferring Large Payloads in Oracle BPEL Process Manager](#)
- [Reading File Names from an Outbound Operation in Oracle BPEL Process Manager](#)
- [Controlling the Size of a Rejected Message](#)
- [Specifying Unique File Names When Using Time Pattern as Part of the Naming Convention for Outbound Partner Links](#)

Appending Files to an Existing File

The File/FTP adapter now enables you to configure outbound interactions to append files to existing files. To append to a file, add `Append="true"` in the `InteractionSpec` for the File/FTP adapter.

Note: For this feature to work properly in the FTP adapter, the destination FTP server must support the RFC 959 APPE command.

The following syntax shows how to append to the same file in the outbound directory.

```
<jca:operation
  FileType="ascii"
  PhysicalDirectory="/home/adapter/out"
  . . .
  FileNamingConvention="MyOutputFile.txt"
  NumberMessages="1"
  . . .
  . . .
  Append="true"
>
```

The file name can either be specified in the WSDL (as shown in the preceding syntax), or through the header.

Transferring Large Payloads in Oracle BPEL Process Manager

There are two new options for supporting large payloads:

- [Attachment Support in Oracle BPEL Process Manager](#)
- [XML Debatching](#)

Attachment Support in Oracle BPEL Process Manager

This feature is only available with Oracle BPEL Process Manager. As a general rule, use this feature to handle nondebatchable files larger than 10 MB.

The following example shows how to transfer a file from a source to a destination without any translation or transformation.

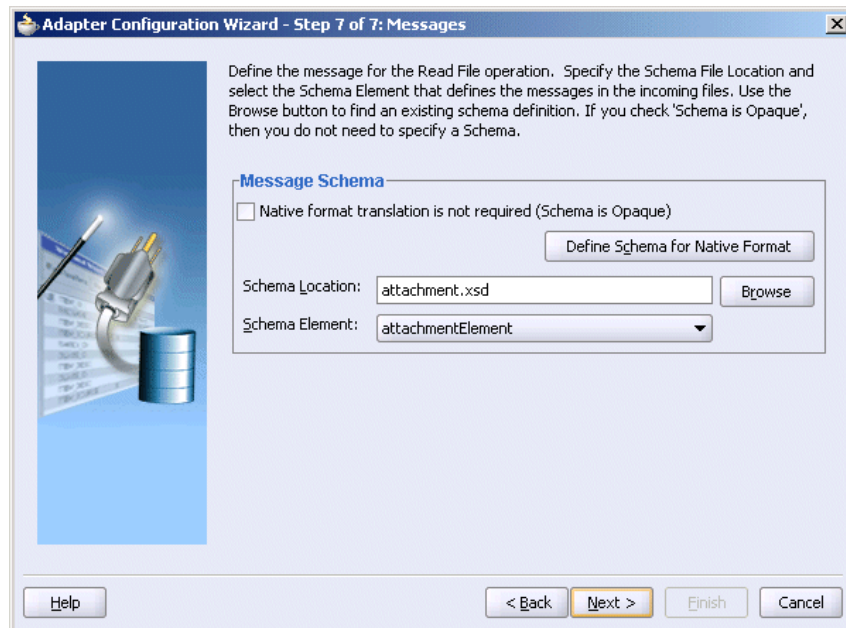
Note: Name the XSD file as `attachment.xsd` in this example.

To configure the adapter to publish files as an attachment, you must do the following:

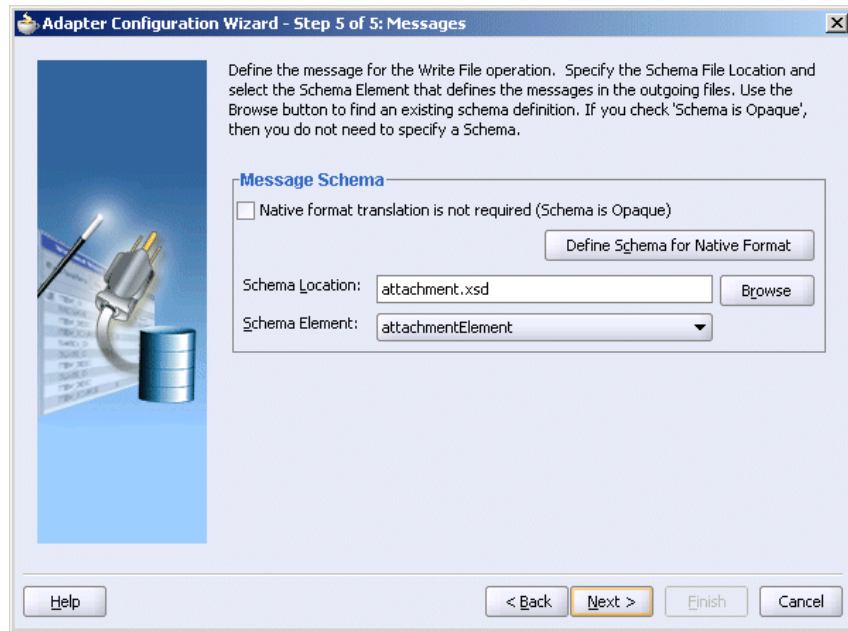
1. Import or add the following schema into the BPEL project. Note that the namespaces must have the exact same names.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/file/attachment/"
elementFormDefault="qualified">
<element name="attachmentElement">
  <complexType>
    <attribute name="href" type="string"/>
  </complexType>
</element>
</schema>
```

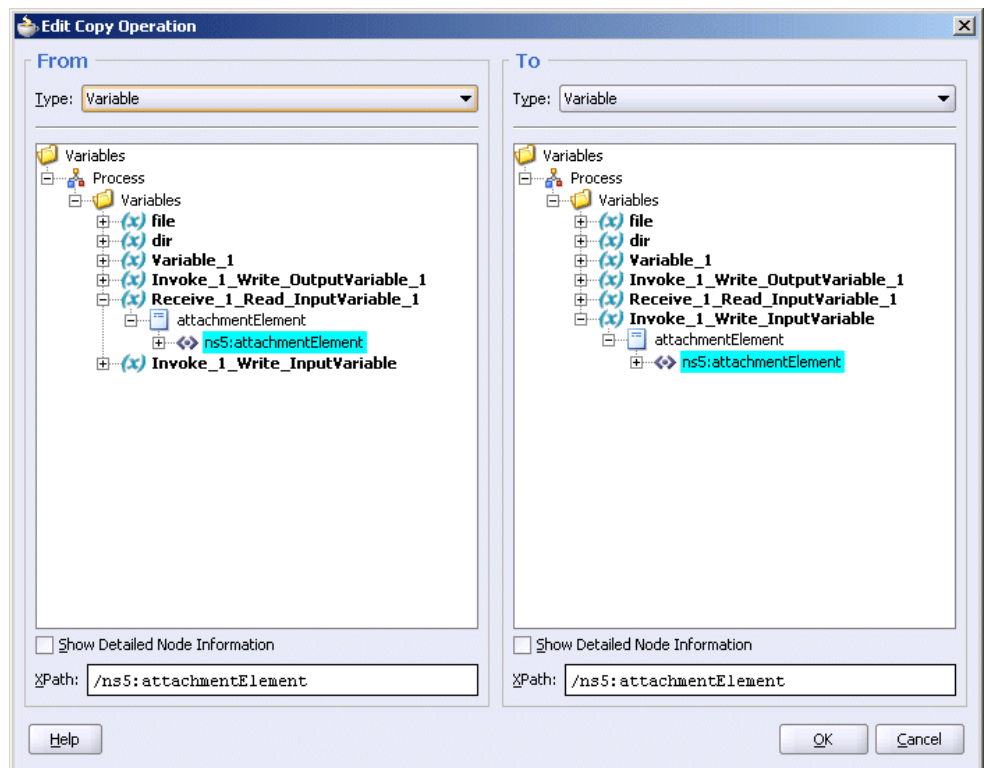
2. Create the inbound file adapter partner link using attachment.xsd.



3. Add a new activation parameter to the adapter WSDL file for the inbound partner link: `AsAttachment="true"`.
4. Create the outbound File adapter partner link using the same attachment.xsd file.



5. Add an assign statement to copy the href attribute between the source and target partner links.



Note: If the transfer operation fails, perform the following steps:

- Increase the transaction timeout in `transaction-manager.xml` (available in `SOA_ORACLE_HOME\j2ee\MIDDLE_TIER\config`).
 - Increase the timeouts in `orion-ejb-jar.xml` (available in `SOA_ORACLE_HOME\j2ee\MIDDLE_TIER\application-deployments\orabpel\ejb_ob_engine`).
-
-

XML Debatching

This feature is available with Oracle BPEL Process Manager and Oracle Enterprise Service Bus. This feature enables you to debatch large XML files and native files. XML debatching in 10.1.3.3 works directly on XML documents, as opposed to the 10.1.2 debatching capability, which assumed a non-XML (flat file) format. Setting up XML debatching requires you to download and configure the XML pull-parsing library (StAX).

1. Go to <http://jcp.org/aboutJava/communityprocess/final/jsr173/index.html>.
2. Go to the "Reference Implementations" section and choose from the available reference implementations.
3. Download the API (`jsr173_1.0_api.jar`) and RI (`jsr173_1.0_ri.jar`). Note that these JAR files are required only for debatching XML files. Non-XML file debatching does not require these files.
4. Copy both JARs to `SOA_ORACLE_HOME\bpel\lib`.
5. Register both JARs in `server.xml` (available in `SOA_ORACLE_HOME\j2ee\MID_TIER\config`) under the `oracle.bpel.common` shared library.

```
<shared-library name="oracle.bpel.common" version="10.1.3">
    . . .
    . . .
    <code-source path="C:\product\bpel\lib\jsr173_1.0_api.jar"/>
    <code-source path="C:\product\bpel\lib\jsr173_1.0_ri.jar"/>
    . . .
    . . .
</shared-library>
```

6. Turn on debatching in the inbound File/FTP adapter partner link by setting the `PublishSize` parameter.

```
<jca:operation
  PhysicalDirectory=" \in"
  ActivationSpec="oracle.tip.adapter.file.inbound.FileActivationSpec"
  . . .
  . . .
  PublishSize="1"
  . . .
  . . .
>
</jca:operation>
```

Note: If the downloaded reference implementation shows a single file (for example, jsr173.zip), extract the API (jsr173_1.0_api.jar) and RI (jsr173_1.0_ri.jar) from the file.

Reading File Names from an Outbound Operation in Oracle BPEL Process Manager

You can read file names and directory names after an outbound interaction.

The following example shows how to read the name of the file that was created after the write operation.

1. Edit the process WSDL file for the outbound File/FTP adapter to receive the file name after the interaction. Add an output message of type outbound header to the portType. Remember that the namespace prefix (hdr) is already defined in the WSDL file.

```
<portType name="Write_ptt">
  <operation name="Write">
    <input message="tns:PurchaseOrder_msg"/>
    <output message="hdr:OutboundHeader_msg"/>
  </operation>
</portType>
```

2. Manually create a variable of type outbound header in the BPEL file; the namespace may vary.

```
<variable name="Invoke_1_Write_OutputVariable"
  messageType="ns3:OutboundHeader_msg"/>
```

3. Use this variable as the outbound variable in the invoke activity.

```
<invoke name="Invoke_1" partnerLink="FileOut"
  portType="ns2:Write_ptt" operation="Write"
  inputVariable="Invoke_1_Write_InputVariable"
  outputVariable="Invoke_1_Write_OutputVariable"/>
```

Controlling the Size of a Rejected Message

You can now control the size of a rejected message by specifying the following endpoint property for the inbound File/FTP adapter partner link.

In this example, you reject 100 lines from the file since the actual file is too large.

```
oracle.tip.adapter.file.debatching.rejection.quantum="100"
```

The acceptable values are 0, EOF, and any non-negative number.

Note: If the endpoint property is not specified, the entire message file (or what is available of the message file) is rejected.

Specifying Unique File Names When Using Time Pattern as Part of the Naming Convention for Outbound Partner Links

When you use the time-pattern as part of the file naming convention for outbound partner links, you lose messages. This is because messages created with the same time stamp overwrite one another.

You can now avoid this problem by mixing file naming conventions. For example, you can specify `%yyMMddHHmmssSSz%__%SEQ%_OrderBookings.xml`. This ensures that the file names are unique.

Note: This works only for nonbatching cases (for example, `NumberMessages="1"`)

MQSeries Adapter New Features

This section describes new features for the MQSeries adapter.

This section contains the following topics:

- [MQSeries Adapter Deprecated Features](#)
- [Outbound Synchronous-Solicit-Request-Response Scenario](#)
- [Outbound Dequeue Scenario](#)
- [Properties for Inbound Messaging](#)
- [Changes to Activation and InteractionSpec Property Names](#)
- [Polling Multiple Queues in a BPEL Process](#)
- [Enabling the Binding Mode for Connections](#)

MQSeries Adapter Deprecated Features

- Support for the polling mode for message retrieval in the inbound adapter direction is deprecated.
- The blocking dequeue mode is now used for getting messages from queues. During blocking dequeue mode, the adapter waits for the message to arrive in a queue rather than polling the queue. This feature provides performance benefits.
- Message filtering based on priority is not a natively-supported feature of the MQSeries itself; therefore, adapter feature provisioning for this feature is deprecated. If you are migrating projects from 10.1.3.1 to 10.1.3.3, ensure that the property `FilteredByPriority` is removed from the `activationSpec` in the inbound adapter direction.

Outbound Synchronous-Solicit-Request-Response Scenario

The 10.1.3.3 MQSeries adapter supports the outbound synchronous-solicit-request-response scenario through use of several `InteractionSpec` properties. In this scenario, the adapter enqueues a normal/request message in a queue and expects the report/reply synchronously. The report/reply message arrives in the `replyToQueueName` of the normal/request message.

InteractionSpec properties for the outbound synchronous-solicit-request-response scenario are as follows:

- **SyncSolicitReqRes** – the permitted value for this property is `true` or `false`. Set this property to `true` for the synchronous-solicit-request-response scenario. This is a mandatory property if you are using the synchronous-solicit-request-response scenario.
- **ResponseWaitInterval** – the permitted value for this property is any interval value (≥ 0). This is the time in milliseconds during which the adapter waits for the report/reply to arrive in `replyToQueueName`. By default, the value of this property is 0 milliseconds. You can change this value, but the value must be less than that of the timeout for the outbound activity. If the report/reply message does not arrive in the stipulated time, the adapter throws an exception. This property is not mandatory.
- **ResponseOpaqueSchema** – the permitted value for this property is `true` or `false`. Set this property to `true` if the report/reply message has an opaque payload. This property is not mandatory. The default value for this property is `false`.

The following examples describe this feature:

Example 1: Enqueue a Normal Message and Dequeue a Message

This example shows how to enqueue a normal message and dequeue a COA report message. The report must arrive within 5000 milliseconds and the report message has an opaque payload. A portion of the WSDL file is shown below:

```
<jca:operation
    InteractionSpec="oracle.tip.adapter.mq.outbound.InteractionSpecImpl"
    QueueName="Request_Queue"
    MessageType="NORMAL"
    MessageFormat="NONE"
    Priority="AS_Q_DEF"
    Persistence="AS_Q_DEF"
    Expiry="NEVER"
    ReportCOA="WITH_FULL_DATA"
    ReplyToQueueName="Report_Queue"
    OpaqueSchema="false"
    SyncSolicitReqRes="true"
    ResponseWaitInterval="5000"
    ResponseOpaqueSchema="true"
>
</jca:operation>
```

Example 2: Enqueue a Request Message and Dequeue a Reply Message

This example shows how to enqueue a request message and dequeue a reply message in an interval of 5000 milliseconds. The response message has a schema associated with it. A portion of the WSDL file is shown below:

```
<jca:operation
    InteractionSpec="oracle.tip.adapter.mq.outbound.InteractionSpecImpl"
    QueueName="Request_Queue"
    MessageType="REQUEST"
    MessageFormat="NONE"
    Priority="AS_Q_DEF"
    Persistence="AS_Q_DEF"
    Expiry="NEVER"
```

```

        ReportCOA="WITH_FULL_DATA"
        ReplyToQueueName="Reply_Queue"
        OpaqueSchema="false"
        SyncSolicitReqRes="true"
        ResponseWaitInterval="5000"
    >
</jca:operation>

```

Note: The ResponseWaitInterval value *must* be less than the timeout for the outbound activity. If the ResponseWaitInterval value exceeds the outbound activity timeout, the adapter can behave ambiguously.

Outbound Dequeue Scenario

The outbound dequeue scenario dequeues a single message from a queue using the outbound MQSeries adapter through use of several InteractionSpec properties. The outbound dequeue scenario supports various filter options. The supported filter options are messageId, correlationId, groupId, and a combination of messageId and correlationId. Filter options can be specified only through headers.

InteractionSpec properties for the outbound dequeue scenario are as follows:

- **InteractionSpec** – This is the InteractionSpec class used for an outbound dequeue. The value for this property must be `oracle.tip.adapter.mq.outbound.SyncInteractionSpecImpl`. This property is mandatory.
- **QueueName** – This is the name of the MQSeries queue from which the message is dequeued. This property is mandatory.
- **WaitInterval** – This is the time (in milliseconds) that the adapter waits if the message is not in the queue. The default value for this property is 0 milliseconds. This property is not mandatory. The permitted value for this property is any integer value (≥ 0). Note that the value of this property must be less than that of the timeout for the outbound activity.
- **FilterByMsgId** – This property sets the message filter option based on the messageId. This property is not mandatory. The value provided for this property must be a hexadecimal-encoded value for some messageId.
- **FilterByCorrelId** – This property sets the message filter option based on the correlationId. This property is not mandatory. The value provided for this property must be a hexadecimal-encoded value for some correlationId.
- **FilterByGroupId** – This property sets the message filter option based on groupId. This property is not mandatory. The value provided for this property must be a hexadecimal-encoded value for some groupId.

The following example shows how to dequeue a message from an OutboundDequeue_Queue using the outbound adapter direction. If the message is not in the queue, wait for a maximum of 1000 milliseconds. A portion of the WSDL file is shown below:

```
<jca:operation
```

```

        InteractionSpec="oracle.tip.adapter.mq.outbound.SyncInteractionSpecImpl"
        QueueName="OutboundDequeue_Queue"
        WaitInterval="1000"
    >
</jca:operation>

```

Properties for Inbound Messaging

In addition to being used for outbound messaging as described in "[Outbound Dequeue Scenario](#)" on page 20, the `FilterByGroupId`, `FilterByMsgId`, and `FilterByCorrelId` properties can also be used for inbound messaging.

Changes to Activation and InteractionSpec Property Names

[Table 1](#) shows the activation and `InteractionSpec` property names changes.

Table 1 *Property Name Changes*

Old Name	New Name
<code>SegmentIfReqd</code>	<code>SegmentIfRequired</code>
<code>BlockingInterval</code>	<code>WaitInterval</code>

Polling Multiple Queues in a BPEL Process

Note: This feature is available as a preview release only and has not been certified.

You can specify multiple `activationAgent` properties in `bpel.xml` to poll more than one queue. The endpoint property name to specify a queue in the inbound direction is `adapter.mq.inbound.queueName`, as shown in the following example.

```

<activationAgents>
<activationAgent
  className="oracle.tip.adapter.fw.agent.jca.JCAActivationAgent"
  partnerLink="inboundService">
    <property name="portType">Dequeue_ptt</property>
    <property name="adapter.mq.inbound.queueName">Queue1</property>
  </activationAgent>
<activationAgent
  className="oracle.tip.adapter.fw.agent.jca.JCAActivationAgent"
  partnerLink="inboundService">
    <property name="portType">Dequeue_ptt</property>
    <property name="adapter.mq.inbound.queueName">Queue2</property>
  </activationAgent>
</activationAgents>

```

Enabling the Binding Mode for Connections

Enable the binding mode for connections by modifying the `oc4j-ra.xml` file for the MQSeries adapter. The following changes are required in the `oc4j-ra.xml` file:

```

hostName - ""

```

portNumber - Any interger value (>1023 & <65356)
channelName - ""
queueManagerName - "Valid_Queue_Manager_Name"

Database Adapter New Features

This section describes new features for the database adapter.

This section contains the following topics:

- [XMLType Support](#)
- [Proxy Authentication Support](#)
- [Inbound Schema Validation](#)

XMLType Support

Pure SQL has been enhanced to support XMLType columns. Because storing XML in CLOB columns was already supported (see the *SOA_ORACLE_HOME\bpel\tutorials\122.DBAdapter\advanced\dmlInvoke\InsertWithClobs* sample), XMLType support was designed to provide the following benefits:

- Enable you to write XQuery expressions in SQL (to query for and update XML records through XPath expressions)
- Import the XSDs of schema-bound XMLTypes from Oracle into your Oracle JDeveloper project process. Importing XMLType columns as `xs:any` is not helpful.
- Support XMLType tables, not just XMLType columns of normal tables

The following SQL statement and autogenerated XSD demonstrates support for these three enhancements:

```
SELECT * FROM MOVIES_XMLTYPE WHERE EXTRACT_VALUE(object_value,  
'/Movies/title') = ?
```

```
<?xml version = '1.0' encoding = 'UTF-8'?>  
<xs:schema targetNamespace="http://xmlns.oracle.com/pcbpel/  
adapter/db/XMLTypeSelectService"  
xmlns="http://xmlns.oracle.com/pcbpel/adapter/db/XMLTypeSelectService"  
xmlns:ns1="http://xmlns.oracle.com/pcbpel/adapter/db/xdb/Movies"  
elementFormDefault="qualified"  
attributeFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  <xs:import  
    namespace="http://xmlns.oracle.com/pcbpel/adapter/db/xdb/Movies"  
    schemaLocation="Movies_xmltype.xsd"/>  
  <xs:element name="XMLTypeSelectServiceInput"  
    type="XMLTypeSelectServiceInput"/>  
  <xs:complexType name="XMLTypeUpdateServiceInput">  
    <xs:sequence>  
      <xs:element name="XML" nillable="true">  
        <xs:complexType>  
          <xs:sequence>  
            <xs:element ref="ns1:Movies" minOccurs="0"/>  
          </xs:sequence>  
        </xs:complexType>  
      </xs:element>  
      <xs:element name="TITLE" type="xs:string" nillable="true"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:schema>
```

```

        </xs:sequence>
    </xs:complexType>

    <xs:element name="XMLTypeSelectServiceOutputCollection"
type="XMLTypeSelectServiceOutputCollection"/>
    <xs:complexType name="XMLTypeSelectServiceOutputCollection">
        <xs:sequence>
            <xs:element name="XML" nillable="true">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="ns1:Movies" minOccurs="0"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:schema>

```

See the new *SOA_ORACLE_*

HOME\samples\tutorials\122.DBAdapter\PureSQLTutorial.txt file, which includes notes about XMLType support. Anonymous XMLTypes are still supported for stored procedures. XMLType support was not added for the **Perform an Operation on a Table** field of the Operation Type page of the Adapter Configuration Wizard. XMLType support adds a new use case to the database adapter, which conceptually takes an existing relational schema as a starting point. If you have an XML schema, but not a relational schema, you can still store and query that XML in an Oracle database. Oracle XDB can remove the XSD to provide performant XML persistence.

Proxy Authentication Support

You can use proxy authentication to connect to an Oracle data store. On a per invoke basis, you can use a combination of the following new header properties:

- proxyUserName
- proxyPassword
- proxyRoles
- proxyCertificate (base64Binary)
- proxyDistinguishedName
- proxyIsThickDriver

For the execution of the invoke, a proxy connection is obtained from the data source. Note that proxyIsThickDriver must be configured and set to true if using the OCI driver, because there is a discrepancy in the JDBC-level API between thick and thin drivers.

See Also: The following URL for additional details about the underlying Oracle TopLink feature:

http://www.oracle.com/technology/products/ias/toplink/doc/1013/main/_html/dblgcfg008.htm

Inbound Schema Validation

You can configure a new activation spec property named `SchemaValidation` in the WSDL file. When set to `true`, all XML produced by the polling database adapter (for receives) is validated against the XSD. If a failure occurs, the XML record is rejected, but still marked as processed by the database adapter. Prior to 10.1.3.3, there was no concept of malformed XML from the database adapter, so message rejection was not implemented. Databases provide structured storage and the XSD is generated by the Adapter Configuration Wizard. However, if you edit the autogenerated XSD and add your own restrictions, you can enable validation. For example, if you import a `VARCHAR(50)` field, the autogenerated XSD is restricted to a maximum length of 50. However, if your BPEL process can only handle values of a fixed length of 22, the XML may be validated.

AQ Adapter New Features

This section describes new features for the AQ adapter:

- [Inbound Schema Validation](#)
- [Using Multiple Activation Agents in the Inbound Direction](#)

Inbound Schema Validation when Using ADT Queues

When using the AQ adapter with ADT (object) queues, no inbound XML validation occurs against the schema generated during design time.

To enable inbound XML validation, add the following Boolean property to the partner link WSDL file performing the dequeue operation:

`SchemaValidation`

This Boolean activation spec property value is set to `false` by default. If set to `true`, inbound XML is validated against the design-time schema during runtime.

Note that the Adapter Configuration Wizard does *not* support this property. To enable this property, you must manually edit the partner link WSDL file.

Using Multiple Activation Agents in the Inbound Direction

If you use a large number of activation agents in the inbound direction, set the `useDefaultConnectionManager` property to `true` for the inbound JNDI entry in the `oc4j-ra.xml` file. If you use an outbound AQ adapter service along with the inbound adapter AQ service with multiple activation agents in the inbound direction, ensure that you do the following:

- Use a different JNDI entry in `oc4j-ra.xml`.
- Set `useDefaultConnectionManager` to `false`.

JMS Adapter New Features

This section describes new features for the JMS adapter.

Support for BEA WebLogic JMS Provider Queues and Topics

The JMS adapter provides support for BEA WebLogic JMS provider queues and topics. To use this feature, set `environment-naming-url-factory-enabled` to `true` in the `server.xml` file.

Oracle ESB Singleton Behavior for Inbound Adapter Endpoints

When using Oracle ESB in a clustered (OC4J) environment, inbound nontransaction JCA resource adapters incur race conditions when consuming messages from the endpoint (for example, an inbound file adapter).

To avoid this problem, add the following ESB endpoint property to the service definition:

```
clusterGroupId
```

This endpoint property can be assigned any value. At runtime, only one node in the cluster owns the endpoint activation.

Decision Service Support for Ilog JRules

The decision service automatically supports integration with Ilog JRules. However, several configuration steps must be manually performed. These steps are described in this section.

Prerequisites

These instructions assume that the Ilog JRules version 6.1 Rule Execution Server (RES) is installed and the RES management application (EAR file) is deployed to Oracle SOA Suite 10.1.3.3. Throughout these instructions, the following environment variables point to directories of installed components:

- `JRULES_HOME` — The directory in which Ilog JRules 6.1 is installed
- `ORACLE_HOME` — The directory in which Oracle SOA Suite is installed
- `JDEV_HOME` — The directory in which Oracle JDeveloper is installed

Oracle JDeveloper Setup

To connect to the Ilog JRules RES from the Decision Service wizard, copy the following JAR file from the Ilog JRules installation directory to the Oracle JDeveloper installation directory.

1. Exit Oracle JDeveloper.
2. Perform the following steps at the operating system command prompt:

```
cd ${JDEV_HOME}/integration
mkdir thirdparty/ilog/lib
cp ${JRULES_HOME}/executionserver/lib/jrules-bres-session-java.jar \
${JDEV_HOME}/integration/thirdparty/ilog/lib
```

3. Start Oracle JDeveloper.

SOA Suite setup

The Oracle Application Server instance hosting Oracle BPEL Process Manager must be configured for the decision service runtime to use Ilog JRules.

1. Stop Oracle SOA Suite.

```
${ORACLE_HOME}/opmn/bin/opmnctl stopall
```

2. Open the `server.xml` file in the `${ORACLE_HOME}/j2ee/oc4j_soa/config` directory.

3. Search for shared library `oracle.bpel.common` and add the following JAR files.

```
<code-source path="${JRULES_
HOME}/executionserver/lib/jrules-bres-execution.jar"/>
<code-source path="${JRULES_
HOME}/executionserver/lib/jrules-bres-manage-tools.jar"/>
<code-source path="${JRULES_
HOME}/executionserver/lib/jrules-bres-session-java.jar"/>
<code-source path="${JRULES_
HOME}/executionserver/lib/jrules-engine.jar"/>
<code-source path="${JRULES_
HOME}/executionserver/lib/commons-digester.jar"/>
<code-source path="${JRULES_
HOME}/executionserver/lib/commons-logging.jar"/>
<code-source path="${JRULES_HOME}/executionserver/lib/sam.jar"/>
<code-source path="${JRULES_HOME}/executionserver/lib/xercesImpl.jar"/>
```

4. Start Oracle SOA Suite.

```
${ORACLE_HOME}/opmn/bin/opmnctl startall
```

5. Open the `${ORACLE_HOME}/bpel/system/services/config/DecisionServiceConfiguration.xml` file.

6. Verify that the decision service is configured to use Ilog JRules by ensuring that the following entries in bold exist:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<!--
  Configuration file for the decision service.
-->
<configuration xmlns="http://xmlns.oracle.com/bpel/rules">
  <!--
    Rule Engine provider implementations
  -->
  <ruleEngine name="Oracle" description="Oracle Business Rules
10.1.3.1.0">
    <ruleEngineClass>
      oracle.bpel.services.rules.rpi.oracle.OracleRuleEngine
    </ruleEngineClass>
  </ruleEngine>

  <ruleEngine name="Ilog" description="Ilog JRules 6.1">
    <ruleEngineClass>
      oracle.bpel.services.rules.rpi.ilog.IlogRuleEngine
    </ruleEngineClass>
    <properties>
```

```

        <property name="jndiXuConnectorName">
            eis/XUConnectionFactory
        </property>
        <property name="ruleSessionProviderClass">
            ilog.rules.bres.session.j2se.IlrJ2SERuleSessionProvider
        </property>
    </properties>
</ruleEngine>

<!--
    Fact context implementations
-->
<factContext name="OracleJaxb" description="Oracle JAXB 1.0">
    <factContextClass>
        oracle.bpel.services.rules.rpi.JAXBFactContext
    </factContextClass>
</factContext>

<factContext name="XOM" description="Ilog JRules 6.1 XOM">
    <factContextClass>
        oracle.bpel.services.rules.rpi.ilog.XOMFactContext
    </factContextClass>
</factContext>

</configuration>

```

The following properties can be configured depending on the rule execution server connectivity setup:

- **jndiXuConnection** — The JNDI name of the Ilog JRules Execution Unit resource adapter (XU) connection factory. The default setting is `eis/XUConnectionFactory`.
- **ruleSessionProviderClass** — The fully-qualified Ilog JRules Java class name that establishes a rule session with the Ilog JRules RES. The default setting is `ilog.rules.bres.session.j2se.IlrJ2SERuleSessionProvider`. See the Ilog JRules API reference for other settings. You may need to specify additional properties. For example, if the rule session class is `ilog.rules.bres.session.ejb.IlrManagedRuleSessionProvider` or `ilog.rules.bres.session.ejb.IlrRemoteRuleSessionProvider`, then appropriate JNDI properties must be configured to initialize the JNDI InitialContext. Those properties can be added to the `properties` element of the Ilog JRules rule engine configuration in the `DecisionServiceConfiguration.xml` file.

Example 1 *DecisionServiceConfiguration.xml* Property Settings

```

<ruleEngine name="Ilog" description="Ilog JRules 6.1">
    <ruleEngineClass>
        oracle.bpel.services.rules.rpi.ilog.IlogRuleEngine
    </ruleEngineClass>
    <properties>
        <property name="jndiXuConnectorName">
            eis/XUConnectionFactory
        </property>
        <property name="ruleSessionProviderClass">
            ilog.rules.bres.session.ejb.IlrRemoteRuleSessionProvider

```

```

</property>
<property name="java.naming.factory.initial">
    com.evermind.server.rmi.RMIInitialContextFactory
</property>
<property name="java.naming.provider.url">
    ormi://localhost:23791/rulesession
</property>
<property name="java.naming.security.principal">
    oc4jadmin
</property>
<property name="java.naming.security.credentials">
    welcome1
</property>
</properties>
</ruleEngine>

```

The properties for the Ilog JRules decision service configuration in the `DecisionServiceConfiguration.xml` file denote default values. However, the values can be overwritten by a specific decision service partner link configuration. To do this, modify the `ruleEngineProvider` element in the decision service configuration file `decisionservices.decs` of the BPEL project with the following properties shown in bold:

```

<ruleEngineProvider provider="Ilog" name="jrules61_myhost">
  <repository type="Service">
    <service>
      <url>service:jmx:rmi:///opmn://myhost.com:6003/home</url>
      <username>oc4jadmin</username>
      <password encrypted="true">AK6qvYcrlNMqnYtluPZFjw==</password>
    </service>
  </repository>

  <properties>
    <property name="jmxConnectorClass">
      oracle.bpel.services.rules.rpi.ilog.OracleJmxConnector
    </property>
    <property name="ruleSessionProviderClass">
      ilog.rules.bres.session.j2se.IlrJ2SERuleSessionProvider
    </property>
  </properties>

</ruleEngineProvider>

```

Limitations

This section describes known limitations on decision service integration with Ilog JRules:

- Parameter type support — The input and output parameters of the rule set you want to expose as a decision service must be derived from an XML-schema element. The decision service does not currently support type models based on JavaBeans or others.
- Deployment of Ilog JRules decision service — Some Ilog JRules specific deployment artifacts must be copied into the Web archive suite of the decision service prior to deployment. This may include configuration files for the JRules XU resource adapter (`ra.xml`) or any other Ilog specific file. See the Ilog JRules documentation for details.

Fault Management Framework

Release 10.1.3.3 provides a generic fault management framework for handling faults in BPEL processes. If a fault occurs during runtime in an invoke activity in a process, the framework catches the fault and performs a user-specified action defined in a fault policy file associated with the activity. If a fault results in a condition in which human intervention is the prescribed action, you perform recovery actions from Oracle BPEL Control. The fault management framework provides an alternative to designing a BPEL process with catch branches in scope activities.

This section contains the following topics:

- [Fault Management Framework Overview](#)
- [Designing a Fault Policy](#)
- [Fault Management Framework Use Case](#)
- [Java Action Fault Policy](#)

See Also: The following fault management framework sample:

```
SOA_ORACLE_  
HOME\bpel\samples\tutorials\122.DBAdapter\InsertWithCatch
```

Fault Management Framework Overview

This section provides an overview of the components that comprise the fault management framework.

- A schema named `fault-policy.xsd` is provided for defining a fault management policy.
- The fault management framework catches all faults (business and runtime) for an invoke activity.
- A fault policy file defines fault conditions and their corresponding fault recovery actions. Each fault condition specifies a particular fault or group of faults, which it attempts to handle, and the corresponding action for it. A set of actions is identified by an ID in the fault policy file.
- A set of conditions invokes an action (known as fault policy).
- Fault policies are defined in multiple files under a directory named `fault-policies` for each domain:

```
SOA_ORACLE_HOME\bpel\domains\domain_name\config\fault-policies\
```

where *domain_name* is the name of the domain (for example, `default` or any additional domains that you have created).

- A fault policy can be associated at the following levels:
 - Partner link
 - Port type
 - Process
 - Domain

- A fault policy bindings file associates the policies defined in a fault policy file with partner links, port types, processes, and domains. The framework looks for fault policy bindings in the following files (in order of priority):
 - In the `bpel.xml` file at the process level
 - In the domain level file:

`SOA_ORACLE_HOME\bpel\domains\domain_name\config\fault-bindings.xml`

Note: A fault policy configured with the fault management framework overrides any fault handling defined in catch branches of scope activities in the BPEL process. The fault management framework can be configured to rethrow the fault handling back to the catch branches.

Designing a Fault Policy

This section describes how to design a fault policy.

This section contains the following topics.

- [Understanding How Fault Policy Binding Resolution Works](#)
- [Task 1: Create a Fault Policy File for Automated Fault Recovery](#)
- [Task 2: Associate a Fault Policy](#)

Understanding How Fault Policy Binding Resolution Works

A fault policy bindings file associates the policies defined in a fault policy file with partner links, port types, processes, and domains. The framework attempts to identify a fault policy binding in the following order:

- Partner link binding in `bpel.xml`
- Port type binding in `bpel.xml`
- Process binding in `bpel.xml`
- Partner link binding specified for the domain
- Port type binding specified for the domain
- Process binding specified for the domain

During the resolution process, if no action is found that matches the condition, the framework assumes that resolution failed and moves to the next resolution level.

For example, assume an invoke activity faults with `faultname="abc"`. There is a policy binding specified in the `bpel.xml` file:

- Partner link name binds to `policy-id-1`
- Port type binds to `policy-id-2`
- Process binds to `policy-id-3`

In the `SOA_ORACLE_HOME\bpel\domains\domain_name\config\fault-bindings.xml` file, the following bindings are also specified:

- Partner link name binds to `policy-id-4`

- Port type binds to `policy-id-5`
- Process binds to `policy-id-6`

The fault management framework behaves as follows:

- First match the resolve binding (in this case, `policy-id-1`).
- If the fault resolution fails, go to the next possible match (in this case, `policy-id-2`).
- If the fault resolution fails, go to the next possible matches (`policy-id-3`, `policy-id-4`, `policy-id-5`, and `policy-id-6`).
- If the fault resolution still fails, the fault is sent to the BPEL fault catch branch.

Task 1: Create a Fault Policy File for Automated Fault Recovery

1. Create directories named `fault-policies` under the `config` directory for every domain in which you want to use the fault management framework.

```
SOA_ORACLE_HOME\bpel\domains\domain_name\config\fault-policies
```

2. Create a fault policy file (for example, named `fault-policy.xml`) in the `fault-policies` directory. This file includes `condition` and `action` sections for performing specific tasks.
3. Define the `condition` section of the fault policy file.
 - Note the following details about the `condition` section:
 - This section provides a condition based on `faultName`.
 - Multiple conditions may be configured for a `faultName`.
 - Each condition has one `test` section (an XPath expression) and one `action` section.
 - The `test` section (XPath expression) is evaluated for the fault variable available in the fault.
 - The `action` section has a reference to the action defined in the same file.
 - You can only query the fault variable available in the fault.
 - The order of condition evaluation is determined by the sequential order in the document.

The following table provides examples of `condition` section use in the fault policy file. All actions defined in the `condition` section must be associated with an action in the `action` section.

Condition Example	Fault Policy File Syntax
This condition is checking a fault variable for code = "WSDLFailure"	<pre><condition> <test>\$fault.code/code="WSDLReading Error" </test></pre>
An action of <code>ora-terminate</code> is specified.	<pre> <action ref="ora-terminate"/> </condition></pre>

Condition Example	Fault Policy File Syntax
No test condition is provided. This is a catch all condition for a given faultName.	<pre><condition> <action ref="ora-rethrow"/> </condition></pre>
If the faultName name attribute is missing, this indicates a catch all branch for faults that have any QName.	<pre><faultName > . . . </faultName></pre>

4. Define the action section of the fault policy file.

- Note the following details about the action section:
 - The list of actions cannot be extended.
 - Validation of fault policy files is done at the domain startup time. Oracle BPEL Server must be restarted for new handlers to be effective.

The following table provides several examples of action section use in the fault policy file. You can provide automated recovery actions for some faults. In all recovery actions except retry and human intervention, the framework performs the actions synchronously.

Recovery Actions	Fault Policy File Syntax
Retry — Provides the following actions for retrying the activity. <ul style="list-style-type: none"> ■ Retry a specified number of times ■ Provide a delay between retries ■ Provide an exponential backoff ■ Provide a retry failure action ■ Provide a retry success action <p>Note: Exponential backoff indicates the next retry attempt is scheduled at 2 x the <i>delay</i>, where <i>delay</i> is the current retry interval. For example, if the current retry interval is 2 seconds, the next retry attempt is scheduled at 4, the next at 8, and the next at 16 seconds until the <code>retryCount</code> value is reached.</p>	<pre><Action id="ora-retry"> <Retry> <retryCount>3</retryCount> <retryInterval>2</retryInterval> <exponentialBackoff/> <retryFailureAction ref="ora-java"/> <retrySuccessAction ref="ora-java"/> </Retry> </Action></pre> <p>Note the following details:</p> <ul style="list-style-type: none"> ■ If multiple WSDL locations are available, the Oracle BPEL Server attempts a connection to the next location for <i>n</i> times with delay. The framework attempts to connect to each specified WSDL location at every retry count attempt. ■ The framework chains to the retry success action in the event of the retry attempt being successful. ■ If all retry attempts fail, the framework chains to the retry failure action.

Recovery Actions	Fault Policy File Syntax
Human Intervention — Causes the current activity to stop processing. You can now go to Oracle BPEL Control and take several actions on this instance. See Also: "Human Intervention in Oracle BPEL Control" on page 41	<pre><Action id="ora-human-intervention"> <humanIntervention/></Action></pre>
Terminate Process — Terminates the process	<pre><Action id="ora-terminate"><abort/></Action></pre>
Java Code — Enables you to execute an external Java class. See Also: "Java Action Fault Policy" on page 47	<pre><Action id="ora-java"> <!-- this is user provided custom java class--> <javaAction className="mypackage.myClass" defaultAction="ora-terminate"> <returnValue value="REPLAY" ref="ora-terminate"/> <returnValue value="RETHROW" ref="ora-rethrow-fault"/> <returnValue value="ABORT" ref="ora-terminate"/> <returnValue value="RETRY" ref="ora-retry"/> <returnValue value="MANUAL" ref="ora-human-intervention"/> </javaAction> </Action></pre>
Rethrow Fault — The framework sends the fault to the BPEL fault handlers (catch branches in scope activities). If none are available, the fault is sent up.	<pre><Action id="ora-rethrow-fault"><rethrowFault/></Action></pre>
Replay Scope — Raises a replay fault	<pre><Action id="ora-replay-scope"><replayScope/></Action></pre>

Notes:

- There is no option for creating a generic fault handler extension. The list of fault policy actions is limited by the framework.
 - The preseeded recovery action tag names (ora-retry, ora-human-intervention, ora-terminate, and so on) are only samples. You can substitute these names with ones appropriate to your environment.
-
-

The following example shows a fault policy file with fully defined condition and action sections.

Note: Fault policy file names are not restricted to one specific name. However, the file must conform to the `fault-policy.xsd` schema available in the `SOA_ORACLE_HOME\bpel\system\xml\lib` folder.

```

<?xml version="1.0" encoding="UTF-8"?>
<faultPolicy version="2.0.1" id="CRM_ServiceFaults"
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.oracle.com/bpel/faultpolicy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.oracle.com/bpel/faultpolicy
C:\oc4j\bpel\system\xml\lib\fault-policy.xsd">
<Conditions>
<!-- Fault if wsdlRuntimeLocation is not reachable -->
<faultName xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
name="bpelx:remoteFault">
<condition>
<test>$fault.code/code="WSDLReadingError"</test>
<action ref="ora-terminate"/>
</condition>
<condition>
<action ref="ora-java"/>
</condition>
</faultName>
<!-- Fault if location port is not reachable-->
<faultName xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
name="bpelx:bindingFault">
<!--ORA-00001: unique constraint violated on insert-->
<condition>
<test>$fault.code/code="1"</test>
<action ref="ora-java"/>
</condition>
<!--ORA-01400: cannot insert NULL -->
<condition>
<test>$fault.code/code="1400"</test>
<action ref="ora-terminate"/>
</condition>
<!--ORA-03220: required parameter is NULL or missing -->
<condition>
<test>$fault.code/code="3220"</test>
<action ref="ora-terminate"/>
</condition>
<condition>
<action ref="ora-retry-crm-endpoint"/>
</condition>
</faultName>
<!-- Business faults -->
<!-- Fault comes with a payload of error, make sure the name space-->
<!-- is provided here or at root level -->
<faultName xmlns:credit="http://services.otn.com"
name="credit:NegativeCredit">
<condition>
<test>$fault.payload/credit:error="Bankruptcy Report"</test>
<action ref="ora-retry"/>
</condition>
<condition>
<test>$fault.payload/credit:error="Illegal SSN"</test>
<action ref="ora-terminate"/>
</condition>
</faultName>
</Conditions>
<Actions>
<Action id="ora-retry">

```

```

        <retry>
            <retryCount>3</retryCount>
            <retryInterval>2</retryInterval>
            <exponentialBackoff/>
            <retryFailureAction ref="ora-java"/>
            <retrySuccessAction ref="ora-java"/>
        </retry>
    </Action>
    <Action id="ora-retry-crm-endpoint">
        <retry>
            <retryCount>5</retryCount>
            <retryFailureAction ref="ora-java"/>
            <retryInterval>5</retryInterval>
            <retrySuccessAction ref="ora-java"/>
        </retry>
    </Action>
    <Action id="ora-replay-scope">
        <replayScope/>
    </Action>
    <Action id="ora-rethrow-fault">
        <rethrowFault/>
    </Action>
    <Action id="ora-human-intervention">
        <humanIntervention/>
    </Action>
    <Action id="ora-terminate">
        <abort/>
    </Action>
    <Action id="ora-java">
    <!-- this is user provided class-->
        <javaAction
            className="com.oracle.bpel.client.config.faultpolicy.TestJavaAction"
            defaultAction="ora-terminate" propertySet="prop-for-billing">
                <returnValue value="REPLAY" ref="ora-terminate"/>
                <returnValue value="RETHROW" ref="ora-rethrow-fault"/>
                <returnValue value="ABORT" ref="ora-terminate"/>
                <returnValue value="RETRY" ref="ora-retry"/>
                <returnValue value="MANUAL" ref="ora-human-intervention"/>
            </javaAction>
        </Action>
    </Actions>
    <Properties>
        <propertySet name="prop-for-billing">
            <property name="user_email_recipient">bpeladmin</property>
            <property name="email_recipient">joe@abc.com</property>
            <property name="email_recipient">mike@xyz.com</property>
            <property name="email_threshold">10</property>
            <property name="sms_recipient">+429876547</property>
            <property name="sms_recipient">+4212345</property>
            <property name="sms_threshold">20</property>
            <property name="user_email_recipient">john</property>
        </propertySet>
        <propertySet name="prop-for-order">
            <property name="email_recipient">john@abc.com</property>
            <property name="email_recipient">jill@xyz.com</property>
            <property name="email_threshold">10</property>
            <property name="sms_recipient">+42222</property>
            <property name="sms_recipient">+423335</property>
            <property name="sms_threshold">20</property>
        </propertySet>
    </Properties>

```

```

        </propertySet>
    </Properties>
</faultPolicy>

```

Task 2: Associate a Fault Policy

1. Associate a fault policy with the level of fault policy binding you are using:

- [Task 2a: Associate a Fault Policy with a Partner Link, Port Type, or Process](#)
- [Task 2b: Associate a Fault Policy with a Partner Link, Port Type, or Process at the Domain Level](#)

Task 2a: Associate a Fault Policy with a Partner Link, Port Type, or Process You can associate a specific partner link, port type, or entire process with a fault policy. This method:

- Provides a more granular approach to fault policy binding than "[Task 2b: Associate a Fault Policy with a Partner Link, Port Type, or Process at the Domain Level](#)"
 - Overrides any fault policy binding defined for "[Task 2b: Associate a Fault Policy with a Partner Link, Port Type, or Process at the Domain Level](#)"
1. Add a new section to `bpel.xml` file. Note the following details:

- Only one fault policy can be bound to a process, port type, or partner link.
- Multiple partner links can be bound to a fault policy.
- Multiple port types can be bound to a fault policy.

The following example is provided:

```

<faultPolicyBindings>
  <process faultPolicy="BillingFaults"/>
  <!--Fault on any plink/port type not specified-->
  <!--below uses policy BillingFaults-->
  <partnerLink xmlns:credit="http://services.otn.com" faultPolicy="CRM_
    ServiceFaults">
    <name>UnitedLoanService</name>
    <!--Fault on these 2 plink will use policy CRM_ServiceFaults-->
    <name>StarLoanService</name>
    <portType>credit:CreditRatingService</portType>
    <!--Fault on these 2 port types uses policy CRM_ServiceFaults-->
    <portType xmlns:united="http://services.uninted.com/loan">
      united:UnitedLoanService</portType>
    </portType>
  </partnerLink>

  <partnerLink faultPolicy="myOtherFaults">
    <name>AnotherPartnerLink</name>
    <!--Fault on this plink uses policy myOtherFaults-->
  </partnerLink>
</faultPolicyBindings>

```

Task 2b: Associate a Fault Policy with a Partner Link, Port Type, or Process at the Domain Level You can associate an entire domain with a fault policy.

1. Provide a fault policy applicable to an entire domain in the `fault-bindings.xml` file in the `SOA_ORACLE_HOME\bpel\domains\domain_name\config` directory.

The following example is provided:

```
<faultPolicyBindings version="2.0.1"
xmlns="http://schemas.oracle.com/bpel/faultpolicy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <process faultPolicy="BillingFaults"/>
  <partnerLink faultPolicy="CRM_ServiceFaults">
    <name>StarLoanService</name>
    <name>BillCardService</name>
    <portType xmlns:credit="http://services.otn.com">
      credit:CreditRatingService</portType>
    <portType xmlns:united="http://services.united.com/loan">
      united:UnitedLoanService</portType>
    </partnerLink>
</faultPolicyBindings>
```

A binding provided for a port type in this file is used for all processes in the domain. These processes can use different partner link names. If the port types are the same, the fault policy specified in this file is used.

Note: The fault policy bindings file must be named `fault-bindings.xml`. This name must conform to the schema available in `SOA_ORACLE_HOME\bpel\system\xml\lib\fault-bindings.xsd`.

Fault Management Framework Use Case

This use case describes the design time and runtime phases of a fault management framework used in a BPEL process.

This section contains the following topics:

- [BPEL Process and Fault Policy Design](#)
- [Scenario 1: Automated Fault Recovery](#)
- [Scenario 2: Human Intervention for Fault Recovery](#)
- [Human Intervention in Oracle BPEL Control](#)
- [Fault Logging](#)

BPEL Process and Fault Policy Design

This section provides a use case of how to design a BPEL process and fault policy using the fault management framework.

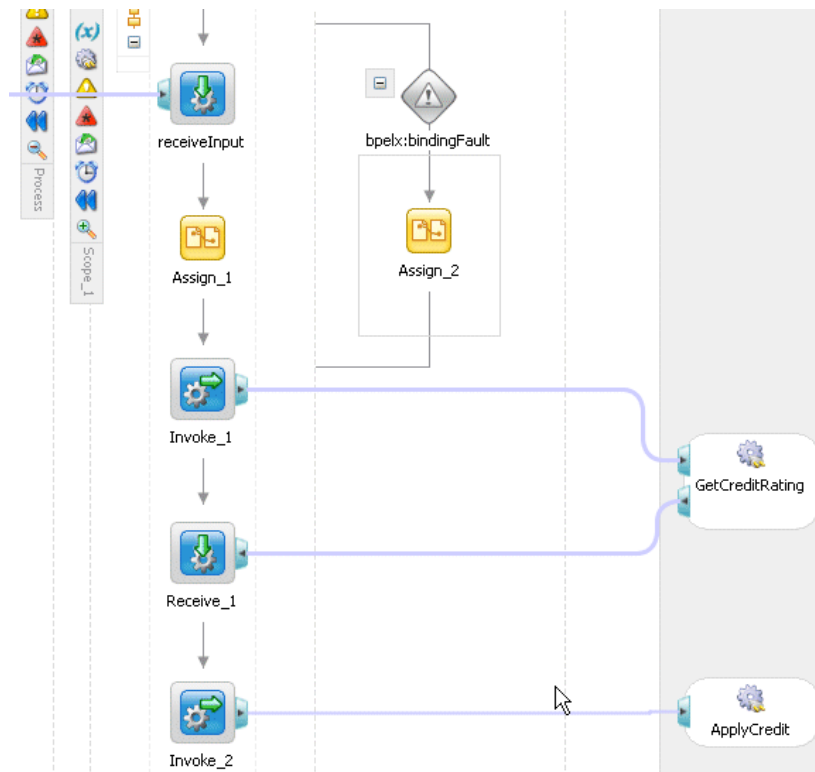
This section contains the following topics:

- [BPEL Process Design](#)
- [Fault Policy Files for the Domain](#)
- [Fault Policy Binding Definitions in the `bpel.xml` File](#)

BPEL Process Design The BPEL process shown in [Figure 1](#) includes the following activities:

- A scope activity named **Scope_1** with a fault handler for **bpelx:bindingFault**.
- Two invoke activities that connect to two partner links.

Figure 1 Scope Activity of BPEL Process



Fault Policy Files for the Domain As shown in Figure 1, there are two partner links invoked by two invoke activities. Fault policy files are created using the fault management framework to handle faults for both invoke activities. The fault policy overrides the fault handling provided in the **bpelx:bindingFault** catch branch in the **Scope_1** scope activity. Example 2 shows the first fault policy file (CreditRating.xml) created in the default domain for the GetCreditRating partner link.

Example 2 SOA_ORACLE_HOME\bpel\domains\default\config\fault-policies\CreditRating.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicy version="2.0.1" id="CreditRating "
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.oracle.com/bpel/faultpolicy"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.oracle.com/bpel/faultpolicy
C:\oc4j\bpel\system\xml\lib\fault-policy.xsd">
  <Conditions>
    <!-- Fault if location port is not reachable-->
    <faultName xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
      name="bpelx:bindingFault">
      <condition>
        <action ref="ora-retry-crm-endpoint"/>
      </condition>
    </faultName>
  </Conditions>
</faultPolicy>
```

```

        </condition>
    </faultName>
</Conditions>
<Actions>
    <Action id="ora-retry">
        <retry>
            <retryCount>3</retryCount>
            <retryInterval>2</retryInterval>
            <exponentialBackoff/>
            <retryFailureAction ref="ora-java"/>
        </retry>
    </Action>
    <Action id="ora-retry-crm-endpoint">
        <retry>
            <retryCount>5</retryCount>
            <retryInterval>5</retryInterval>
            <retryFailureAction ref="ora-java"/>
            <retrySuccessAction ref="ora-java"/>
        </retry>
    </Action>
    <Action id="ora-replay-scope"><replayScope/></Action>
    <Action id="ora-rethrow-fault"><rethrowFault/></Action>
    <Action id="ora-human-intervention">
        <humanIntervention/>
    </Action>
    <Action id="ora-terminate"><abort/></Action>
    <Action id="ora-java">
        <!-- this is user provided class-->
        <javaAction className="myPackage.LogFault"
            defaultAction="ora-terminate">    Note: property set is optional
            <returnValue value="REPLAY" ref="ora-terminate"/>
            <returnValue value="RETHROW" ref="ora-rethrow-fault"/>
            <returnValue value="ABORT" ref="ora-terminate"/>
            <returnValue value="RETRY" ref="ora-retry"/>
            <returnValue value="MANUAL" ref="ora-human-intervention"/>
        </javaAction>
    </Action>
</Actions>
----Note: Properties section is optional----
</faultPolicy>

```

[Example 3](#) shows the second fault policy file (`ApplyCredit.xml`) created in the default domain for the `ApplyCredit` partner link. The contents of the action section in [Example 3](#) are the same as those shown in [Example 2](#).

Example 3 SOA_ORACLE_HOME\bpel\domains\default\config\fault-policies\Apply-Credit.xml

```

<FaultPolicy id="ApplyCredit">
    <!-------Conditions----->
    <Conditions>
        <faultName
            name="{http://schemas.oracle.com/bpel/extension}bindingFault">
            <condition>
                <action ref="ora-human-intervention"/>
            </condition>
        </faultName>
    </Conditions>
</FaultPolicy>

```

```

. . .
. . .
. . .
</Actions>

```

Fault Policy Binding Definitions in the `bpel.xml` File The fault policy binding is defined in the `bpel.xml` file. Note the following details:

- A fault on `CreditRating` causes the framework to use the `CreditRating` fault policy.
- A fault on `ApplyCredit` falls back to the process level binding and causes the framework to use the `ApplyCredit` fault policy.

```

<faultPolicyBindings>
  <process faultPolicy=" ApplyCredit " />
  <!--Process binds to ApplyCredit policy-->
  <partnerLink faultPolicy="CreditRating ">
    <name>creditRatingService</name>
    <!--partner link binds to CreditRating policy-->
    <portType xmlns:credit="http://services.otn.com">
      StarRatingService</portType>
    </partnerLink>
  </faultPolicyBindings>

```

Scenario 1: Automated Fault Recovery

In this scenario, the automated fault recovery process is demonstrated. The `CreditRating` endpoint is unavailable. The following series of actions occur:

- The `Invoke_1` activity raises a fault that is caught by the framework.
- The fault policy resolution process is started.
 - The fault policy `CreditRating` is identified.
- The action resolution process is started:
 - The fault policy has a catch branch for `bpelx:bindingFault`. The `ora-retry-crm-endpoint` action defined in the `CreditRating.xml` file shown in [Example 2](#) on page 38 is identified.
- Get action details:
 - This is a retry action in which the retry count is 5, the retry interval is 5 seconds, the retry failure action is `ora-java`, and the retry success action is `ora-java`.
 - The retry action is an asynchronous operation.
- Dehydrate the activity and other policy-related data.
- The fault management framework attempts a retry.
- All retry attempts fail.
- The `RetryFailureAction` attribute is `ora-java`.
- The steps from the action resolution process are repeated.
- Since `ora-java` is a synchronous operation, no dehydration is performed and the Java code provided in `myPackage.LogFault` is invoked.
- The Java code logs the string and returns a `RETHROW`. This causes chaining to `ora-rethrow-fault`.

- The Scope_1 fault handler for the bindingFault defined in the BPEL process catches the exception.

If the retry attempt is successful, the framework chains to ora-java, but invokes a different method on this object.

Note: The framework handles the exception even if the BPEL process has a fault handler.

Scenario 2: Human Intervention for Fault Recovery

In this scenario, human intervention is demonstrated. The ApplyCredit service endpoint receives a business fault. The following series of actions occur:

- The Invoke_2 activity raises a fault that is caught by the framework.
- The fault policy resolution process is started.
 - The partner link binding resolves to no policy; resolution of process level binding is attempted.
 - The ApplyCredit fault policy is identified.
- The action resolution process is started.
 - Action ora-human-intervention defined in the ApplyCredit.xml file shown in [Example 3](#) on page 39 is identified.
- Get action details:
 - This is a pause action; only the activity is put in a pause status.
 - A pause action (human intervention) is an asynchronous operation
- Dehydrate the activity and other policy-related data.
- All other activities in the process are not affected.
- Go to Oracle BPEL Control and take the next action.

Human Intervention in Oracle BPEL Control

1. Go to the faulted instance in the Oracle BPEL Control.
2. Click the **Activities** tab.

The **Activity State** list enables you to display activities based on their current state. For this example, only activities currently pending are displayed.

Dashboard
BPEL Processes
Instances
Activities

Locate Activities

Instance Id#

Activity Label

Fault Name

Index

Activity State
Pending
BPEL Process
All Processes
Last Modified
All Times
Go

Activities 1 - 3

	Activity Label	Instance	Process	Due	Fault Name
<input type="checkbox"/>	invokeCR	10018	LoanFlow (v. 1.0)	9/17/07 2:29:53 PM	NegativeCre
<input type="checkbox"/>	invokeCR	10013	LoanFlow (v. 1.0)	9/17/07 2:24:44 PM	NegativeCre
<input type="checkbox"/>	invokeCR	10008	LoanFlow (v. 1.0)	9/17/07 12:24:55 PM	NegativeCre

Check All - Clear All

Actions available:
Retry
Recover

State	Description
All States	Displays all activities, regardless of their state.
Open	Displays only open activities.
Completed	Displays only completed activities.
Cancelled	Displays only cancelled activities.
Stale	Displays only stale activities.
Pending	Displays only pending activities.

3. Click the faulted activity.

The **Available Actions** list displays a set of available recovery actions.

Instance: [10018 - Instance #10018 of LoanFlow](#)
Label: invokeCR
BPEL Process: [LoanFlow \(v. 1.0\)](#)

Due Date: 9/17/07
State: open.pe
Priority: 3

Available Variables:
Get Set Skeleton Value

Value:

Actions available:

Retry
Abort
Rethrow
Replay
Continue

New Instance
tails

On Retry Success Chain to this action: None

Once an activity is marked for recovery through human intervention, the recovery actions described in the following table are possible.

Recovery Action	Description
Retry	Retries the activity with an option to provide a retry success action.
Abort	Terminates the process instance of the faulted activity.

Recovery Action	Description
Rethrow	Rethrows the exception and allows the BPEL fault handlers (catch branches) to handle the fault. By default, all exceptions are caught by the fault management framework unless an explicit rethrow fault policy is provided. See Also: "Rethrow the Exception" on page 45
Replay	Replays the scope in which the fault occurred.
Continue	Skips the activity. The framework assumes the activity completed with no fault.

4. Expand **Fault Details** to view details about the faulted instance.

Actions available: Retry On Retry Success Chain to this action: None

Recover New Instance

☒ Fault Details

Fault Name	NegativeCredit	Fault Message Type	CreditRatingServiceFaultMessage
Creation Date	9/17/07 2:29:54 PM	Modify Date	9/17/07 2:29:54 PM
Fault Message	<pre> - <NEGATIVECREDIT xmlns="http://services.otn.com"> - <PART name="payload"> <ERROR xmlns="http://services.otn.com">Bankruptcy Report</ERROR> </PART> </NEGATIVECREDIT> </pre>		

Note: The Oracle BPEL Process Manager API enables you to programmatically perform the abort, retry (with a success action), continue, rethrow, and replay recovery options.

5. See the following sections to use these recovery actions:

- [Retry the Activity](#)
- [Change the Input Variable Contents and Retry](#)
- [Set the Output and Continue](#)
- [Replay the Scope](#)
- [Rethrow the Exception](#)
- [Abort The Process](#)
- [Abort, Change Input, and Create New Instance](#)
- [Retry Several Faulted Activities](#)

Retry the Activity An example of a scenario in which to use this recovery action is when the fault occurred because the service provider was not reachable due to a network error. The network error is now resolved.

1. Click **Recover** to retry the activity.

Actions available: Retry On Retry Success Chain to this action: None

Recover New Instance

☒ Fault Details

None
ora-java

All Java actions available in the fault policy used by the automated fault recovery are shown in the dropdown list.

Note the following details about the retry action:

- If the retry faults again, the framework goes through the same steps as earlier and returns to this point.
- If the same fault is identified, the activity is marked for human intervention.
- If the retry attempt is successful, the framework invokes `handleRetrySuccess` on the custom Java class provided under the action selected here.

Change the Input Variable Contents and Retry You can modify the input variable contents used by the faulted activity. All variables available to this faulted activity are listed in a dropdown list.

1. Select the variable and click **Get**. This retrieves the value of the variable in the context of the faulted activity.

The screenshot shows the 'BPEL Processes' tab with a table of process instances. The instance '10018 - Instance #10018 of LoanFlow' is selected. Below the table, the 'Available Variables' dropdown is open, showing a list of variables including 'crInput', 'loanApplication', 'input', 'crError', 'selectedLoanOffer', 'loanOffer2', 'loanOffer1', and 'crOutput'. The 'crInput' variable is selected, and its value is displayed in the text area as `://services.otn.com">012-22-1234</ssn>`. The 'Type' is `{http://services.otn.com}ssn` and the 'Help' link is [Schema Type Formats \(wsdl\)](#). The 'Actions available' dropdown is set to 'Retry', and the 'On Retry Success Chain to this action' is set to 'None'. The 'Recover' and 'New Instance' buttons are visible, along with a checkbox for 'Fault Details'.

2. Alter the variable contents in the text area and click **Set** to modify the variable in the context of the faulted activity.

The screenshot shows the 'BPEL Processes' tab with the 'crInput' variable selected. The value in the text area has been changed to `<ssn xmlns="http://services.otn.com">512-22-1234</ssn>`. The 'Type' is `{http://services.otn.com}ssn` and the 'Help' link is [Schema Type Formats \(wsdl\)](#). The 'Get' and 'Set' buttons are visible, along with the 'Skeleton Value' button.

Note that the **Skeleton Value** button enables you to retrieve the variable for editing without displaying its actual value.

The screenshot shows the 'BPEL Processes' tab with the 'crInput' variable selected. The 'Skeleton Value' button is highlighted, and the text area displays the skeleton value `<ns:ssn xmlns:ns="http://services.otn.com"></ns:ssn>`. The 'Type' is `{http://services.otn.com}ssn` and the 'Help' link is [Schema Type Formats \(wsdl\)](#).

3. Select **Retry** in the list of available actions and click **Recover** to retry the activity.

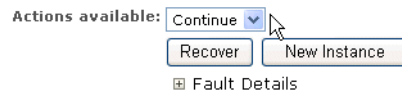


Set the Output and Continue An example of a scenario in which to use this recovery action is when a service provider billing system is not reachable. You decide to either:

- Skip the call to billing
- Manually perform the billing and continue the current instance

Note: This action is only useful for a synchronous process. For an asynchronous process, marking the invoke activity as a success is not useful because Oracle BPEL Server waits at the matching receive activity.

1. Retrieve and modify the variable data as described in ["Change the Input Variable Contents and Retry"](#) on page 44.
2. Select **Continue** as the action and click **Recover** to mark the activity as a success. When an activity is marked as a success, the faulted activity is ignored and processing continues to the next activities.



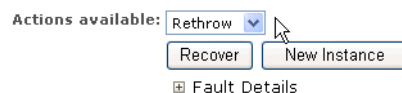
Replay the Scope You can replay the scope in which the fault occurred.

1. Retrieve and modify the variable data as described in ["Change the Input Variable Contents and Retry"](#) on page 44.
2. Select **Replay** as the action and click **Recover** to replay the scope.



Rethrow the Exception You can rethrow the exception and enable the BPEL fault handlers (catch branches) to process the fault.

1. Retrieve and modify the variable data as described in ["Change the Input Variable Contents and Retry"](#) on page 44.
2. Select **Rethrow** as the action and click **Recover** to rethrow the fault.



Abort The Process You can abort the instance of the faulted activity.

1. Select **Abort** as the action and click **Recover** to abort the instance.



Abort, Change Input, and Create New Instance An example of a scenario in which to use this recovery action is if the instance faulted because of bad input to the process, the current instance must be terminated, and a new instance with a modified payload must be instantiated.

1. Click **New Instance** to access the Initiate page. The selection for operation in the dropdown list (**initiate**) and the input payload are already filled in.
2. Modify the value for the payload input variable used in the faulted instance and click **Initiate** to initiate a new instance.

BPEL Process: LoanFlow Version: 1.0 Lifecycle: Active
Statistics: [5 Open Instances](#) | [0 Closed Instances](#)

Manage	Initiate	Descriptor	WSDL	Sensors	Source	Test Suites
--------	----------	------------	------	---------	--------	-------------

Testing this BPEL Process Through SOAP | Through

Initiating a test instance
To create a new 'test' instance of this BPEL Process, fill the following text area with the XML representation of a message and click on the 'Post XML Message' button.

Operation: initiate HTML Form XML Source

```
<ns1:loanApplication xmlns:ns1="http://www.autoloan.com/ns/autoloan">
  <ns1:SSN>012-22-1234</ns1:SSN>
  <ns1:email>john@test.com</ns1:email>
  <ns1:customerName>John</ns1:customerName>
  <ns1:loanAmount>15000.0</ns1:loanAmount>
  <ns1:carModel>Lexus</ns1:carModel>
  <ns1:carYear>2007</ns1:carYear>
  <ns1:creditRating>740</ns1:creditRating>
</ns1:loanApplication>
```

Retry Several Faulted Activities You can attempt bulk recoveries on multiple faulted activities.

1. Return to the **Activities** tab of the faulted instance.
2. Select all activities that must be recovered. Check boxes appear only for activities that can be recovered.
3. Select a recovery action from the dropdown list and click **Recover**.

Dashboard
BPEL Processes
Instances
Activities

Locate Activities

Instance Id#

Activity Label

Fault Name

Index

Activity State
Pending

BPEL Process
All Processes

Last Modified
All Times

Go

Activities 1 - 3

	Activity Label	Instance	Process	Due	Fault Name
<input checked="" type="checkbox"/>	invokeCR	10018	LoanFlow (v. 1.0)	9/17/07 2:29:53 PM	NegativeCre
<input checked="" type="checkbox"/>	invokeCR	10013	LoanFlow (v. 1.0)	9/17/07 2:24:44 PM	NegativeCre
<input checked="" type="checkbox"/>	invokeCR	10008	LoanFlow (v. 1.0)	9/17/07 12:24:55 PM	NegativeCre

Check All - Clear All

Actions available:

Retry
Abort
Rethrow
Replay
Continue

Recover

Oracle BPEL Console v10.1.3

Local intranet

Fault Logging

All fault information is logged using the current logging scheme. There is no option to add to logging and no additional data that you can choose to log in the event of a fault. While you cannot control the level of logging scheme, you can provide a custom Java class using `JavaAction` in the fault policy. This enables custom logging to be performed based on a fault.

To log to another file or database, attach a policy with `JavaAction` that invokes a user Java class, and perform all user-specific logging in the Java code.

To use the `JavaAction` policy to perform logging only, chain `JavaAction` to a rethrow action using the `defaultAction` attribute. This sends the fault to the process-defined fault catch branch. In this case, the custom Java class performs logging only.

Java Action Fault Policy

Note the following details when using the Java action fault policy.

- The Java class provided follows a specific interface. This interface returns a string. Multiple values can be provided for output and fault policy to take after execution.
- Additional fault policy can be executed by providing a mapping from the output value (return value) of implemented methods to a fault policy.
- If no `ReturnValue` is specified, the default fault policy is executed:

```

<Action id="ora-java">
  <JavaAction ClassName="mypackage.myclass"
    defaultAction="ora-human-intervention" propertySet="prop-for-billing">
    <!--defaultAction is a required attribute, but propertySet is optional-->
    <!-- attribute-->
    <ReturnValue value="RETRY" ref="ora-retry"/>
    <!--value is not nilable attribute & cannot be empty-->
    <ReturnValue value="RETHROW" ref="ora-rethrow-fault"/>
  </JavaAction>

```

</Action>

Table 2 provides an example of ReturnValue use.

Table 2 System Interpretation of Java Action Fault Policy

Code	Description
<code><ReturnValue value="RETRY" ref="ora-retry" /></code>	Execute the ora-retry action if the method returns a string of RETRY
<code><ReturnValue value="" ref="ora-rethrow" /></code>	Fails in validation
<code><JavaAction ClassName="mypackage.myclass" defaultAction="ora-human-intervention"></code>	Execute ora-human-intervention after Java code execution. This attribute is used if the return from the method does not match any provided ReturnValue.
<code><ReturnValue value="RETRY" ref="ora-retry" /> <ReturnValue value="" ref="" /></code>	Fails in validation
<code><JavaAction ClassName="mypackage.myclass" defaultAction=" ora-human-intervention"> <ReturnValue></ReturnValue></code>	Fails in validation

To invoke a Java class, you can provide a class that implements the IFaultRecoveryJavaClass interface. This interface has two methods:

```
public interface IFaultRecoveryJavaClass
{
    public void handleRetrySuccess(IFaultRecoveryContext ctx );
    public String handleBPELFault( IFaultRecoveryContext ctx );
}
```

Note the following details:

- handleRetrySuccess is invoked upon a successful retry attempt. The retry policy chains to a Java action on retrySuccessAction.
- handleBPELFault is invoked to execute a policy of type javaAction.

The following data is available with FaultRecoveryContext:

```
public interface IFaultRecoveryContext
{
    public Map getProperties();
    public String getActionId();
    public String getPolicyId();
    public String getActivityType();

    public String getActivityId();
    public String getActivityName();
    public String getWsdLocation();
    public String getPartnerLinkName();
    public QName getPortType();
    public String getCorrelationId();
    public BPELFault getFault();
    public BPELProcessId getProcessId();
    public String getStatus();
}
```



```

public void setStatus(String status);
public String setTitle(String title);
public String getTitle();
public int getPriority();
public void setPriority(int priority);
public long getInstanceId();
public Locator getLocator();
public void addAuditTrailEntry(String message, Object detail);
public void addAuditTrailEntry(String message);
public void addAuditTrailEntry(Throwable t);
public Object getVariableData(String name) throws BPEL Fault;
public Object getVariableData(String name, String partOrQuery)
    throws BPEL Fault;
public Object getVariableData(String name, String part, String query)
    throws BPEL Fault;
public void setVariableData(String name, Object value) throws BPEL Fault;
public void setVariableData(String name, String partOrQuery, Object value)
    throws BPEL Fault;
public void setVariableData(String name, String part, String query,
    Object value) throws BPEL Fault;
}

```

[Example 4](#) provides an example of javaAction implementation.

Example 4 Implementation of a javaAction

```

public class TestJavaAction implements IFaultRecoveryJavaClass
{
    public void handleRetrySuccess(IFaultRecoveryContext ctx)
    {
        System.out.println("This is for retry success");
        handleBPEL Fault(ctx);
    }
    public void dumpProperties(Map properties)
    {
        // check if there were properties
        if(properties.size() == 0 )
            return;
        System.out.println("----Begin propeties -----");
        Set entries = properties.entrySet();
        Iterator iterator = entries.iterator();
        while (iterator.hasNext())
        {
            Map.Entry entry = (Map.Entry) iterator.next();
            System.out.println("\nKey="+entry.getKey());

            // all values are list of strings
            List propValueList = (List)entry.getValue();
            Iterator it = propValueList.iterator();
            while (it.hasNext())
            {
                System.out.print("Value="+it.next()+"\t");
            }
        }
        System.out.println("\n----End propeties -----");
    }

    public String handleBPEL Fault(IFaultRecoveryContext ctx)

```

```

    {
        System.out.println("-----Inside handleFault-----\n" + ctx.toString());

        dumpProperties(ctx.getProperties());

        ctx.addAuditTrailEntry("hi there");
        System.out.println("-----End Inside handleFault-----");
        return "MANUAL";
    }
}

```

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

<http://www.oracle.com/accessibility/>

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Oracle SOA Suite New Features
E10381-02

Copyright © 2005, 2007, Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted

Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

