Chapter 5
"Oracle BPEL Process Manager and BPEL Designer: Overview"
In this package, you will find:

A Biography of the authors of the book

A preview chapter from the book, Chapter 5 "Oracle BPEL Process Manager and BPEL Designer: Overview"

A synopsis of the book’s content

Information on where to buy this book

About the Authors

Matjaz B. Juric holds a Ph.D. in computer and information science. He is Associate Professor at the University of Maribor. In addition to this book, he has coauthored Professional J2EE EAI, Professional EJB, J2EE Design Patterns Applied, and .NET Serialization Handbook, published by Wrox Press. He has published chapters in More Java Gems (Cambridge University Press) and in Technology Supporting Business Solutions (Nova Science Publishers). He has also published in journals and magazines, such as Java Developer’s Journal, Java Report, Java World, Web Services Journal, eai Journal, theerverside.com, OTN, and ACM journals, and presented at conferences such as OOPSLA, Java Development, XML Europe, OOW, SCI, and others. He is a reviewer, program committee member, and conference organizer. Matjaz has been involved in several large-scale object technology projects. In cooperation with IBM Java Technology Centre, he worked on performance analysis and optimization of RMI-IIOP, an integral part of the Java platform.

Matjaz is an author of courses and consultant for the BPEL and SOA consulting company BPELmentor.com. For more information, please visit http://www.bpelmentor.com/

My efforts in this book are dedicated to my family. Special thanks to Jerneja, my friends at the University of Maribor, and to Louay and Damian at Packt Publishing.
Benny K. Mathew is a Sr. Software Engineer at IBM Global Services (India), Bangalore. He holds a Masters degree in Computer Applications. His fascination for computers started at the age of 14, when he first experienced the delight of programming on a Sinclair ZX Spectrum+.

He has co-authored books and articles on technologies like .NET, BizTalk, BPEL, etc.

During his free time, Benny likes to read blogs and help people on the newsgroups related to .NET and BizTalk. He has been awarded Microsoft Most Valuable Professional (MVP) for two consecutive years.

Before joining IBM, he was with companies like Hewlett Packard, Thomson Financials, and Delphi Software.

Poornachandra Sarang, Ph.D., is CEO of ABCOM Information Systems. Dr. Sarang is one of the leading software architects in the industry, has more than 20 years of IT experience and provides consulting services to worldwide clients in architecting and designing IT solutions based on Java, CORBA, Oracle, and .NET platforms. He has been a Visiting Professor of Computer Engineering at the University of Notre Dame, USA and is currently a visiting professor for Post-Graduate Computer Science studies at the University of Mumbai. He conducts lectures/seminars on emerging technologies across the world and has made several presentations at international conferences. He has authored/coauthored several books and journal articles on Java, C++, J2EE, e-Commerce, Open-Source Technologies, and .NET.
In this chapter, we will get familiar with the Oracle BPEL Process Manager 10g Release 2, a server for deployment, execution, and management of business processes defined in BPEL. Oracle BPEL Process Manager offers several features that make it one of the most powerful BPEL servers at the time of writing this book. We will look at the BPEL Server, command-prompt utilities, and the BPEL Console. We will also look at the BPEL Designer, an integrated graphical development environment. BPEL Designer is available integrated into JDeveloper or as an Eclipse plug-in. BPEL Designer enables BPEL process development using a graphical editor instead of writing BPEL code by hand. It supports debugging, provides several wizards, enables automatic deployment of BPEL processes, and eases the development and maintenance of BPEL processes considerably.

In this chapter, we will discuss the following:

- Architecture of the BPEL Process Manager
- Major features
- Process deployment
- Management and debugging of processes with the BPEL Console
- Graphical development with JDeveloper BPEL Designer
- Graphical development with Eclipse BPEL Designer

Overview and Architecture

The Oracle BPEL Process Manager is a run-time environment for BPEL processes. BPEL Process Manager 10g Release 2 (10.1.2) fully supports BPEL version 1.1 and provides additional tools for deployment, monitoring, and management of BPEL processes. At the time of writing this book, Oracle BPEL Process Manager is one of the most complete BPEL servers available.

BPEL Process Manager has been developed in Java and runs on a Java Enterprise Edition-compliant application server, for example, the Oracle Application Server 10g or OC4J (Oracle Containers for Java). Oracle also provides versions for the open-source JBoss and for the BEA WebLogic Server. With manual installation, Oracle BPEL Process Manager can also be used with IBM and Sun application servers.
Let us look at the architecture of the BPEL Process Manager, shown in the following figure:

The Oracle BPEL Process Manager has four major parts:

- BPEL Server
- BPEL Console
- BPEL Designer (JDeveloper or Eclipse)
- Database

BPEL Server

BPEL Server runs in a Java Enterprise Edition-compliant application server. It has the following main parts:

- Core BPEL engine
- WSDL bindings
- Integration services

Core BPEL Engine

The core BPEL engine is the run-time environment where BPEL processes are deployed and executed. In addition to full BPEL 1.1 support, this engine provides support for key web services orchestration stack technologies, particularly WS-Addressing, WS Reliable Messaging, WS-Security, and the BPEL compensating transaction model.

The BPEL engine also provides support for version control. This enables development of several versions of a business process, which can be deployed side by side. This feature is important in real-world scenarios because business processes evolve over time. Having effective versioning support simplifies the management.

Another very important feature is dehydration. In previous chapters, we have explained that business processes can be long running because the involved partners might not be able to react instantly to the requests. This happens particularly in asynchronous scenarios where a business process invokes a partner web service (using the `<invoke>` activity) and then waits for the response (using the `<receive>` or `<pick>` activities). While waiting for the response the Oracle engine can store the process (and its state) in the database, thus freeing up server resources. This is called dehydration. When the engine receives the response it first restores the process with its state from the database (hydration) and then continues with the execution of the process. In real-world scenarios where many business processes might be running side by side, the dehydration capability is important as it reduces the demands on hardware performance.

Oracle BPEL engine also provides support for clustering. Clustering increases server reliability because fail-over can be configured on the engine. Clustering also improves scalability with load balancing. These features are very important in real-world usage of the product.

WSDL Bindings

The WSDL binding framework is responsible for communication with the BPEL processes deployed on the server. This includes clients that would like to access a BPEL process and BPEL processes that would like to access other web services (partner links). Although BPEL specification talks only about web services, Oracle BPEL Process Manager enables connectivity using protocols other than SOAP. This is particularly useful in real-world usage scenarios where business processes often have to connect to existing applications or systems. Using the WSDL binding framework, the reach of BPEL is extended to these systems without the need to convert them to web services.

Of particular interest here is connectivity to Java EE artifacts, such as EJBs (Enterprise Java Beans), RMI (Remote Method Invocation), JMS (Java Message Service), JCA (Java Connector...
Architecture), and also to email, HTTP GET and POST, and sockets. The integration is achieved through the WSIF (Web Services Invocation Framework). All this enables relatively easy and efficient integration of back-end systems, particularly existing and legacy systems, which cannot be simply exposed as web services. Through WSDL bindings BPEL Process Manager provides some features usually offered by ESB (for more ESB features please refer to Oracle Fusion).

Integration Services

Business processes described in BPEL communicate with web services and exchange XML documents. The integration services provide support for transformations (on these XML documents) that go beyond the support of XPath. Oracle BPEL Server provides support for XSLT (Extensible Stylesheet Language for Transformations) transformations and XSQL. It provides engines and the XSLT Mapper, which simplifies the development of stylesheets.

XSLT provides support for complex transformations of XML vocabularies and can be also used to transform XML to other markup formats such as HTML, WML, or VoiceXML for presentation purposes. For more information on XSLT please refer to http://www.w3.org/TR/xslt, and XQuery, which can also be used, are XML query languages with functionality that goes beyond simple XPath queries. For more information on XSQL please refer to Oracle documentation. For more information on XQuery please refer to http://www.w3.org/XML/Query.

Through the Workflow service BPEL Server provides support for human interaction with BPEL process. Human interaction is often required in processes—typical examples include human reviews, confirmations, or decisions before carrying out further steps—but not covered by the BPEL specification. The Workflow service provides a WSDL interface through which human interaction is modeled as partner links. The Workflow service has augmented the User Task service, known from previous versions of BPEL Process Manager.

The Notification service provides a relatively easy-to-use interface through which BPEL processes can send notifications to clients using different (most asynchronous) channels, such as email, short message service (SMS), voice message, fax, etc. The Notification service has augmented the Mail and JMS services, known from previous versions of BPEL Process Manager.

The Identity service provides access to the application server security infrastructure through the web services layer and enables user authentication, authorization, and access to various used properties. The Notification, Workflow, and Identity services are often used together.

BPEL Server also provides Sensors, through which access to activities, variables, and faults is provided. These can be monitored during the execution of BPEL processes. Sensors can be useful for reporting, integration with other processes, and for debugging.

To integrate BPEL processes with access to files, FTP servers, database tables and queues, JMS, or Oracle Applications, BPEL Server provides support for technology adapters. Adapters provide access to these resources through WSDL interfaces. They are based on the JCA 1.5 resource adapters and on the WSIF.

BPEL Console

Through the BPEL Console we can deploy, manage, administer, and debug BPEL processes. The most important features of the BPEL Console include:
Oracle BPEL Console uses a web-based interface, which is basically a set of JSP (Java Server Pages) and servlets that call the BPEL Server API (in Java). Through the API we can extend the functionality of the console or even develop our own console if we need specific handling of BPEL processes.

**BPEL Designer**

BPEL Designer enables the development of BPEL processes in a graphical environment without having to write BPEL code by hand. Instead of writing code, we drag and drop activities to the process. We can add partner links and locate services. We can also use function and copy wizards, XPath expression builder, and XSLT mapper. BPEL Designer can deploy the developed processes directly to the BPEL Server. This eases the development and maintenance of BPEL processes considerably. BPEL Designer internally uses BPEL as its native format. Because it uses standard BPEL, processes developed by the BPEL Designer can be used with other BPEL servers (and vice-versa) as long as we do not use functionality specific to the Oracle product.

BPEL Designer is available in two versions. It is either integrated with JDeveloper (called JDeveloper BPEL Designer) or a plug-in for the Eclipse platform (Eclipse BPEL Designer). JDeveloper and Eclipse BPEL Designer offer comparable features. JDeveloper BPEL Designer, however, also provides support for some advanced Oracle features, such as sensors and adapters. We will discuss BPEL Designer later in this chapter.

**Database**

The database is used by BPEL Server to store the BPEL schemas and to provide support for additional features, particularly the dehydration, which stores the process state in the database. BPEL Process Manager provides support for different databases. Usually Oracle Database, or other production-quality DBMS systems such as IBM DB2 or Microsoft SQL Server will be used. The trial version, which can be downloaded from Oracle's website, comes bundled with Oracle Database Lite, which is not appropriate for production use as it offers poor performance and does not support all features.

**Process Deployment Example**

Let us now show how we deploy a BPEL process on the Oracle BPEL Server. We will assume that Oracle BPEL Process Manager has been successfully installed according to the installation instructions and that it uses the default port 9700. If another port has been selected during installation, the examples have to be modified accordingly.
We will use the Business Travel BPEL process example that we have developed in Chapters 3 and 4. The travel example is a simplified business process that selects the best airline ticket offer. To refresh our memory, let us have a look at the process activity diagram:

In the previous chapters we developed the BPEL code for the example; this consists of a `Travel.bpel` file with the source code and the `Travel.wsdl` file where the WSDL definitions are stored. We will not show the source of these files here as they have been already shown in previous chapters. They can also be downloaded from [http://www.packtpub.com](http://www.packtpub.com).

**Process Descriptor**

Each BPEL process we deploy to the Oracle BPEL Process Manager requires a process descriptor. This process descriptor is not covered by the BPEL standard and is specific to the BPEL server.

The deployment process descriptor is the only part of the implementation of a process on a given platform that must be rewritten to run the process on a different BPEL server.

The Oracle process descriptor is an XML file specifying the following details about the BPEL process:

- BPEL source file name
- BPEL process name (ID)
- WSDL locations of all partner link web services
- Optional configuration properties

The default file name for the process descriptor is `bpel.xml`, but we can use any other name. Let us now write the process descriptor for our process. First, we have to specify the XML header and the `<BPELSuitcase>` root element. In the `<BPELProcess>` element we specify two attributes, `src`, which denotes the BPEL source file name (`Travel.bpel`), and `id` (`id`), which denotes BPEL process name as shown in the BPEL Console (we will use the `TravelProcessCh5` ID):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<BPELSuitcase>
  <BPELProcess src="Travel.bpel" id="TravelProcessCh5">
    ...
  </BPELProcess>
</BPELSuitcase>
```

Next, we specify the partner link binding properties for the location of the WSDL for each partner link that we use in the process. In our travel example process we use the following partner links:

- `client`: Used for client interaction with the process
- `employeeTravelStatus`: The link to the Employee web service
- `AmericanAirlines`: The link to the American Airlines web service
- `DeltaAirlines`: The link to the Delta Airlines web service

The WSDL for the `client` partner link is stored locally in the `Travel.wsdl` file. For the location of the other three partner web services' WSDLs, we have to specify the corresponding URLs. Here we have provided simplified implementations of all three web services, which can also be downloaded and deployed on the Oracle BPEL Server. The rest of the process descriptor with the location of the WSDLs is shown below:

```xml
...<partnerLinkBindings>
  <partnerLinkBinding name="client">
    <property name="wsdlLocation">
      Travel.wsdl
    </property>
  </partnerLinkBinding>

  <partnerLinkBinding name="employeeTravelStatus">
    <property name="wsdlLocation">
    </property>
  </partnerLinkBinding>
</partnerLinkBindings>
```
The only required partner link binding property is `wsdlLocation`. Optionally we can add other partner link binding properties. The most important ones are listed in the table below (for the complete list please refer to Oracle documentation):

<table>
<thead>
<tr>
<th>Partner link binding property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wsdlRuntimeLocation</code></td>
<td>Specifies the run-time WSDL location of the partner link services (if different from <code>wsdlLocation</code>).</td>
</tr>
<tr>
<td><code>correlation</code></td>
<td>Specifies the type of correlation used (correlation is discussed in Chapter 4):</td>
</tr>
<tr>
<td></td>
<td>• <code>wsAddressing</code> for WS-Addressing</td>
</tr>
<tr>
<td></td>
<td>• <code>correlationSet</code> for BPEL correlation sets</td>
</tr>
<tr>
<td><code>timeout</code></td>
<td>Specifies the timeout for partner link invocation calls.</td>
</tr>
<tr>
<td><code>httpUsername</code> and <code>httpPassword</code></td>
<td>Specifies the username and password for HTTP authentication.</td>
</tr>
</tbody>
</table>

### Configuration Properties

Optionally we can add configuration properties to the deployment descriptor, such as introduction text outputted by the BPEL Console when starting the process and default input data. The introduction text should be included within the `<property>` element with the attribute `name` set to `testIntroduction`:

```xml
...<configurations>
    <property name="testIntroduction">
        The Business Travel Process example.
    </property>
...```

To add the default input data (also optional) we have to define the `<property>` element with the attribute `name` set to `defaultInput` and provide the input XML message as CDATA:

```xml
...<property name="defaultInput">
    <![CDATA[
        <TravelRequest xmlns="http://packtpub.com/bpel/travel/">
            <employee xmlns="http://packtpub.com/service/employee/">
                <FirstName>Matjaz B.</FirstName>
                <LastName>Juric</LastName>
                <Departement>University</Departement>
            </employee>
            <flightData xmlns="http://packtpub.com/service/airline/">
                <OriginFrom>Ptuj</OriginFrom>
            </flightData>
    ]]>
</property>
```
Other important optional configuration properties are listed in the table below (for the complete list please refer to Oracle documentation):

<table>
<thead>
<tr>
<th>Configuration property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loadSchema</td>
<td>If this is set to False, BPEL processes become typeless because XML schemas are not loaded. Default is True.</td>
</tr>
<tr>
<td>relaxTypeChecking</td>
<td>If this is set to True, BPEL compiler does not check types for compatibility in assigns. Default is False.</td>
</tr>
<tr>
<td>xpathValidation</td>
<td>If this is set to False, BPEL compiler does not validate XPath expressions. Default is True.</td>
</tr>
<tr>
<td>relaxXPathQName</td>
<td>If this is set to True, BPEL compiler does not check name spaces for steps in XPath expressions. Default is False.</td>
</tr>
<tr>
<td>transaction</td>
<td>If this is set to participate, faults not handled by fault handlers produce a rollback for transactions in which they are executed.</td>
</tr>
</tbody>
</table>

Setting the Environment

We are now ready to start the BPEL Process Manager. We can do this from the Start menu (if using Windows) or by executing the startOraBPEL script, which can be found in the C:\OraBPELPM_1\integration\orabpel\bin directory (assuming Oracle BPEL Process Manager has been installed in the default directory). It is recommended that we include this directory in the path for easy access.

Next, we will need a command prompt where we have to set the environment variables. We can select the Developer Prompt from the Start menu or execute the obsetenv script in the C:\OraBPELPM_1\integration\orabpel\bin directory. The script sets several environment variables—we list the most important here:

- OB_HOME specifies the path to the Oracle BPEL installation directory.
- OB_PLATFORM specifies the application server (oc4j_10g if using Oracle OC4J).
- MY_CLASSPATH and MY_CLASSES_DIR specify the class path for Oracle BPEL Server.
- JAVA_HOME points to the Java SDK home directory.
- J2EE_APPLICATIONS specifies the application server directory where J2EE applications can be deployed.

Setting the environment variables is essential for successful deployment of BPEL processes.
BPEL Compiler and Revision Numbers

After we have written the process descriptor and set the environment, we are ready to deploy the BPEL process. For this, Oracle BPEL Process Manager provides the BPEL Compiler, which can be started with the `bpelc` command from the command line. `bpelc` compiles the BPEL process source files and creates the BPEL process archive JAR file. It can also automatically deploy the process to the Oracle BPEL Server. This is discussed in the next section.

The BPEL compiler has the following syntax:

```
> bpelc [options] process_descriptor_name.xml
```

The default for `process_descriptor_name.xml` is `bpel.xml`. The most important options are:

- `-rev <revision_tag>`: This specifies the revision (version) number for the deployed BPEL process.
- `-force`: This directs the compiler not to check the timestamps of the `.bpel`, `.wsdl`, and `.xml` files.

Revision numbers are used to deploy different versions of the same BPEL process on the Oracle BPEL Process Manger and run them simultaneously.

Revision numbers are most useful in production environments, where it is more appropriate to deploy new version of a process under a new revision number than to override an existing revision number. In the latter case, all existing process instances get stalled and cannot be accessed anymore. If we use a new revision number, we can mark older process revisions as retired but retain access to already created instances. We will explain how to mark processes as retired later in this chapter.

We can use the following command to generate the BPEL process JAR archive with the revision number 1.0 for our travel example:

```
> bpelc -rev 1.0
```

This command generates the `bpel_TravelProcessCh5_1.0.jar` archive file, as shown:

```
Z:\Chapter5\i_TravelProcess>bpelc -rev 1.0
Oracle BPEL Processor Version 10.1.2.0.0
http://otn.oracle.com/bpel
Copyright (c) 2002-2004 Oracle
(!type bpelc -help for help)

validating "Z:\Chapter5\i_TravelProcess\Travel.bpmn" ... BPEL suitecase generated in: Z:\Chapter5\i_TravelProcess\bpel_TravelProcessCh5_1.0.jar
bpelc completed successfully.
Z:\Chapter5\i_TravelProcess>
```

The generated archive includes the BPEL source and the related WSDL and XML files. It also includes the process model file (in our example called TravelModel.xml) which is a normalized BPEL representation with an added id for each activity.

**Deployment and Domains**

To deploy our travel process we have several options:

- Copy the BPEL archive to the server domain manually
- Use `bpelc` to deploy the process
- Use the `obant` utility to do the deployment
- Use the BPEL Console to deploy the BPEL process archive

Before we deploy, let's discuss the Oracle BPEL Server architecture. Each Oracle BPEL Server installation can be logically partitioned into several domains. The default domain is created automatically by the installation (called `default`). Additional domains can be created using the BPEL Console; this is discussed later in the chapter. The domains are located in the `C:\OraBPELM_1\integration\orabpel\domains` directory.

To manually deploy a BPEL process we simply copy the JAR archive to the corresponding directory. In our case this is `C:\OraBPELM_1\integration\orabpel\domains\default\deploy`. The BPEL Server will automatically pick up the process.

Using `bpelc` to deploy the process requires us to use the `deploy <domain_id>` option, which directs the compiler to automatically deploy the archive to the specified domain. The domain to which the deployment is done must be accessible via the file system. To deploy our travel example using `bpelc` we need to use the following command:

```
> bpelc -rev 1.0 -deploy default
```

The following screenshot shows the output:

![Screenshot of BPEL deployment output]

**Ant Utility**

The Oracle BPEL Process Manager provides the Ant utility called `obant`. This can be used to configure complex compilation and deployment scenarios. `obant` is just a wrapper around standard
Ant, which sets the environment and then invokes the standard Ant Java task. To use it we have to prepare the corresponding project file, usually called build.xml. The project file for our travel example process is shown below:

```xml
<?xml version="1.0"?>
<project name="TravelProcessCh5" default="main" basedir="."/>
  <property name="deploy" value="default"/>
  <property name="rev" value="1.0"/>
  <target name="main">
    <bpelc home="${home}" rev="${rev}" deploy="${deploy}"/>
  </target>
</project>
```

For more information on Ant, visit http://ant.apache.org/.

To compile and deploy our BPEL process we simply start `obant` from the command line. The output is shown in the following screenshot:

![Developer Prompt](image)

### Process Management with the BPEL Console

Now that we have successfully deployed a BPEL process on the Oracle BPEL Server, let's execute it. In Chapter 3 we mentioned that each BPEL process is a web service. Therefore, to start the BPEL process we need to invoke it just like any other web service. This requires writing a web services client based on the WSDL. Because web services are not bound to a particular platform or programming language, we can do this using most languages (Java, C#, VB.NET, Delphi, etc.), applications (SAP, Navision, even Microsoft Office), tools (XML Spy), or other BPEL processes.

In addition to these options, Oracle BPEL Process Manager provides a BPEL Console through which we can execute, monitor, manage, and debug BPEL processes on a BPEL Server domain. The BPEL Console is accessible at http://localhost:9700/BPELConsole/. Of course we can replace localhost with the valid computer name URL. Once we enter the domain password, we can start our travel process and create a new process instance by clicking the process name (TravelProcessCh5) on the BPEL Console dashboard, as shown in the following screenshot:
Note that in addition to the Travel Process, the Employee, American Airline, and Delta Airline web services have to be deployed as well. After clicking `TravelProcessCh5` we have to enter the input XML message (the default from the process descriptor is shown) and click the `Post XML Message` button:
We can also switch to the HTML form and enter the necessary fields:

Next, we see a screen notifying us that the process instance is being processed asynchronously, because this is an asynchronous process. If this was a synchronous process, we would see the final result (returned through the `<reply>` activity) immediately. The screen is shown below:
Visual Flow

In the next step, we can select the visual flow of the execution, instance auditing, or instance debugging. The visual flow of the instance graphically shows the execution of a BPEL process instance. We can monitor the execution of the process and its state (running, completed, canceled, or stale):

![Visual Flow screenshot](image)

The important thing is that we can click on each activity symbol (such as `<receive>`, `<assign>`, etc.) and we will see the corresponding XML input and output. This enables us to verify the processing of each activity. Clicking on the first `<receive>` activity (client `TravelApproval`) would open this screen, showing the received message, `Travel Request`:
Instance Auditing

The audit view of the process instance, which we can activate by selecting the Audit tag, shows a complete BPEL process with the received and sent messages. Clicking on the More… link reveals the complete XML of messages exchanged. This view is useful for auditing the messages exchanged by the process and the execution of other activities, particularly those manipulating data, such as <assign>. The following screenshot shows the audit trail of our travel example process:

Debugging

The debug view of a process instance (accessible via the Debug tag) shows the BPEL source code. Clicking on the underlined variable names provides access to variable content. We can debug already completed instances or instances that are still running; this is called in-flight debugging. The debug view shows the current state of the instance. If we use in-flight debugging, the point where execution is paused is shown highlighted. The following figure shows the debug view of our process instance, once completed:
Clicking on the TravelResponse variable, for example, gives us the following output:

```xml
<variable name="TravelResponse" messageType="http://iwmw.com/bpel/travelResponse">
  
  <confirmationData>
    
    <flightNo>
      <mshs="http://iwmw.com/service/airline"/>
    </flightNo>
  
    <returnDate>
      <mshs="http://iwmw.com/service/airline"/>
    </returnDate>
  
    <departDate>
      <mshs="http://iwmw.com/service/airline"/>
    </departDate>
  
    <approved>
      <mshs="http://iwmw.com/service/airline"/>
    </approved>
  
  </confirmationData>

  <TravelResponse>
    
    <confirmationData>
      
      <flightNo>
        <mshs="http://iwmw.com/service/airline"/>
      </flightNo>
    
    <returnDate>
      <mshs="http://iwmw.com/service/airline"/>
    </returnDate>
  
    <departDate>
      <mshs="http://iwmw.com/service/airline"/>
    </departDate>
  
    <approved>
      <mshs="http://iwmw.com/service/airline"/>
    </approved>
  
  </confirmationData>

</variable>
```
Overview of Other BPEL Console Functions

Using the BPEL Processes tab, we can manage the process lifecycle and process state. The state of a process can be on or off. When the state of a process is off, new instances cannot be created and access to existing instances is blocked. The lifecycle of a process can be active or retired. When a process is retired, new instances cannot be created. Existing instances can, however, complete normally. The retired state is usually used for process revisions that have been superseded by newer revisions:

From this view we can navigate to the following:

- **Initiate**: Initiate new instances of the process (described in previous section)
- **Descriptor**: Look at the properties defined in the BPEL process descriptor (bpe1.xmi)
- **WSDL**: Get the WSDL location and the BPEL process endpoint location
- **Sensors**: Get an overview of the sensors defined for the process
- **Source**: Look at the process source code

Under the **Instances** tab we can overview process instances. We can archive and purge instances, remove completed process instances, and supervise those that have not completed yet. A BPEL process instance can have the following states:

- Running
- Completed
- Canceled
- Stale

For our example, we can see that four instances have completed successfully:

Under the **Activities** tab we can locate activities by name and find relations to instances and processes. An activity can also have four states: open, completed, canceled, or stale:

Deploying Processes

We have already mentioned that new processes can be deployed using the BPEL Console. To do this we first have to generate the process JAR archive (using `bpelc` or `obant`). We then click the Deploy New Processes link on the Dashboard or BPEL Processes tabs. We specify the full path to the process JAR archive and press the Deploy button, as shown below:
Management

The BPEL Console also provides tools for BPEL domain management. These can be accessed by pressing the Manage BPEL Domain link in the upper right corner of the screen, which brings us to the following screen:

The management console has the following important options:

- Configuration descriptor (Configuration)
- Setting the password (Password)
- List of XPath extension functions for the domain (XPath Library)
- Setting the logging configuration (Logging)
- Thread allocation statistics (Threads)
- Runtime performance statistics (Statistics)
- Adapter statistics (Adapter Stats)

The configuration descriptor enables us to set various important parameters that affect the BPEL Server operation. These parameters include:

- Process instance stale check interval (specified in seconds)
- Allocation of invocation threads and load factor

- Instance cache size (minimum and maximum)
- Persistence and database parameters
- Management of large XML documents
- Recovery agent settings
- Audit trail and other settings

We can also set a parameter that defines the behavior of the server when performing the `<assign>` activity. This parameter is called `Relax BPEL4WS1.1 spec assign rules`. BPEL specification sets certain rules by assignments (discussed in Chapter 4). For example, null assignments are not allowed by default. If these rules are too restrictive for us, we can change the behavior by setting this parameter to `true`. However, doing this is not recommended because it can hinder portability of BPEL processes. The default value of this parameter is `false`.

**Performance Tuning**

The above-mentioned parameters on stale check, threads and load factor, cache size, etc. affect the performance of the Oracle BPEL Server. Together with the runtime performance statistics (Statistics tab) and thread allocation statistics (Threads tab), they can be used to tune the performance. The runtime performance statistics provide comprehensive data about the execution time of processes and a breakdown of times by activities, as shown in the following screenshot:
The thread allocation statistics provide information on the usage of threads and their allocation, and on the number of requests on BPEL processes, as shown in the next screenshot:

When creating a BPEL process test instance, the Oracle BPEL Server provides an option through which we can perform stress tests. Stress tests enable us to monitor the performance and to do load testing of processes. With the performance statistics, we can identify the possible bottlenecks and optimize the performance. To perform a stress test we simply select the Perform stress test option in the BPEL Processes/Initiate tab, as shown in the following screenshot. We then have to specify the number of concurrent threads allocated to the process, number of loops for the test, and the delay between invocations. We also have to select whether to clear statistics before running the stress test. In this way we can identify the most appropriate number of threads:
Note that the default download edition of Oracle BPEL Process Manager on Windows platform bundles Oracle Lite as the database, which will not yield meaningful results for a stress test. The BPEL Server should be configured to use a production-quality database (Oracle, SQL Server, or DB2) before doing stress testing. Please refer to Oracle documentation and to http://otn.oracle.com/bpel for description how to configure the Oracle BPEL Process Manager for a database server other than Oracle Lite.

Domains and Administration

BPEL Console can also be used for server administration. The BPEL administration can be accessed if we follow the Goto BPEL Admin link on the main logon page or go to the URL http://localhost:9700/BPELAdmin/. We have to supply the administration password, which is initially oracle. The BPEL Admin has three major functions:

- Administration of server-related parameters and administration passwords (Server)
- Managing BPEL domains (BPEL Domains)
- Threads statistics (Threads)
Administration of Server-Related Parameters

The administration of server-related parameters includes four tabs: Configuration, Password, Logging, and WSIF:

Within Configuration we can modify various server parameters, including the location of WSDLs, ports of legacy servers accessed through Connect, J2EE container platform (OC4J, JBoss, or WebLogic), workflow mode, which specifies how human interactions are archived, authentication parameter, data sources, BPEL SOAP server URL, and a list of WSIF providers. We can also specify whether users need to supply a password when they log on to a BPEL domain and whether they can see a list of available domains.

Here we can also change the BPEL Server mode from production to development. In production mode, if we try to redeploy a process with the same name and revision tag, the server asks whether we would like to replace an existing version or use a different revision number. In development mode, a process is redeployed without any questions asked. The server mode can be changed by modifying the productionServer property. The default is true, which is used for production mode. For development mode it should be changed to false.
Under the Password tab, we can change the administrator password. This should be done immediately after the installation.

The Logging tab enables us to set the logging level for various server components, such as infrastructure, services, WSIF, etc. Several logging levels are available, such as off, fatal, error, warn, info, debug, and all.

The WSIF tab shows the supported WSIF bindings. We will say more on WSIF bindings in Chapter 6.

Managing BPEL Domains

We have already mentioned that Oracle BPEL Server is organized into domains. Since domains enable us to logically organize business processes, using several domains is recommended in real-world scenarios. After installation, the default domain is created automatically. Additional domains can be created under the BPEL Domains tab, as shown in the following screenshot:

If we follow the Create New BPEL Domain link, we can create new domains. We have to specify the domain ID, password, and JNDI (Java Naming and Directory Interface) addresses for regular and transactional data sources:
With this, we have concluded our review of the BPEL Console. In the next section, we look at the BPEL Designer.

**Graphical Development with BPEL Designer**

Writing BPEL processes by hand, as we have done in previous chapters, can become time consuming and complex. Therefore, Oracle has developed the BPEL Designer, which enables graphical development of BPEL processes. Instead of writing BPEL code we can develop processes in a graphical environment where we can add activities using drag-and-drop. BPEL Designer simplifies development and makes it faster. In addition to drag-and-drop modeling it provides a browser through which we can locate web services. It also provides a copy assistant, an XPath editor, the ability to compile and deploy a process on the BPEL Server with one mouse click, and several other useful tools. Oracle BPEL Designer natively supports BPEL version 1.1, so we can also use it for developing BPEL processes that will be deployed to any other BPEL server.

BPEL Designer is available for two environments:

- JDeveloper BPEL Designer, integrated with Oracle JDeveloper
- Eclipse BPEL Designer, a plug-in for the Eclipse platform
Both versions of BPEL Designer offer comparable functionality, but differ in several details. Some of the differences are based on the differences of the host environments, JDeveloper and Eclipse respectively. JDeveloper BPEL Designer also provides support for Oracle-specific functions, particularly for sensors and adapters (file, FTP, database, etc.). Let us now have a look at both versions of BPEL Designer, first at JDeveloper and then Eclipse.

**JDeveloper BPEL Designer**

The BPEL Designer integrated with JDeveloper is particularly useful for developers who use JDeveloper as their main development environment, or plan to do so. Before you start using JDeveloper BPEL Designer, get familiar with JDeveloper. Please refer to Oracle documentation on JDeveloper for more information or visit [http://www.oracle.com/technology/products/jdev/index.html](http://www.oracle.com/technology/products/jdev/index.html).

After creating a new BPEL Process project, or loading or importing an existing project, we see the JDeveloper BPEL Designer window. It shows the graphical structure of the process and partner links:

BPEL Designer also provides the source-code view where we can edit the BPEL code directly. Changes made in source view are reflected immediately in the BPEL Designer visual representation, and vice versa.
Please notice that it is not the intention of this section to provide in-depth instructions on using JDeveloper BPEL Designer. Rather we will highlight the most important features including importing existing BPEL processes, adding partner links, creating variables, adding activities, copying variables, entering XPath functions, and using XSLT Mapper and Validation Browser. For detailed instructions on using JDeveloper please refer to Oracle documentation.

Importing Existing BPEL Processes

First, let us look at how we can import existing BPEL processes into JDeveloper. After selecting the workspace in Applications – Navigator we choose Import from the File menu. In the Import dialog window we select BPEL Process. Next, we have to enter the source file name and path and the project name. Optionally we can select the project directory:

After clicking OK, the project is imported into the selected workspace and validated. JDeveloper BPEL Designer in version 10.1.2 requires that the process name specified in the name attribute of the <process> root element is equal to the name of the file in which the source is stored. If the names are not equal we have to manually add the BPEL and WSDL files to the project.

Partner Links and Web Services

To add partner links we have to select the Process Activities in the Component Palette (upper right side of the screen). From there we select the PartnerLink and drag-and-drop it to the Partner Links section of the main design window. Alternatively, we can right-click in the main design window and use the context menu. A dialog window opens where we have to enter the partner link name and the details, including the WSDL location, partner link type, and both roles:

If we do not know the exact location of the WSDL, we can use WSDL Chooser by clicking on the flashlight icon (second from the left). The WSDL Chooser uses UDDI and WSIL (Web Services Inspection Language) for locating web services. The following screenshot shows the selected web service on the local BPEL server:

### Variables

To add variables to the BPEL process we have two choices. We can add them manually, as we will show in this section, or we can use automatic variable creation, as we will see in the next section. Let us look at how to add variables manually or edit existing variables. In the BPEL Structure navigator window (lower left side of the screen) we navigate to the variables. Then we
either click an existing variable or right-click to create a new variable. A dialog window opens where we have to specify the variable name and type:

To select the variable type we can use the Type Chooser by clicking the flashlight icon. Here we can navigate through message types in partner links or project WSDL files and select the appropriate type:
If we are using a simple type, we can choose from the predefined XML types:

- `<assign>` activity: Provides support for parallel execution
- Java Embedding: Enables inclusion of Java code into BPEL
- Notification: Wizard for using the Notification service
- Transform: Enables access to the XSLT engine and XSLT Mapper
- User Task: Wizard for using the Workflow service

We will explain these Oracle-specific activities in Chapter 6.

Now we will show how to add an `<assign>` activity. First, we select the Assign process activity from the Component Palette and drag-and-drop it to the process. Then a dialog window opens, where we have to specify the copy rules, as shown on the screenshot opposite. Here we can edit existing copy rules or add new rules:
Copy Rule Editor

For adding new copy rules in the `<assign>` activities or editing existing rules, we can use the Copy Rule editor, which simplifies the procedure. Using this editor we can create the `<from>` and `<to>` expressions by navigating through the variables trees. The Copy Rule editor supports all forms of `<assigns>` with a variable, expression, XML fragment, or partner link being the source and/or the destination of the assignment. When copying variables we can navigate thorough the variables tree, as shown on the screenshot overleaf. We can also enter expressions using XPath Expression Builder (shown in the next section), enter XML, or edit/create a new partner link:
XPath Expression Builder

To simplify the creation or editing of XPath expressions JDeveloper BPEL Designer provides the XPath Expression Builder. It is accessible from all dialogs where XPath expressions need to be entered by pressing on the XPath Expression Builder icon. In the upper-left section, we can navigate through BPEL variables. In the lower-left section, we can select various functions. XPath Expression Builder supports XPath 1.0, XPath 2.0, and Oracle-specific XPath extensions functions, which are covered in Chapter 6. In the lower-right section, we can access operators. Pressing Ctrl-Space in the Expression Body window we get context-sensitive help. The screenshot opposite shows how we can create XPath expressions using the builder.
XSLT Mapper

In BPEL processes we often need to transform XML (stored in variables). For example, we have to modify the vocabulary to adapt the output of one service to the input of another. We might also want to transform XML to other markup languages. Instead of using XPath, Oracle BPEL Process Manager provides a built-in XSLT engine. To compose XSLT stylesheets JDeveloper BPEL Designer provides a built-in XSLT Mapper, which simplifies the mapping definition considerably.

To activate the XSLT engine from a BPEL process we need to add the transformation activity to the process. We do this by dragging-and-dropping the Transform activity from the Process Activities Component Palette to the process. This activity simply calls the built-in XSLT engine, explained in the Extension Functions and Activities section in Chapter 6.

After double-clicking the activity, a Transform window appears. Here we select the source and the target variable name along with the corresponding parts. Then we enter the XSLT file name:
After pressing the icon to create a mapping (second from the left) the XSLT Mapper window opens. We can drag-and-drop the elements from the left side to the right side. We can also use XPath functions, which we can select from the Component Palette in the upper right side of the window (Oracle provides support for XPath 1.0, some XPath 2.0, and Oracle-specific functions). As the result, an XSLT stylesheet is generated, which takes care for the transformation:
With XSLT Mapper we use several functions that influence the mapping and also transform the source data. We can also use an Auto Map feature, which tries to map attributes automatically. In this way we can develop XSLT transformations easily without being familiar with the XSLT language. From more information on XSLT please refer to [http://www.w3.org/TR/xslt](http://www.w3.org/TR/xslt). For more information on XSLT Mapper please refer to Oracle documentation.

### BPEL Validation Browser

To simplify finding and correcting errors and warnings JDeveloper provides a BPEL Validation Browser. We can access it by clicking the tool icon (second from the left) in the main design window. The BPEL Validation Browser window opens. In the screenshot, we can see that there is one error in the BPEL process, related to the assign activity where no copy rules are specified. The browser also suggests the solution:
Building and Deploying

BPEL processes can be built and deployed directly from JDeveloper. To deploy a BPEL process we have to right-click on the Integration Content in the System – Navigator window and select Deploy. Next, we see a list of connections to the BPEL servers—in our case we use the default connection LocalBPELServer. We can then select the domain to which we would like to make the deployment, as shown in the screenshot below. To start the deployment we have to enter the password for the domain:

Alternatively, we can select the Invoke Deployment Tool. Then a dialog window appears where we have to select the connection to the server, the address, domain name and password, and version:
From JDeveloper we can access the Deployment Descriptor Properties window, where we can set various deployment properties, which is usually easier than editing $bpel.xml$ manually. Please refer to the Process Descriptor section of this chapter for more information on $bpel.xml$. To access the Deployment Descriptor Properties window we have to click the icon (first from the left in the main design window):

![Deployment Descriptor Properties](image)

### Eclipse BPEL Designer

The Eclipse BPEL Designer is an Eclipse plug-in. Therefore, we should get familiar with the basics of the Eclipse platform (http://www.eclipse.org/). For detailed installation instructions please refer to Oracle documentation and tutorials, which can be downloaded from http://www.oracle.com/technology/products/ias/bpel/index.html.

The following screenshot shows the main Eclipse BPEL Designer screen with the opened Travel process. The main window shows the overview of the process with partner links and global XML variables exposed. This view is specific to Eclipse and not found in JDeveloper:
From this view we can add partner links and variables to our process. BPEL Designer also provides the source-code view where we can edit the BPEL code directly. Changes made in source view are reflected immediately in the BPEL Designer visual representation, and vice versa.

**Partner Links and Web Services**

To add a partner link to the process, we simply click Add Partner Link, located in the lower right corner of the main window. After entering the partner link name and WSDL location, the designer will help us in selecting the partner link types and roles. This is shown in the following screenshot:
If we do not know the exact location of the WSDL, we can use the UDDI browser, through which we can locate and select the appropriate web service. The web services can be located on the local BPEL server or on remote BPEL servers. We can use the WSIL interface or use the built-in services, which we will discuss later. The following screenshot shows the view of the root WSIL document:

**Variables**

By following the Add XML Variable link, we can add a global variable to the process. Variables in BPEL processes can be defined globally or within scopes. Adding them with the Variable Wizard requires us to fill out the following form:
We have to enter the variable name and type, which can be a message type, element, or XML Schema type. We can select the available types from the drop-down list.

**XML Type Browser**

From the overview process window we can access the XML Schema Type Browser by clicking the View XML Types link. The XML Schema Type Browser allows us to browse schemas in different namespaces. It also shows the XPath expressions, which are particularly useful for `<assign>` activities and complex schemas:

Process Map

Let us now switch from Overview to the Process Map view. BPEL Designer will show the graphical representation of the process, similar to what we have seen in the BPEL Console. We can expand activities to see more details. We can also click on each activity to see the details in the right-hand window—the BPEL Inspector.

To add new activities to the process we can drag-and-drop them from the BPEL Palette in the lower left-hand window. In Eclipse, the BPEL Palette does not show all activities in one pane. Rather, they are organized into five groups:

- Core BPEL: Contains the standard BPEL activities
- BPEL Extensions: Contains Oracle extension activities such as `<flow>` and `<exec>`
- Notifications: Provides access to the Notification service
- Experiment: Provides access to other extension activities, such as XSLT transformations
- User Interaction: Provides access to the Workflow service

The Process Map view allows us to control whether we want to develop standard BPEL processes or include specific extensions. The screenshot below shows the Process Map view with opened BPEL Palette:
Copy Rule Editor

Similar to JDeveloper, Eclipse also provides the Copy Rule editor, which simplifies adding new copy rules in the \texttt{<assign>\textunderscore\textgreater\textless\textgreater} activities or editing existing rules. Using the editor we can create the \texttt{<from>\textless\textgreater} and \texttt{<to>\textgreater\textless\textgreater} expressions by navigating through the variable trees. The Copy Rule editor supports \texttt{<assigns>\textless\textgreater} with a variable, expression, or XML fragment as the source. We can use the Variable & Query Picker to navigate to the variable, part, and query:

![Copy Rule Editor](image)

Function Wizard

To enter an expression we can open the BPEL Function Wizard. This wizard helps us to compose XPath expressions. In the first step, we have to select a function from the list of available functions. Note that it also offers some Oracle-specific functions (with namespace prefixes other than bpws), which are discussed in Chapter 6:
After selecting the function, we have to fill in the required parameters. In our case we have selected the `bpws:getVariableData()` function and have to specify the variable, part, and query in the second step:

**Building and Deploying**

Eclipse BPEL Designer offers direct compilation and deployment on the Oracle BPEL Server. This can be done from the toolbar or from the BPEL menu. In addition to building and deploying, we can also validate our project and open the BPEL Console:
Summary

In this chapter, we have become familiar with the Oracle BPEL Process Manager 10g. We have seen that Oracle BPEL Process Manager is a Java-based BPEL server that provides comprehensive support for BPEL. We have provided an overview of BPEL Process Manager architecture and reviewed the major features, such as dehydration, version control, and clustering. We have become familiar with how processes can be deployed, with the deployment descriptor, and with the BPEL Console. Using the BPEL Console we can deploy, test, manage, and debug BPEL processes. We have also become familiar with domains.

For BPEL development, Oracle provides an integrated graphical development environment called BPEL Designer in two versions: JDeveloper BPEL Designer and Eclipse BPEL Designer. BPEL Designer offers several tools that simplify development, such as the Copy Rule editor, XPath Expression builder, XSLT Mapper, and BPEL Validation browser.
In the next chapter, we will discuss the advanced features provided by the Oracle BPEL Process Manager 10g Release 2. We will overview the extension functions and activities provided by the BPEL Process Manager. We will take a detailed look at dynamic parallel flows and dynamic partner links. We will explain and demonstrate how BPEL processes can be integrated with resources other than web services using Web Services Invocation Framework. We will also look at Java embedding. Finally, we will overview three important services provided by Oracle BPEL Process Manager: Notification service for sending and receiving asynchronous notifications using email and other channels, Workflow service for including human interaction into BPEL processes, and Identity service for user authentication and authorization.

Business Process Execution Language for Web Services (BPEL, WS-BPEL, or BPEL4WS) is the new standard for defining business processes with composition of services. It is the cornerstone of Service Oriented Architecture (SOA). With its ability to define executable and abstract business processes it opens new doors in business process management and represents the top-down approach to the realization of SOA.

BPEL is supported by the majority of software vendors including Oracle, Microsoft, IBM, BEA, SAP, Hewlett-Packard, Siebel, and others. Most of them already have products that support BPEL; others will follow soon.

This book explains the BPEL standard, provides a step-by-step guide to designing and developing business processes in BPEL, defines the role of BPEL in SOA, and discusses how BPEL relates to the web services stack and to other standards. It also covers two important BPEL servers—the Oracle BPEL Process Manager and Microsoft BizTalk Server. The book presents the service-oriented approach to business process definition using web services, which enables us to develop loosely coupled solutions.

What This Book Covers

Chapter 1 provides a detailed introduction to BPEL and Service Oriented Architecture (SOA). It discusses business processes and their automation, explains the role of BPEL, web services, and Enterprise Service Buses (ESB) in SOA, provides insight into business process composition with BPEL, explains the most important features, compares BPEL to other specifications, provides an overview of BPEL servers, and discusses the future of BPEL.

Chapter 2 provides a detailed introduction to the Web Services Technology Stack. It discusses the important standards and specifications for using BPEL and implementing SOA with web services, such as WS-Security, WS-Addressing, WS-Coordination, WS-AtomicTransaction, WS-BusinessActivity, WS-Reliable Messaging, etc.

Chapter 3 discusses the composition of web services with BPEL. The chapter introduces the core concepts of BPEL and explains how to define synchronous and asynchronous business processes with BPEL. The reader gets familiar with BPEL process structure, partner links, sequential and parallel service invocation, variables, conditions, etc.

Chapter 4 goes deeper into the BPEL specification and covers advanced features for modeling complex business processes. Advanced activities, scopes, serialization, fault handing, compensations, event handling, correlation sets, concurrent activities and links, process lifecycle, and dynamic partner links are covered in detail.

Chapter 5 explains how to use the Oracle BPEL Process Manager for deploying and executing business processes defined in BPEL. It describes the server architecture, tools, features, and common approaches for managing and debugging BPEL processes. The chapter also looks at graphical development of BPEL processes using Oracle BPEL Designer for JDeveloper and for Eclipse.

Chapter 6 takes a detailed look at the advanced features of the Oracle BPEL Process Manager including extension functions, dynamic parallel flows, Web Services Invocation Framework, Java embedding, Notification service, Workflow service, Identity service, and Oracle BPEL Server APIs.

Chapter 7 discusses MS BizTalk Server 2004 and its support for BPEL. It explains how to develop business processes in BizTalk and export them to BPEL. It also explains how to import BPEL processes into BizTalk and how to use the Orchestration Designer tool to define processes graphically, and compares BizTalk and BPEL constructs.

Appendix A provides a syntax reference for BPEL version 1.1. The appendix covers standard BPEL activities and elements, functions, attributes, and faults.

Where to buy this book

Free shipping to the US, UK, Europe, Australia, New Zealand and India.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.