Oracle® Fusion Middleware

Business Process Composer User's Guide for Oracle Business Process Management

11*g* Release 1 (11.1.1.6.3)

E15177-10

August 2012

Provides information for process analysts and developers interested in using Oracle Business Process Composer.

ORACLE

Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management, 11*g* Release 1 (11.1.1.6.3)

E15177-10

Primary Author: Steven Leslie

Contents

2 Overview of Business Process Design

3 Introduction to Oracle Business Process Composer

Part II
Using Oracle Business Process Composer

4 Working with BPM Projects

5
Working with Processes and the Process Editor

6
Modeling Business Processes with Oracle BPM

7 Working with the Project Life Cycle

8 Using Oracle Business Rules

Part III
Advanced Business Process Composer Functionality

9
Advanced Business Process Composer Functionality

10
Working with Data Objects and Expressions

11 Working with Human Tasks

12 Performing Administrative Tasks

BPMN Flow Object Property Reference 1

Preface

This guide describes the Oracle Business Process Composer application.

Intended Audience

This guide is intended for process analysts who use the Business Process Composer application to create and edit the business processes and Oracle BPM projects used to created process-based applications using the Oracle Business Process Management Suite.

This guide is also intended for process developers who must use Business Process Composer. See Section 1.2, "Oracle BPM User Personas" for more information on these user personas.

This manual assumes that you have basic knowledge of business process design and are familiar with Business Process Management Notation (BPMN) 2.0.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following Oracle resources:

Oracle Business Process Management

See the following for more information about the Oracle BPM Suite:

- *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*
- *Oracle Fusion Middleware Business Process Management User's Guide*

Oracle SOA and BPM Suite Installation and Administration

- *Oracle Fusion Middleware Installation Guide for Oracle SOA Suite*
- *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle BPM Suite*
- *Oracle Fusion Middleware High Availability Guide*

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

What's New in This Guide for Release 11.1.1.6.*x*

For Release 11.1.1.6.*x*, this guide has been updated in several ways. The following table lists the sections that have been added or changed. If a feature was not available in the first release of 11.1.1.6.*x*, the last columns denote which documentation release contains the update.

For a list of known issues (release notes), see the "Known Issues for Oracle SOA Products and Oracle AIA Foundation Pack" at http://www.oracle.com/technetwork/middleware/docs/soa-aiafp-knownissuesindex-364630.html.

Sections	Changes Made	11.1.1.6.0	11.1.1.6.1
Chapter 4 Working with BPM Projects			
Section 4.4, "Sharing Projects with Other Users"	Section revised to describe new security model for sharing BPM projects and determining which Business Process Composer users can view, edit, or deploy a project.	X	
Section 4.5.5, "How to Share a Project with Other Users"	Section added to describe the procedures to share a BPM project with other Business Process Composer or Oracle BPM Studio users.	X	
Section 4.5.11, "How to View the History of Changes Made to a Project"	Section added to describe how to view the history of changes made to a BPM project.	X	
Section 4.5.12, "How to View and Edit Project Properties"	Section added to describe how to view and edit BPM project properties.	X	
Section 4.5.13, "How to Mark a Project as a Favorite"	Section added to describe how to mark a project as a favorite.	X	
Section 4.7.1, "Introduction to Project Roles"	Section added to describe how to create and use project roles. Project roles are used to model the users or groups that are responsible for performing the work performed by you business process	X	
Chapter 5 Working with Processes and the Process Editor			

Sections	Changes Made	11.1.1.6.0	11.1.1.6.1
Section 5.2.3, "Introduction to the BPMN Component Palette"	Section added to describe the BPMN component palette.	X	
Section 5.5, "Working with Business Catalog Components"	Section revised to describe procedures for working with the BPMN component catalog.	X	
Section 5.6, "Working with Draft Processes"	Section added to describe how to work with draft processes.	X	
Section 5.8, "Importing and Exporting Process Models"	Section revised to describe how to import process models and convert them to BPMN.	X	
Chapter 7 Importing BPMN Processes from a BPA Repository			
Chapter 7, "Working with the Project Life Cycle"	Chapter added to document more advanced functionality related to the project life cycle. This includes sections on importing and exporting projects, working with project templates, and deploying projects.	X	
Section 7.5, "Deploying a Project"	Section added to describe how to generate a deployment plan. This section was also revised to reflect changes to project deployment.	X	
Chapter 9 Advanced Business Process Composer Functionality			
Chapter 9, "Advanced Business Process Composer Functionality"	Chapter added to document Business Process Composer functionality that is primarily used by process developers.	X	
Chapter 9, "Advanced Business Process Composer Functionality"	Chapter added to document Business Process Composer functionality that is primarily used by process developers.	X	
Chapter 12 Performing Administrative Tasks			
Section 12.2, "How to Assign Global Roles"	Section revised to reflect changes to security model for projects shared using the BPM repository.	X	
Appendix B Preparing Processes for Import into BPM			
Appendix B, "Preparing Processes for Import into BPMN"	Appendix added to describe best practices when importing processes into BPM. This appendix provides information on how to prepare Visio and XPDL process flows for conversion to BPMN.	X	

Part I

Introduction to Oracle Business Process Composer

This part provides a general introduction to the Oracle Business Process Composer application. It also provides an overview of the Oracle BPM Suite and shows how Business Process Composer is used within the overall process development life cycle.

This part contains the following chapters:

1

Oracle Business Process Management Suite Overview

This chapter provides a general overview of the Oracle Business Process Management (BPM) Suite.

This chapter includes the following sections:

1.1 Introduction to the Oracle Business Process Management Suite

The Oracle BPM Suite provides an integrated environment for developing, administering, and using business applications centered around business processes.

The Oracle BPM Suite provides the following:

- Enables you to create process models based on standards with user-friendly applications. It enables collaboration between process developers and process analysts. Oracle BPM supports BPMN 2.0 and BPEL from modeling and implementation to run time and monitoring.
- Enables process analysts and process owners to customize business processes and Oracle Business Rules.
- Provides a web-based application for creating business processes, editing Oracle Business Rules, and task customization using predefined components.
- Expands business process management to include flexible, unstructured processes. It adds dynamic tasks and supports approval routing using declarative patterns and rules-driven flow determination.
- Enables collaboration by providing integration with Process Spaces which drives productivity and innovation.
- Unifies different stages of the application development life cycle by addressing end-to-end requirements for developing process-based applications. Oracle BPM unifies the design, implementation, run time, and monitoring stages. Oracle BPM

enables different personas to participate through all stages of the application life-cycle.

See Section 1.2, "Oracle BPM User Personas" for more information on the user personas defined for the Oracle BPM Suite.

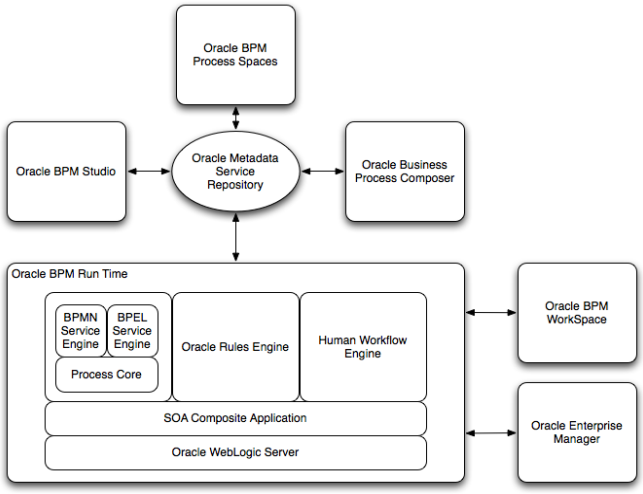
The Oracle BPM Suite provides a seamless integration of all stages of the application development life cycle from design-time and implementation to run-time and application management.

The Oracle BPM Suite is layered on the Oracle SOA Suite and shares many of the same product components, including:

- Business Rules
- Human Workflow
- Oracle Adapter Framework for Integration

Figure 1–1 shows a high-level architectural view of the Oracle BPM Suite.

Figure 1–1 The Oracle BPM Suite



This figure depicts the general architecture of the Oracle BPM Suite. Each of these components displayed in this diagram are described in the following sections.

****

Section 1.3, "Oracle BPM Suite Components" provides more information on each of these components shown in Figure 1–1.

1.2 Oracle BPM User Personas

Different stages of the application development life cycle require interaction from different types of users. Table 1–1 outlines the typical users of Oracle BPM Suite and their responsibilities. It also lists the components of the Oracle BPM they would use to perform their work.

These user personas are used within the examples in this guide.

Table 1–1 Oracle BPM User Persona

User Persona	Description
Process Analyst	Process analysts are responsible for creating the initial flow of a business process and documenting its steps. This also includes identifying and defining the KPIs and high level rules that define the routing artifacts of the business process. This persona may also perform simulations to calculate and estimate ROI. Process analysts typically use the Oracle Business Process Analysis (BPA) Suite or Business Process Composer to create process models. They may also use process analyst role within Oracle BPM Studio.
Process Developer	Process developers are responsible for implementing the process models created by process analysts. Each step in the process requires an implementation. The process developer is responsible for integrating the business process with back-end applications like databases. Process developers typically use Oracle BPM Studio to model and implement the components of a business application. They may occasionally use Business Process Composer for modeling basic processes.
Business Administrator	Business administrators are responsible for administering the BPM infrastructure. Typical activities include the installation and setup of BPM environments and the overall management of the BPM Engines that are hosting business processes. This persona may be delegated responsibilities for administering the organization structure assets like users, groups, organizational units, calendars and holidays. The main tool used by business administrators is the Oracle Enterprise Manager and automated tools like Ant. Business administrators also use Process Workspace to manage organizational units, role assignments and perform other activities like creating workflow advanced routing declarations
Process Owner	Process owners are responsible for controlling and managing deployed business processes. They are responsible for the overall supervision of the running business process. They often use metric analysis tools like dashboards to understand the current state of the managed business processes. Process owners typically use Process Workspace. They also use Business Process Composer to change the behavior of a process by editing Oracle Business Rules. They may also use the Oracle BAM console to view metrics dashboards.
Process Participant	Process participants are the people who use the business applications created with the Oracle BPM Suite. Process participants typically use Process Workspace or Process Spaces.

1.3 Oracle BPM Suite Components

This section provides a general description of the major components of the Oracle BPM Suite. See Section 1.5, "Introduction to the Application Development Life Cycle" for information on how these components interact within the application development process.

1.3.1 Process Modeling and Implementation

This section describes the applications and components used to model and implement business processes and process-based business applications.

The Oracle BPM Suite provides two primary applications for modeling and implementing business processes.

Note: Oracle BPM can also integrate business processes created using the Oracle Business Process Analysis (BPA) Suite. See Section 1.4, "Oracle Business Process Analysis (BPA) Suite" for more information.

1.3.1.1 Oracle BPM Studio

Oracle BPM Studio is a component of the Oracle BPM Suite that provides a user-friendly environment where process analysts can create business process models and run process simulations. Oracle BPM Studio supports Business Process Management Notation (BPMN) 2.0.

Oracle BPM Studio also enables process developers to create working process-based applications. These applications are Oracle BPM projects that are integrated as SOA composite applications.

You can use Oracle BPM Studio to implement business processes with other Oracle components such as adapters, human workflow and business rules. You can then deploy these processes to Oracle BPM run time.

Oracle BPM Studio is a part of the Oracle JDeveloper IDE. Oracle BPM Studio enables IT users to use a single integrated tool to model and edit business processes, implement the required IT elements, and deploy applications to the run-time environment.

Oracle BPM Studio also provides a BPM role that enables business users to use a simplified version of Oracle JDeveloper that only displays functionality relevant to process design.

See the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for more information.

1.3.1.2 Oracle Business Process Composer

Oracle Business Process Composer is a web-based application that enables business users to collaborate with process developers and designers. It provides a user friendly environment for editing processes and process templates created in Oracle BPM Studio.

Process developers can create a catalog of preconfigured components such as services, tasks, and rules in Oracle BPM Studio. This catalog can be included in project templates that process analysts can use to create new projects using Oracle Business Process Composer.

After creating a project based on a project template, process analysts can incorporate business catalog elements and perform other required edits defined by the project template. Process analysts can then deploy these projects to the Oracle BPM run time.

Business Process Composer also enables process analysts to create new projects. These are initial versions of a project that can be used by process developers who use Oracle BPM Studio to add further implementation details and refinement.

Business Process Composer also enables you to edit Oracle Business Rules at run time. This is important because policies tend to evolve faster that business processes.

See Chapter 3, "Introduction to Oracle Business Process Composer" for more information.

1.3.1.3 Oracle Metadata Service (MDS) Repository

Oracle Metadata Service (MDS) provides a repository that is used to store date about applications deployed within an Oracle Fusion Middleware environment. Oracle BPM uses this repository to store information about deployed applications.

Oracle BPM also uses a separate MDS partition to share projects and project templates between process analysts and process developers. Figure 1–1, "The Oracle BPM Suite" shows how the MDS repository fits within the overall Oracle BPM architecture.

1.3.1.4 Oracle BPM Projects

Oracle BPM projects are containers for the business processes and related resources used to create a process-based business application. An Oracle BPM project can contain the following:

- Organizational data
- Activity guides
- BPMN process models
- Business catalog
- Simulation models
- Other resources

Oracle BPM projects are deployed at run time as SOA composite applications. For more information on working with projects and SOA composite applications see the following documentation:

- "Working with Projects and Project Templates" in *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*
- "Working with Projects and Project Templates" in *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management*
- *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*

1.3.2 Oracle BPM Run Time Components

Oracle BPM run time is responsible for controlling deployed applications. Oracle BPM run time includes the following components.

1.3.2.1 Oracle BPM Engine

The Oracle BPM Engine provides a run-time environment for running business processes. It provides native support for both BPMN and BPEL processes.

The BPM engine is composed of three separate components:

- BPMN Service Engine

  The BPMN engine provides an environment for running BPMN processes.

- BPEL Service Engine

  The BPEL engine provides an environment for running BPEL processes.

- Process Core

  Provides engine functionality that is shared by the BPMN and BPEL engines. Some of the key functionality performed by the process core includes:

  – Manage security

  – Generate audit trails

  – Invoke services

  – Manage persistence

1.3.2.2 Oracle Human Workflow

Many end-to-end business processes require human interactions with the process. For example, humans may be needed for approvals, exception management, or performing activities required to advance the business process. The human workflow service provides features such as:

- Task routing to users, groups or application roles.
- Deadlines, escalations, notifications, and other features required for ensuring the timely performance of a task.
- Task forms for presentation of tasks to end users through a variety of mechanisms, including a workspace and portals.
- Organization, filtering, prioritization, dispatching rules and other features required for end users to productively perform their tasks.

1.3.2.3 Oracle Business Rules

Oracle Business Rules are a component of the Oracle SOA Suite that enable dynamic decisions at run time allowing, among other features, applications to rapidly adapt to regulatory and competitive pressures. This increased agility is possible because process analysts using Oracle Business Rules can create and change business rules that are separated from the application code. By using Oracle Business Rules, process analysts can change business rules without stopping business processes. Also, externalizing business rules enables process analysts to manage business rules directly, without involving process developers.

1.3.2.4 Oracle WebLogic Application Server

Oracle WebLogic Server is an application server that provides a platform for creating and running J2EE-compliant applications.

1.3.2.5 Oracle Enterprise Manager

The Oracle Enterprise Manager is a web-based application that enables system administrators to control and manage applications running on the Oracle SOA Suite. Enterprise Manager enables business administrators to configure and manage business applications and process instances.

1.3.3 Oracle BPM Suite Process Participant Applications

The following sections describe the components of the Oracle BPM Suite that are used by process participants to perform their day-to-day work. These applications enable process participants to interact with running business applications managed by Oracle BPM run time.

1.3.3.1 Oracle Business Process Management Workspace (Process Workspace)

Process Workspace enable process participants to interact with the applications you create using Oracle BPM. The Process Workspace user interface provides tabs for each of the following:

- **Tasks**: This page enables process participants to view and work with their assigned tasks.
- **Process Tracking**: This page enables process participants to view running process instances.
- **Standard Dashboards**: This page provides out-of-the-box dashboards for monitoring process performance, task performance, and workload.
- **Custom Dashboards**: This page enables process participants to define and use custom dashboards based on the measurement data generated by process instances.

Process Workspace also enables business administrators to configure and maintain organizations and roles. See the *Oracle Fusion Middleware Business Process Management User's Guide* for more information.

1.3.3.2 Oracle Business Process Management Process Spaces (Process Spaces)

Process Spaces is a collaborative workspace built on top of Oracle WebCenterSpaces and enables more productive BPM by increasing collaboration.

See the *Oracle Fusion Middleware Business Process Management User's Guide* for more information.

1.3.4 Other Oracle BPM Suite Components

The following sections describe other components of the Oracle BPM Suite.

1.3.4.1 Process Analytics

Business Process Analytics enables process participants to monitor the performance of a running process-based applications. It measures the key performance indicators defined in a BPM project and stores them in a database. Process participants and analysts can view the metrics stored in the process analytics databases using Process Workspace dashboards or Oracle BAM.

1.3.4.2 Guided Business Processes

Guided Business Processes enable process analysts and developers to group the interactive activities in a business process into a set of milestones that are meaningful to the process participants. They outline the steps the process participants have to complete, hiding the complexity of the business process.

See "Introduction to Guided Business Processes" in *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*

1.4 Oracle Business Process Analysis (BPA) Suite

The Oracle BPA Suite is a separate Oracle product suite based on the Aris platform from IDS Scheer. The Oracle Business Process Analysis (BPA) Suite provides comprehensive modeling, analysis and simulation capabilities for enterprise wide business processes. Oracle BPA supports capturing business architecture artifacts such

as strategic objectives, goals, higher level KPIs, risks and controls, and conceptual models such as value chain diagrams.

Additionally, the Oracle BPA Suite supports the following:

- Alignment of business processes with business strategy.
- Service discovery & linking to business processes. Drives service requirements for the Oracle SOA Suite.
- Loading and creating simulation scenarios which allow you to determine optimal resource allocation. Simulations allow you to perform throughput analysis, activity based costing and resource utilization. Additionally, you can create simulation analysis reports for easy analysis of simulation results.
- Comprehensive version management including check-in, check-out, and change management capabilities.

The business architecture defined by the Oracle BPA Suite is the formal link between strategic objectives and the actual business applications created using Oracle BPM. The Oracle BPA Suite supports modeling of Business Architecture artifacts such as strategy maps, goals, objectives, risk and controls and linking them to business processes.

This provides the ability to prioritize efforts, justify decisions, and trace activities of the business process improvement initiatives to strategic goals of the business, hence improving business/IT alignment. It provides tremendous value as it offers a clear understanding of which BPM projects to undertake, which processes are currently most strategic to the company, and which services are most aligned with business strategy.

The Oracle BPA Suite complements the functionality of the Oracle BPM Suite by adding orthogonal dimensions to the modeling phases including organization goals. See the *Oracle BPA Quick Start Guide* for more information

Processes created in the Oracle BPA Suite can be imported into the Oracle BPM Suite. Using Oracle BPM Studio, you can integrate your business process with other Oracle technologies including adapters, business rules, and human tasks.

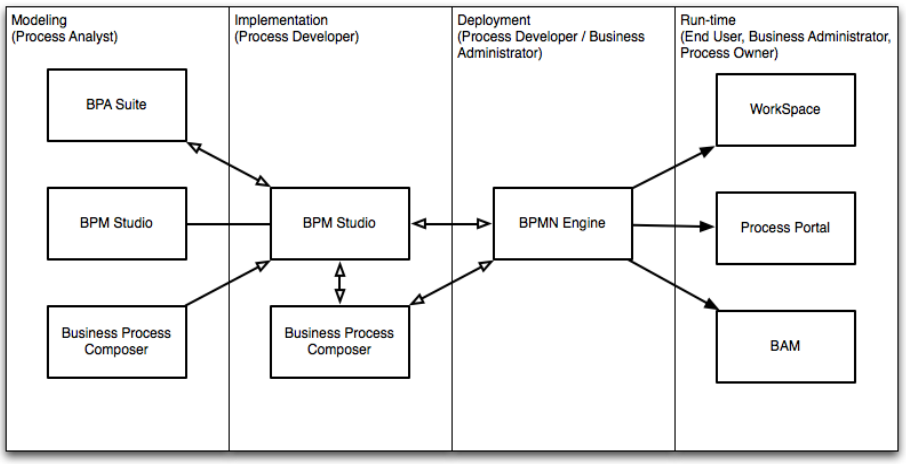
See the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for more information on using business processes created in Oracle BPA within Oracle BPM Studio.

1.5 Introduction to the Application Development Life Cycle

This section outlines the stages of the development life cycle of an Oracle BPM application. It describes how different components of Oracle BPM are used within each stage.

Figure 1–2 lists the four stages of the application development life cycle, the user personas applicable to each stage, and the Oracle BPM tools and applications that are used.

Figure 1–2 Stages of the Oracle BPM Application Development Life Cycle



This figure shows each of the components of the Oracle BPM Suite within the four stages of the application development life cycle.

****

1.5.1 Process Modeling

The first stage of the application development life cycle is process modeling. During this stage a process analyst creates process models based on real-word business processes and problems.

Oracle BPM provides three distinct tools for modeling business processes. Each tool has a different role within the Oracle BPM Suite. The tool you use depends on your business requirements, the stage of the application development cycle, and your user persona.

- Oracle BPM Studio

  Oracle BPM Studio runs on the Oracle JDeveloper IDE platform. Oracle BPM Studio provides a process analyst role that displays a simplified set of JDeveloper functionality that focuses on designing process models.

  Oracle BPM Studio enables process analysts and process developers to design and implement detailed process flows that are deployed to Oracle BPM run time and run as working applications. Additionally, detailed process flows from Oracle BPA Suite or Business Process Composer can be opened in Oracle BPM Studio for further implementation, then deployed to the Oracle BPM run time.

- Oracle Business Process Composer

  Business Process Composer is a collaboration tool that enables process analysts to collaborate with process developers.

- Oracle Business Process Analysis (BPA) Suite

  The Oracle BPA Suite enables you to create robust models of your business processes from high-level models of your entire organization down to lower-level business processes that you can implement as running processes.

See Section 1.6, "Oracle BPM Use Cases" for more information on how each of these tools fit within the typical Oracle BPM uses cases. See Section 3.2, "Overview of the Application Development Life Cycle" for more information on how Oracle Business

Process Composer and Oracle BPM Studio interact within the application development life cycle.

1.5.2 Implementation

After process analysts model business processes, process developers are responsible for creating business applications based on these models. Using Oracle BPM Studio, process developers implement reusable services and integrate other business systems.

Implementation may include the following types of tasks generally performed by process developers:

- Data mapping and transformation
- System fault handling
- Designing and implementing user interfaces using Oracle Human Workflow.
- Designing Oracle Business Rules
- Creating dashboards

After a process developer finishes the implementation of the application, it is compiled and deployed like other SOA composite applications. It can be compiled and deployed using Oracle BPM Studio.

1.5.3 Deployment

Deployment is the process of transferring an Oracle BPM project from the development environment to the run-time environment. This can be either a testing or production run-time environment.

After finishing the integration of business processes with back-end systems and reusable services, process developers create and compile a working process-based application. The application is then deployed to Oracle BPM run time.

Oracle BPM Suite contains the following typical scenarios for deploying to Oracle BPM run time:

- Deployment directly from Oracle BPM Studio

  Applications created with Oracle BPM can be deployed directly to the run-time environment like any other SOA composite application. This is typically performed by a process developer using BPM Studio within a test or development environment.

  See the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite* for more information on deploying SOA composite applications.

- Deployment directly from Business Process Composer

  Oracle BPM enables you to deploy projects directly to the same run-time environment where Business Process Composer is installed. You can specify an approval workflow that must be completed before the project is deployed.

  You can deploy from Business Process Composer when it is installed in the same server infrastructure as Oracle BPM run time. Deploying from Business Process Composer enables process analysts to easily deploy and test process-based business applications. The is generally done in a testing environment.

- Deployment using an exported SAR file

Oracle BPM Studio and Business Process Composer enable you to export applications using a SAR file. The SAR file can be deployed to run time by business administrators using Oracle Enterprise Manager.

In a production environment, this is generally how applications are deployed.

- Deployment Using the WebLogic Scripting Tool (WLST)

  Oracle BPM provides customized WLST commands for managing and deploying Oracle BPM projects.

1.5.4 Oracle BPM Run Time

After an application is deployed, the run-time environment makes the Oracle BPM application available to process participants based on the roles assigned in the organization where the business processes were deployed. This stage is divided into two distinct functions:

- User interaction

  Process participants and process owners are responsible for interacting with the running application using Process Workspace.

  Process analysts and owners can also monitor the process and revise Oracle Business Rules at run time using Business Process Composer.

- Process management and monitoring

  Process owners are responsible for monitoring and maintaining running processes using Process Workspace. Process analysts and owners use Oracle Process Analytics to monitor the real-time performance of business processes.

- Process creation

  Process participants who have the necessary permissions can create new processes using Process Workspace or Oracle Business Process Management Process Spaces

- System administration

  Business administrators are responsible for maintaining running business applications and the overall run-time infrastructure using Oracle Enterprise Manager and the Oracle Weblogic Server administration console.

1.6 Oracle BPM Use Cases

This section describes typical uses cases of the Oracle BPM Suite from process modeling to run time.

1.6.1 Use Case: Using BPM Studio to Create Project Templates

This use case involves using Oracle BPM Studio to create project templates. These templates are used by process analysts to create new projects using Business Process Composer.

Typical Workflow for Using Oracle BPM Studio to Create Project Templates

1. Determine the business requirements. (process analyst)
2. Model the required business processes using Oracle BPM Studio (process analyst / process developer).

   Process analyst can use the Process Analyst role in Oracle JDeveloper.

3. Implement the processes by integrating each element of the process with back-end systems and reusable services. (process developer).
4. Create a project template using Oracle BPM Studio (process developer).
5. Publish the project template to the Oracle BPM MDS repository (process developer).
6. Create a new Oracle BPM project based on a project template (process analyst).
7. Implement the required reusable services defined by the project template (process analyst).
8. Deploy the project to Oracle BPM run time (process analyst).

1.6.2 Use Case: Using BPM Studio to Model Processes and Deploy an Application

This use case involves using Oracle BPM Studio to create process models. These models are used to create working business applications that are deployed to the Oracle BPM run time.

Typical Workflow for Using Oracle BPM Studio to Model Processes

1. Determine the business requirements (process analyst).
2. Model the required business processes using Oracle BPM Studio (process analyst / process developer).

   Process analyst can use the Process Analyst role in Oracle JDeveloper.
3. Run a simulation to test and improve process performance. (process analyst / process developer).
4. Implement the processes by integrating each element of the process with back-end systems and reusable services. (process developer).
5. Compile the Oracle BPM project as a composite application (process developer).
6. Deploy the application to the run time environment (process developer, business administrator).
7. Interact with the deployed processes as part of a running business application (process participants, process owner).
8. Maintain and monitor the running process-based applications (business administrator, process owner).

1.6.3 Use Case: Using Business Process Composer to Create Projects

This use case involves creating new projects using Business Process Composer. These projects are then shared with process developers who import them into Oracle BPM Studio, where they perform further refinement and implementation.

Typical Workflow for Using Business Process Composer to Create New Projects

1. Create a new project using Business Process Composer (process analyst).
2. Provide implementation details for the project and prepare the process-based business application for deployment (process developer).
3. Create project templates and publish them to the Oracle BPM Metadata Store repository using Oracle BPM Studio (process developer).
4. Create a project based on a the project template (process analyst).

5. Edit the project as defined by the edit policies of the project template (process analyst).
6. Deploy the project to Oracle BPM run time (process analyst, process administrator).

1.6.4 Use Case: Using Business Process Composer to Revise Oracle Business Rules

The use case involves using Business Process Composer to edit Oracle Business Rules at run time. After an application is deployed, process analysts and owners can open the deployed project and edit Oracle Business Rules.

Typical Workflow for Using Business Process Composer to Revise Oracle Business Rules

1. Model a set of business processes (process analyst).
2. Implement and deploy an application (process developer).
3. Edit the Oracle Business Rules at run time using Business Process Composer (process owner).

1.6.5 Use Case: Using The Oracle Business Process Analysis Suite to Model Your Business Processes

This use case involves using the Oracle BPA Suite to model your business processes. These processes can be imported into Oracle BPM Studio.

Typical Workflow for Using the Oracle Business Process Analysis Suite and Oracle BPM Suite to Model Processes

1. Determine the business requirements (process analyst).
2. Design your business architecture by capturing strategic objectives, process maps, and value chain diagrams using Oracle BPA (process analyst).
3. Perform strategic analysis to determine potential process candidates for Oracle BPM projects (process analysts).
4. Design detailed process flows for the process candidates identified above (process analyst).
5. Import your process models into Oracle BPM Studio (process analyst, process developer).
6. Implement the processes by integrating each process component with back-end systems and reusable services (process developer).
7. Deploy the business processes as a BPM project to the run time environment (process developer, business administrator).
8. Interact with the deployed processes as part of a business application (process participant, process owner).
9. Maintain the processes (business administrator, process owner).

2

Overview of Business Process Design

This chapter provides an overview of business process design. It provides a basic introduction to Business Process Management Notation (BPMN). This introduction is primarily designed to introduce the BPMN-specific terminology used within this guide.

See Chapter 6, "Modeling Business Processes with Oracle BPM" for more detailed information about Oracle's implementation of BPMN 2.0.

This chapter also describes the Sales Quote example. This project is used throughout the examples within the Oracle BPM documentation set.

This chapter includes the following sections:

- Section 2.1, "Introduction to Business Process Management Notation (BPMN)"
- Section 2.2, "Introduction to the Sales Quote Example"

2.1 Introduction to Business Process Management Notation (BPMN)

This section provides a brief introduction to Business Process Management Notation (BPMN). It is primarily designed to introduce the BPMN terminology used throughout this guide.

2.1.1 What is Business Process Management Notation (BPMN)

Business Process Management Notation (BPMN) is an industry standard notation for defining business processes. Oracle BPM supports BPMN 2.0.

For general information on BPMN see: `http://www.bpmn.org`. For information on the BPMN functional specification supported by Oracle BPM, see `http://www.omg.org/spec/BPMN/2.0/`.

2.1.2 Business Processes

A business process can be generally defined as a sequence of tasks that, after performed, result in a well-defined outcome. As the term business process implies, a business process usually represents work that is performed within the context of a company or organization.

The Sales Quote example project shows an example of a business process. It contains a sequence of tasks that result either in the approval or rejection of a sales quote.

Within the context of Oracle BPM, a business process is also something that can be managed by software. Oracle BPM enables you to model real-world business processes like the Sales Quote example and integrate them within an IT environment.

2.1.2.1 Process Instances

A process instance refers to the specific instance of a business process. While a business process generally defines how an organization performs its work, a process instance refers to the work of a specific person within that organization. In Oracle BPM, this person is referred to as a process participant.

For example, the Sales Quote example shows the overall definition, or model, of a business process, including the roles of the process participants who are responsible for performing the work. It defines how a sales quote is created and approved and defines the types of people responsible for performing that work.

In contrast, a process instance refers to a specific sales quote and the specific people responsible for approving it.

This distinction between process and process instance is important because Oracle BPM enables you to model business processes, convert them into running business applications, and manage the process instances created within those applications.

2.1.2.2 Process Tokens

Process tokens are an abstract concept in BPMN. They refer to the current point of execution within a process. A business process can have multiple tokens that indicate that the process is running in multiple paths.

For example, gateways are often used to split the path of a process. Splitting a process path creates multiple process tokens.

2.1.3 Flow Objects

Flow objects are the BPMN components that represent the work performed within a process. The following sections describe the types of flow objects available in BPMN.

2.1.3.1 Tasks

In Oracle BPM, tasks are used to represent the work performed by a process.

2.1.3.2 Events

Events define something that happens during a process.

2.1.3.3 Gateways

Gateways are used to determine the flow of your process.

2.1.3.4 Sequence Flows

Sequence flows are used to connect flow objects.

2.1.4 Data Objects

While flow objects are used to define the behavior of a business process, data objects are used to define and store the information used by a business process. Data objects are variables that are defined during the modeling and implementation of a process. A process instance uses these variables to store specific information.

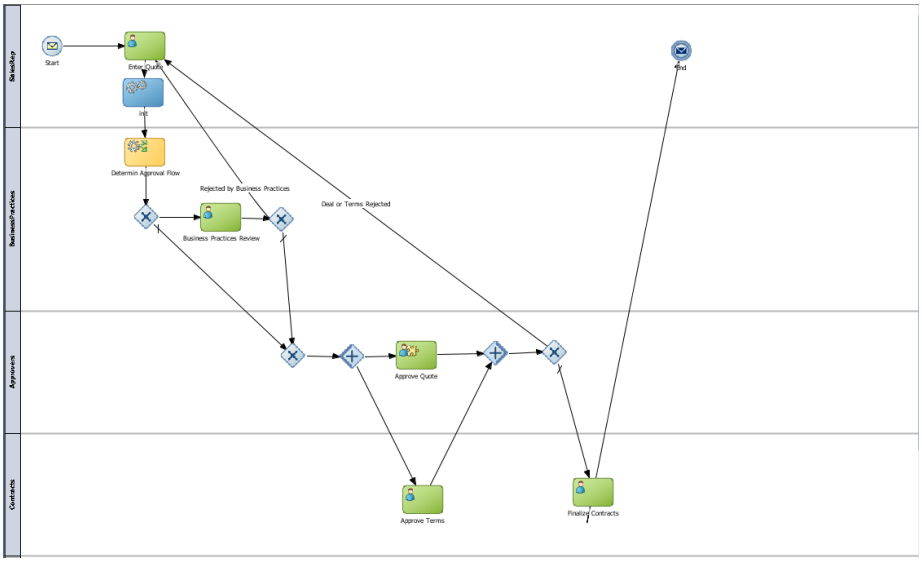
For example, the Sales Quote example defines several data objects used to store information about the sales quote. At run time, the process instance generates and stores specific values for these variables.

2.2 Introduction to the Sales Quote Example

The Sales Quote project provides real-world examples of different Oracle BPM features. This project is used within the Oracle BPM documentation set to provide examples of the features being described.

The Sales Quote example is shown in Figure 2–1.

Figure 2–1 The Sales Quote Example



This graphic shows the BPMN notation for the Sales Quote example. It has four swimlanes labeled as follows from top to bottom they are: SalesRep, Business Practices, Approvers, Contracts.

The rest of the graphic is described in the following sections.

****

2.2.1 Breakdown of the Sales Quote Example

The following sections describe how the Sales Quote example process works. This example can be broken down into the following high-level tasks:

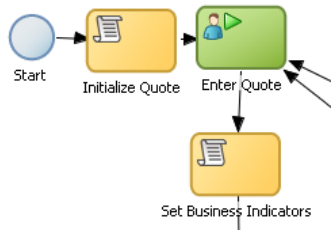
- Section 2.2.1.1, "Initiate Sales Quote"
- Section 2.2.1.2, "Determine Business Practice Review"
- Section 2.2.1.3, "Approve Quote"
- Section 2.2.1.4, "Approvals Outcome"

These high-level tasks are described in the following sections. Within each section, the specific flow objects required to perform each task are detailed.

2.2.1.1 Initiate Sales Quote

The initial flow objects within the Sales Quote project are used to set the initial values for data objects and initiate the process instance as shown in Figure 2–2.

Figure 2–2 Initiate Sales Quote



This graphic shows a start event with a sequence flow extending to a task labeled Initialize Quote.

From the initialize quote task, a sequence flow extends to a user task labeled Enter Quote.

From the Enter Quote task, a sequence flow extends to a task labeled Set Business Indicators.

****

The Initiate Sales Quote Portion Performs the Following:

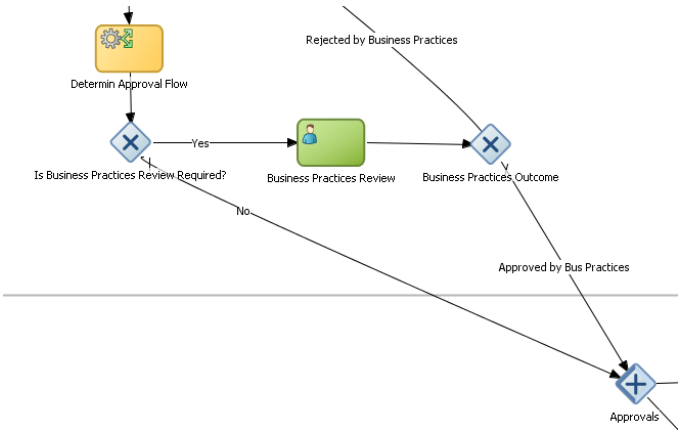
1. Define start point (none start event)
2. Initialize data objects (script task)
3. Initiate the process instance (user task with initiator pattern)
4. Set values for the business indicator data objects (script task)

2.2.1.2 Determine Business Practice Review

The next set of flow objects in the Sales Quote example determine if a review of corporate business practices is necessary. If a review is required, the process proceeds to a part of the process flow that performs the review. If a review is not required, the process proceeds directly to the approval stage.

Figure 2–3 shows the BPMN flow objects used to perform the business practice review.

Figure 2–3 Determine Business Practice Review



In this graphic, a service task labeled Determine Approval Flow has a sequence flow extending downward to an exclusive gateway. From this exclusive gateway, two sequence flows extend.

The first sequence flow, labeled yes, extends to a User task labeled Business Practices Review.

From the Business practices review task, a sequence flow extends to an exclusive gateway labeled Business practices Outcome.

From the first exclusive gateway mentioned earlier, a second sequence flow, labeled No, extends into the next swimlane to a parallel gateway labeled Approvals.

That Approvals parallel gateway has a sequence flow labeled Rejected by Business Practices extending to it from outside the graphic.

****

The following procedure demonstrates how a process path passes through the business practice review.

Determine Business Practice Review

1. Determine Approval Flow (business rules task).

   This stage begins with a business rules task which implements an Oracle Business Rule to determine whether a business practice review is required.

2. Check Approval Flow (exclusive gateway).
   - If Yes, perform Business Practice Review (user task).
   - If No, the process flow proceeds directly to the approve quote stage.

3. Approvals (parallel gateway).

2.2.1.3 Approve Quote

The next set of flow objects in the Sales Quote example defines how the sales quote is approved. After the business practice review is finished, the quote moves to the approval phase as shown in Figure 2–4.

In this example, the approval process is defined by two separate flows that are executed simultaneously. These are:

- Approve the sales quote.

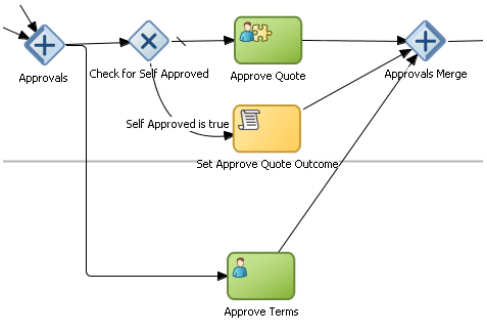
  This process path is also split into two paths. In this example, it is possible for the quote to be self approved, which means that the quote is approved based on certain criteria, or it may require explicit approval from a process participant.

- Approve the terms of the contract related to the sales quote.

  This process path requires a process participant to approve the terms of the sales contract.

After these parallel process paths complete, they are merged. The process path then proceeds to the approval outcome stage.

Figure 2–4 Approve Quote



In this graphic, a parallel gateway, labeled Approvals, has two sequence flows extending from it.

The first sequence flow extends to an exclusive gateway, labeled Check for Self Approved.

From the Check for Self Approved, two sequence flows extend.

The first, a sequence flow with a tick, extends to a user task labeled Approve Quote.

From the Approve Quote task, a sequence flow extends to a parallel gateway labeled Approvals Merge.

From the Check for Self Approved task, a second sequence flow labeled Self Approved is true extends to a task labeled Set Approve Quote Outcome.

From the Set Approve Quote Outcome task, a sequence flow extends to the Approvals Merge parallel gateway.

From the Approvals parallel gateway mentioned earlier, a second sequence flow extends to a User task labeled Approve terms in the swimlane below.

From the Approve Terms task, a sequence flow extends to the Approvals Merge parallel gateway in the swimlane above.

****

Approve Quote

1. Approvals (parallel gateway - split)
   a. Check for self approval (gateway)

      Approve Quote

      Set Approve Quote Outcome

   b. Approve Terms
2. Merge Approve Quote (parallel gateway - merge)

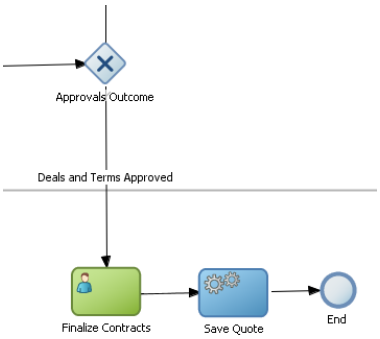
2.2.1.4 Approvals Outcome

The approvals outcome stage represents the final stage of the Sales Quote example. It begins with a check to determine if the sales quote has been approved.

If the sales quote is approved, the process proceeds to the final process flow which proceeds to the end event.

If the sales quote is not approved, the process flow returns to the enter quote task where the quote must be reentered and the process repeats.

Figure 2–5 Approval Outcome



In this graphic, an exclusive gateway labeled Approvals Outcome has a sequence flow labeled Details and terms Approved extending into the next swimlane to a User Task labeled Finalize contracts.

From the Finalize Contracts task, a sequence flow extends to a service task labeled save Quote.

From the Save Quote task a sequence flow extends to an end event.

****

Approval Outcome

1. Approvals outcome (exclusive gateway)

   The approvals outcome is implemented using an exclusive gateway. This gateway contains two outgoing sequence flows which determine the path the process takes out of the exclusive gateway.

   a. Approved

      This is implemented with a default sequence flow.

      Finalize Quote

      Save Quote

      End Event

   b. Rejected: Sends the process flow back to the enter quote.

      This is implemented with a conditional sequence flow. The expression used for this conditional sequence flow determines if the process path continues

3

Introduction to Oracle Business Process Composer

This chapter provides an introduction to the Oracle Business Process Composer application.

This chapter includes the following sections:

- Section 3.1, "Oracle Business Process Composer Overview"
- Section 3.2, "Overview of the Application Development Life Cycle"
- Section 3.3, "Signing On to Oracle Business Process Composer"
- Section 3.4, "Introduction to the Business Process Composer Application Interface"

3.1 Oracle Business Process Composer Overview

Oracle Business Process Composer is a web-based application that enables process analysts to create and customize business processes. These processes are contained within an Oracle BPM project. Business Process Composer enables process analysts to easily collaborate with process developers who use Oracle BPM Studio to create process-based business applications.

See the following section for more information on the use cases for Business Process Composer.

3.1.1 Oracle Business Process Composer Use Cases

There are three typical use cases for Oracle Business Process Composer:

- Create, edit and deploy projects based on project templates.

  Project templates are created in Oracle BPM Studio and are stored within the Oracle BPM Metadata Service partition. Using Business Process Composer, you can use these templates to create new or modify existing business processes.

  You can then re-publish them to the Oracle BPM Metadata Service partition or deploy them to the Oracle BPM run time.

  See Section 7.2, "Using BPM Project Templates" for information on using project templates in Business Process Composer. For information on creating project templates see the *Oracle BPM Modeling and Implementation Guide*

- Create new projects.

Business Process Composer enables you to create new BPM projects. These projects can be shared between Oracle BPM Studio and Business Process Composer.

New projects created in Business Process Composer often do not contain any of the implementation details required for an Oracle BPM application. Process developers can use Oracle BPM Studio to add the required implementation.

From Business Process Composer, you can save the project to the Oracle BPM repository where process developers can add the required technical implementation using Oracle BPM Studio. You can also import and export projects between Business Process Composer and Oracle BPM Studio.

- Edit business rules for in-flight processes

  You can use Business Process Composer to edit Oracle Business Rules in a running application. Oracle Business Rules enable you to define business policies within your process using the business rules task. You can easily change these policies at run time without having to remodel your business process or redeploy your business application.

3.2 Overview of the Application Development Life Cycle

Oracle Business Process Composer is a collaboration tool that enables process analysts to easily interact with process developers. It provides a user-friendly interface for creating process-based business applications based on project templates. These templates are created using Oracle BPM Studio.

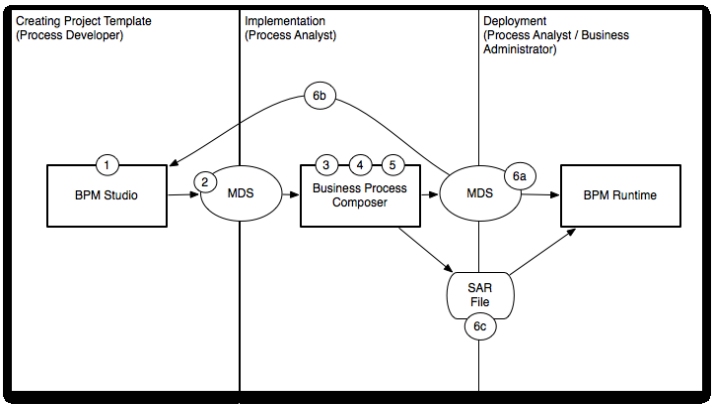
Business Process Composer also provides an environment for creating new BPM projects. These projects can easily be shared with process developers who are using Oracle BPM Studio to implement business processes as part of a business application.

The following sections describe typical scenarios by showing the interactions between different Oracle BPM components and the persona, including process analysts and process developers. However, the exact workflow you use depends on your business needs. There may be multiple iterations where process analysts and developers collaborate to create and refine a business process.

3.2.1 Workflow: Create Projects Based on Project Templates

Figure 3–1 shows a typical workflow for using Oracle BPM Studio at the beginning of the development cycle to create process templates. These templates are used to create new Oracle BPM projects. These projects can edited by process analysts using Business Process Composer.

Figure 3–1 Using BPM Studio to Create Project Templates



This graphic is a rectangle divided into three sections. The first section is labeled Creating Process Template (Process Developer), the second is labeled Implementation (Process Analyst), and the third is labeled Deployment (Process Analyst/Business Administrator)

The Creating Process Template (Process Developer) section contains a rectangle with the number 1 and labeled Oracle BPM Studio. From this rectangle, an arrow numbered two extends into the oval labeled MDS.

From the oval labeled MDS and arrow continues section to a rectangle numbered 3 and 4, and labeled Business Process Composer.

From this Business Process Composer rectangle in the Implementation section, two arrows extend.

The first arrow extends to an oval on the divider between the Implementation and Deployment sections. The oval is numbered 6a and labeled MDS.

From the oval labeled MDS, an arrow extends into the Deployment section to a rectangle labeled BPM Runtime.

The second arrow from the BPM Studio rectangle extends to a rectangle on the divider between the Implementation and Deployment sections. The rectangle is numbered 6b and labeled SAR File.

From the SAR File rectangle, an arrow extends to the BPM Runtime rectangle in the Deployment section mentioned earlier.

****

The following steps describe each stage of this workflow:

1. Create a project template using Oracle BPM Studio (process developer)
2. Publish the project template to the Oracle BPM Metadata Service (MDS) partition (process developer)
3. Create a project based on a project template using Business Process Composer. (process analyst)
4. Edit the processes within the project based on the edit policies defined in the template (process analyst).
5. Validate the project (process analyst).
6. Deploy the project or return the project to the process developer.

a. Deploy the project directly to Oracle BPM run time (process analyst, business administrator). This may require an approval workflow.

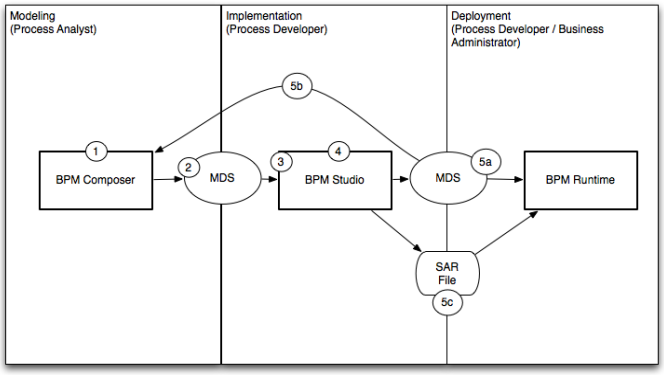
b. Republish the project to the Oracle BPM MDS partition. (process analyst) Republishing the project enables you to share it with other process analysts or with process developers who are responsible for implementing your business processes within an overall application.

c. Export the project as an SAR file (process analyst). This file can be deployed to Oracle BPM run time (process administrators).

3.2.2 Workflow: Creating New Projects

Figure 3–2 shows a typical workflow for using Business Process Composer to perform the initial process modeling stages of the application development life cycle. This workflow involves using Business Process Composer to new BPM projects which can be opened in Oracle BPM Studio where process developers complete the implementation.

Figure 3–2 Using Oracle Business Process Composer to Create New Projects



This graphic is a rectangle divided into three sections. The first section is labeled Modeling (Process Analyst/Process Developer), the second is labeled Implementation (Process Developer), and the third is labeled Deployment (Process Developer/Business Administrator)

On the divider between the Modeling and Implementation sections, there is a rectangle numbered 1, 2, and 3, and labeled BPM Studio.

From the BPM Studio rectangle, an arrow extends to an oval on the divider between Implementation and Deployment. The oval is numbered 4 and labeled MDS.

An arrow extends from this oval to a rectangle in the Deployment section labeled BPM Runtime.

****

The following steps describe each stage of this workflow:

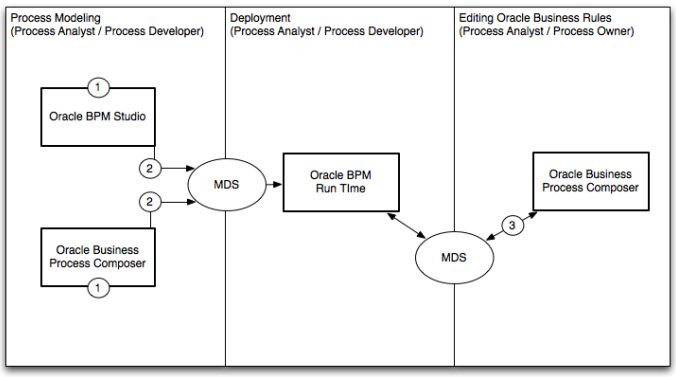
1. Create a new project using Business Process Composer (process analyst)
2. Save the project to the Oracle BPM (MDS) repository (process analyst)
3. Open the project in Oracle BPM Studio (process developer)
4. Implement the project as part of a process-based business application. (process developer)

5. Deploy the project to run time or create a project template.
   a. Deploy the process to run time (process developer, business administrator), or
   b. Save the application as a project template (process developer)
   c. Export the project as a SAR file (process analyst), which is then deployed to Oracle BPM run time (business administrator).

3.2.3 Workflow: Editing Business Rules at Run Time

Figure 3–3 shows the workflow for creating and deploying a process-based business application, then using Business Process Composer to edit the Business Rules at run time.

Figure 3–3 Using Oracle Business Process Composer to Edit Oracle Business Rules at Run Time



This graphic is a rectangle divided into three sections. The first section is labeled Process Modeling (Process Analyst, Process Developer), the second is labeled Deployment (Process Analyst, Process Developer), and the third is labeled Editing Oracle Business Rules (Process Analyst/ Process Owner)

The Process Modeling (Process Analyst, Process Developer) section contains two rectangles each with the number 1. They are labeled Oracle BPM Studio andOracle Business Process Composer.

From each rectangle, an arrow numbered two extends to the border between the Process Modeling and Deployment section to an oval labeled MDS.

From this MDS oval, an arrows extends to a rectangle labeled Oracle BPM Run Time in the Deployment section.

From the rectangle labeled Oracle BPM Run Time, an arrow extends to another oval labeled MDS that lies on the boundary between Deployment and Editing Oracle Business Rules.

From this oval labeled MDS, and arrow extends to a rectangle within the Editing Oracle Business Rules labeled Business Process Composer.

****

1. Model your business processes and create a process-based business application using Oracle BPM Suite. This can include a combination of the workflows defined in the previous sections. (process analyst / process developer)
2. Deploy the application to Oracle BPM run time. (process analyst / process developer)

3. Edit business rules during run time using Business Process Composer (process analyst).

3.3 Signing On to Oracle Business Process Composer

Before signing on to Business Process Composer your business administrator must provide the following:

- **URL**: The location of your Business Process Composer installation.
- **Username**: The username you use to access Business Process Composer.
- **Password**: The security credential you use to access Business Process Composer.

> **Note:** Oracle Application Server Single Sign-On is enabled by default in Oracle BPM Suite. OracleAS Single Sign-On enables you to use one sign on session to access multiple web-based applications. If OracleAS Single Sign-On is enabled and you have previously signed on to another application, the Business Process Composer sign on screen may not appear.

3.3.1 How to Sign On to Oracle Business Process Composer

The following procedures describe how to sign on to the *Oracle Business Process Composer* application.

To sign on to Oracle Business Process Composer

1. Go to the Business Process Composer URL.

   Figure 3–4 shows the sign-on screen that appears after the Oracle BPM application loads.

Figure 3–4 Oracle Business Process Composer Sign-on Screen

This figure shows the Business Process Composer sign-on screen. It contains two text fields, one labeled username and the other labeled password. Below these text fields is a button labeled Login.

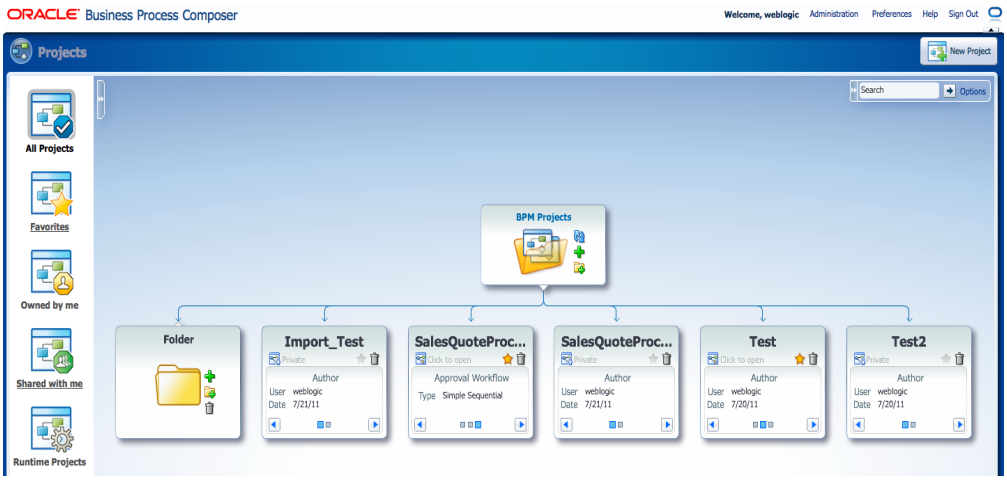
****

2. Enter your username and password, then click **Login**.

3.4 Introduction to the Business Process Composer Application Interface

The Business Process Composer user interface is shown in Figure 3–5.

Figure 3–5 The Oracle Business Process Composer Application User Interface



This graphic shows the Oracle Business Process Composer application user interface.

****

The major areas of the Business Process Composer user interface are described in the following sections.T

3.4.1 Introduction to the Business Process Composer Toolbar

The Business Process Composer toolbar provides access to the general application functionality. Figure 3–6 shows the application toolbar. This toolbar is located in the upper right-hand corner of the application. It is available from each page of the application.

> **Note:** The **Administration** menu item is only available to Business Process Composer administrators. This item does not appear for other users.

Figure 3–6 The Business Process Composer Toolbar

This image displays the Application Toolbar.

****

The application toolbar provides access to the following:

Table 3–1 The Business Process Composer Application Toolbar

Toolbar element	Description
Username	Displays the name of the current user. This text field is read-only.
Administration	Provides access to the Business Process Composer administration page. This item is only visible to users who have been granted the Administrator security role. See Chapter 12, "Performing Administrative Tasks" for more information.
Preferences	Enables you to configure general application preferences.
Help	Displays an HTML version of this guide.

Table 3–1 (Cont.) The Business Process Composer Application Toolbar

Toolbar element	Description
Sign Out	Sign out the current user.
Network connectivity	Shows if there is network activity. When there is no network activity, the icon appears as an "O."

3.4.2 Introduction to the Business Process Composer Welcome Page

The Business Process Composer application welcome page enables you to view and work with the project in the BPM repository. Figure 3–7 shows the project Welcome page. The components of this page is described in the following sections.

Figure 3–7 The Project Browser

This graphic displays the project browser.

****

Chapter 4, "Working with BPM Projects" for information on using the application welcome page to work with BPM projects.

3.4.2.1 Project Views

Project views enable you to view the project browser based on certain criteria. You can select a project view by selecting its icon from the left-hand side of the project welcome page as shown in Figure 3–7. The different types of project views are described in Table 3–2.

Table 3–2 Project Views

Project view	Description
All projects	Shows all projects within the BPM repository that the current user has permissions to view or edit. For Business Process Composer administrators this shows all projects that are not private. Private projects are only visible to the project owner.
Favorites	Shows only the projects marked as favorites by the current user.
Owned by me	Shows only the projects owned by the current user.
Shared with me	Shows only the projects shared with the current user.
Runtime projects	Shows only projects that have been deployed to runtime.

3.4.2.2 Project Browser

The project browser provides hierarchical view of the BPM repository, including projects and project folders as shown in the center of figure Figure 3–7.

The project browser also enables you to create new projects and project folders and delete projects.

3.4.2.3 Control Panel

The control panel enables you to control how projects and project folders are displayed in the project browser. The project browser control panel is shown in Figure 3–8.

Figure 3–8 The Project Browser Control Panel

This graphic displays the Project Browser Control Panel.

****

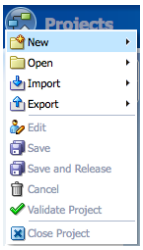
3.4.2.4 Search

The search field enables you to search for projects within the repository based on name, author, or description. The search field is available in the upper-right hand corner of the project welcome page as shown in Figure 3–7.

3.4.3 Introduction to the Business Process Composer Main Menu

The application main menu provides access to frequently used commands and functionality. This menu is accessible from the application and project welcome pages. It is accessible by clicking on the icon shown at the top of figureFigure 3–9.

Figure 3–9 Application Main Menu



This graphic displays the Application Main Menu.

****

The application main menu provides access to the menu items describe in Table 3–3.

Table 3–3 Main Application Menu Items

Menu item	Description
New	Enables you to create a new BPM project.

Table 3–3 (Cont.) Main Application Menu Items

Menu item	Description
Open	Enables you to open BPM projects stored within the BPM repository.
Import	Provides functionality for importing projects into the BPM repository.
Export	Enables you to export projects to the local file system.
Edit	Switches the current project to edit mode.
Save	Saves changes made to the current project.
Save and release	Saves changes made to the current project and releases the lock. This enables other uses who have access to begin editing the project.
Cancel	Releases the lock held on the project without saving changes.
Validate project	Validates the project.
Close project	Closes the current project.

Part II

Using Oracle Business Process Composer

This part describes how to use Business Process Composer to model your business processes. It includes a general overview of the application. It also contains a detailed description of Oracle's BPMN 2.0 implementation.

This part contains the following chapters:

4

Working with BPM Projects

This chapter describes how to create and use projects using Oracle Business Process Composer.

This chapter includes the following sections:

4.1 Introduction to Oracle BPM Projects

Projects are the core element of an Oracle BPM application. BPM projects contain the resources used to create and support a business application. These include business processes and components of the business catalog including data objects, services, Business Rules, Human Tasks, and roles.

You can create new projects directly in Business Process Composer or you can create and edit projects based on project templates created using Oracle BPM Studio.

Oracle BPM projects promote collaboration between process analysts and process developers. Projects can be shared between Business Process Composer and Oracle BPM Studio using the BPM MDS repository. See Section 4.2, "Introduction to the Oracle BPM Repository" for more information.

Projects can also be validated and deployed to run time using Oracle Business Process Composer. See Section 7.5, "Deploying a Project" for more information.

See Section 3.2, "Overview of the Application Development Life Cycle" for information on how projects are created, edited and shared within the development life cycle.

4.1.1 Introduction to Project Components and Resources

Each project contains one or more business processes and may include other resources used by the business processes or application. The latter include reusable resources that allow you to connect your application to other applications and systems.

Note: Oracle BPM Studio enables you to view, create, and edit all elements of an Oracle BPM project. Some of these are not visible from Business Process Composer.

See the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for more information.

4.1.1.1 Editable Project Resources

Using Business Process Composer you can create and edit the following project resources:

- Processes

  Processes are the core element of a business application. A process is a related set of tasks or activities. An Oracle BPM application can contain one or more processes. Business Process Composer enables you to create and edit BPMN processes.

  From the BPM Project navigator you can create new processes and edit existing ones. See Chapter 5, "Working with Processes and the Process Editor" for information on creating and working with processes.

- Human Tasks

  Oracle Business Process Composer enables you to create and edit human tasks. Human tasks are used to define how users interact with your process-based applications.

  From the BPM Project navigator you can create new human tasks and edit existing ones. See Chapter 11, "Working with Human Tasks" for more information. After creating a human task, they are accessible within the business catalog.

- Activity Guide

  An activity guide is part of Guided Business Processes that enables you to define milestones for a project. Each project contains one activity guide where you can define project milestones. Business Process Composer enables you to use milestones previously defined in a project template or create new milestones.

  See Section 4.6, "Using Guided Business Processes to Create Project Milestones" for more information on creating and using project milestones.

- Oracle Business Rules

  Oracle Business Rules are statements that describe business policies or describe key business decisions. In Business Process Composer, Oracle Business Rules are editable components of a project, but they also appear as part of the business catalog.

  See Chapter 8, "Using Oracle Business Rules" for information.

  Note: You cannot create new business rules using Business Process Composer. You can only edit business rules that were added to the project using Oracle BPM Studio.

4.1.1.2 The Business Catalog

Project templates may include resources defined in the business catalog. These are reusable services created by process developers using Oracle BPM Studio.

Using Business Process Composer, process analysts can implement these reusable services within a BPM process. Although you cannot create or edit these components in Business Process Composer, you can assign them to specific BPMN artifacts.

Reusable project resources are accessible from the component palette. The reusable resources in the business catalog are:

- Services

  Services are used to connect a BPMN process with other processes, systems, and services, including BPEL processes and databases.

  Using Business Process Composer you can create new services based on web services. See Section 9.1, "Working with Services" for more information.

  You can also use other types of services that were created as part of the project template.

  Within a BPMN process, services are implemented by assigning the service to a service task. See Section 6.4.1, "Introduction to the Service Task" for more information.

- External references

  References are the interfaces that you can use to define the interface of your BPMN processes. References are used for implementing message events and send and receive tasks with an interface.

  See Section 6.4, "Communicating With Other Processes and Services" for more information.

- Human Tasks

  Human tasks enable you to define how end users interact with your BPMN processes. You can add human tasks to your business process by using the user task. You can assign a human task from the business catalog to each user task in your business process. See Section 6.3, "Adding User Interaction to Your Process" for information on using human tasks within a BPMN process.

- Oracle Business Rules

  Oracle business rules are statements that describe business policies or describe key business decisions. Business rules are integrated into a process using the business rules task.

  See Chapter 8, "Using Oracle Business Rules" for information on working with Oracle Business Rules using Business Process Composer. See Section 6.5.2, "Introduction to the Business Rule Task" for more information on using the business rules task within a BPMN process.

4.1.1.2.1 Components of the Business Catalog that Can Be Edited or Created

Table 4–1 lists the components of the business catalog and shows which components can be created or edited using Oracle Business Process Composer.

Table 4–1 Editable Business Catalog Components

Business Catalog Component	Can be created using Business Process Composer?	Can be edited using Business Process Composer?
Business rules	No	Yes
Human tasks	Yes	Yes
Services	Yes	Yes

Table 4–1 (Cont.) Editable Business Catalog Components

Business Catalog Component	Can be created using Business Process Composer?	Can be edited using Business Process Composer?
External References	No	No

4.2 Introduction to the Oracle BPM Repository

The Oracle BPM repository is based on Oracle Metadata Service (MDS). Oracle MDS is a component of Oracle Fusion Middleware that stores information about deployed applications. Oracle BPM uses this repository when applications are deployed to Oracle BPM run time.

Additionally, Oracle BPM also uses a partition in the Oracle MDS Repository to share projects and project templates between Oracle BPM Studio and Business Process Composer. Within the Oracle BPM repository, there are two partitions used to store projects and project templates. These are:

- Public: used to store Oracle BPM projects.
- Templates: used to store Oracle BPM project templates.

The Oracle BPM repository is installed and configured by your system administrator when installing Business Process Composer.

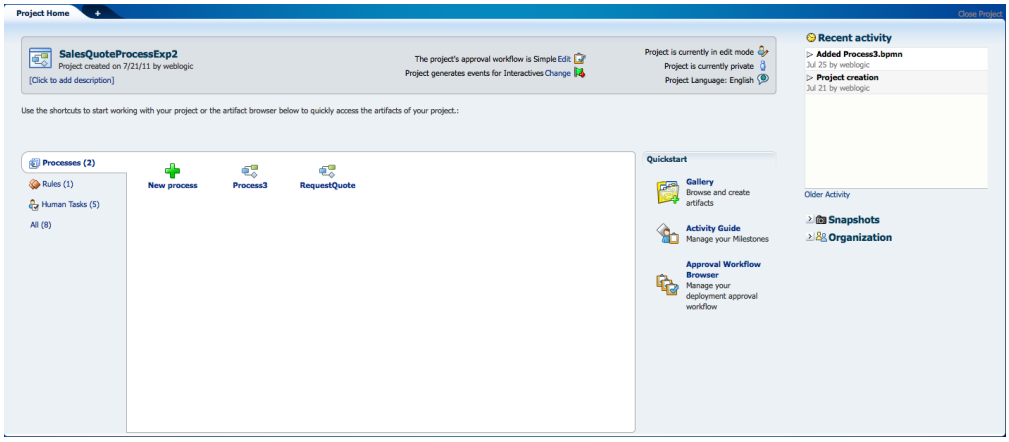
See Section 3.2, "Overview of the Application Development Life Cycle" for more information on how the Oracle BPM repository is used to share projects and project templates between Oracle BPM Studio and Business Process Composer.

See Section 4.5, "Creating and Working with Projects" for more information on how to open projects in the Oracle BPM repository. See Section 7.2, "Using BPM Project Templates" for information on creating and working with projects based on project templates.

4.3 Introduction to the Project Welcome Page

The project welcome page provides access to information about a BPM project as well as access to common project-related features. Figure 4–1 shows the project welcome page for the Sales Quote example project.

Figure 4–1 The BPM Project Welcome Page



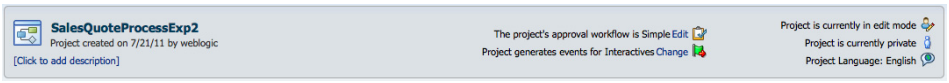
This graphic displays the Project Welcome page.

****

4.3.1 Introduction to the Project Information Pane

The project information panel displays general information about the project. Figure 4–2 shows an example of the project information pane for the SalesQuote demo process.

Figure 4–2 Project Information Pane



This graphic displays the Project Information Pane.

****

This pane displays the following information:

- **Description**: provides a general description of the project.
- **Approval workflow:** Specifies if an approval workflow has been defined for the project.
- **Generate events**: Determines how events are generated for this project.
- **Edit mode:** Specifies whether the current project is being edited.
- **Sharing:** Lists the users or groups among whom the project is being shared.
- **Project language:** Specifies the current language of the project.

4.3.2 Introduction to the Project Component Pane

The project component pane enables you to view and create processes and human tasks and view business rules. Figure 4–3 shows how the project component pane appears in the SalesQuote example project.

Figure 4–3 The Project Component Pane

This graphic displays the Project Component Pane.

****

4.3.3 Introduction to the Quickstart Menu

The Quickstart menu enables you to quickly access the following common functionality within Business Process Composer:

- **Gallery**: enables you to browse and create and browse processes and human tasks. You can also browse business rules.
- **Activity guide**: enables you to create and manage milestones within your processes.
- **Approval workflow browser:** enables you to view and manage the deployment approval workflow.

4.3.4 Introduction to the Recent Activity Browser

The recent activity browser provides a history of the major changes made to the current BPM project.

See Section 4.5.11, "How to View the History of Changes Made to a Project" for information on how to view recent project activity.

4.3.5 Introduction to the Snapshot Browser

The snapshot browser enables you to create and view project snapshots.

Project snapshots enable you to keep a record of the changes made to a project during the development life-cycle. See Section 7.3, "Using Project Snapshots" for more information,

4.3.6 Introduction to the Organization Browser

The organization browser enables you to create and manage project roles.

4.3.7 Introduction to the Oracle Business Process Composer Editors

Editors enable you to work with various elements of an Oracle BPM project, including processes and components of the business catalog. Editors are displayed in the center of the Business Process Composer application.

Each editor appears as a tabbed pane in the Business Process Composer application, allowing you to open multiple resources at the same time.

The different types of editors available in Business Process Composer are described in the following sections.

4.3.7.1 Process Editor

The process editor enables you to view and edit business processes. You can access the process editor by opening a process from the project navigator.

The editor window also contains a component palette. The exact components available depend on the following:

- In projects based on project templates, the component palette contains BPMN flow objects and elements from the business catalog. This includes services, Human Tasks, and Oracle Business Rules that are defined by the project template.
- In new projects created in Business Process Composer, the component palette displays BPMN flow objects as well as business catalog elements that you have

created. Services and human tasks can be created directly in Business Process Composer.

See Chapter 5, "Working with Processes and the Process Editor" for more information.

4.3.7.2 Activity Guide Editor

The Activity Guide editor enables you to view, create, and edit milestones within an activity guide. You can access the Activity Guide editor by clicking the **Activity Guide** link in the Quickstart menu. See Section 4.6, "Using Guided Business Processes to Create Project Milestones".

4.3.7.3 Human Task Editor

The Human Task editor enables you edit human tasks that are included as part of the business catalog. See Chapter 11, "Working with Human Tasks" for more information.

4.3.7.4 Business Rules Editor

The business rules editor enables you to view and edit Oracle Business Rules. You can access the business rules editor by opening a business rule from the project navigator. See Section 8, "Using Oracle Business Rules" for more information.

4.3.7.5 Data Associations Editor

The data associations editor enables you to define the input and output for flow objects that contain implementations. To access the data associations editor, right-click a flow object within your business process and select **Data Associations**.

See Section 10.2, "Working with Data Objects and Data Associations" for information on using data associations.

4.3.7.6 Expression Editor

The expression editor enables you to define the expressions used within data associations and conditional sequence flows. See Section 10.4, "Introduction to Expressions" for information on using expressions and accessing the expression editor.

4.3.8 Introduction to the Supporting Browsers and Editors

Business Process Composer contains additional editors for viewing and editing other project components. These appear in the lower portion of the application window.

> **Note:** These editors are not displayed by default. They appear after performing actions related to each window. However, you can display them manually by clicking the Restore Pane icon in the lower right corner of the Business Process Composer application.

4.3.8.1 Project and Process Validation Browser

The project and process browsers display any validation errors for each individual process or the whole project. See Section 4.5.8, "How to Validate a Project" for more information on project validation.

4.3.8.2 Documentation Editor

The documentation editor enables you create and edit documentation for your processes. See Section 5.7, "Documenting Your Process" for more information.

4.3.8.3 Approval Workflow Browser

The approval workflow browser displays the status of a project within the approval workflow. Once all users defined as approvers have responded, the project can be deployed to Oracle BPM run time. See Section 7.5, "Deploying a Project" for more information.

4.4 Sharing Projects with Other Users

Oracle BPM enables you to share projects with other BPC users. Shared projects are stored in the BPM repository. You can also control who has access to view or edit projects.

4.4.1 Private and Public Projects

Private projects are can only be viewed or edited by the project owner. Public projects are viewable or editable by the project owner as well as other users who have the correct permissions.

See Section 4.5.5, "How to Share a Project with Other Users" for information on how to share projects.

4.4.2 Edit Mode

Shared projects have an edit mode that determines whether you can make changes or not. The edit mode has the following values:

- **Read-only**: The project is open for viewing only. In this mode, some project functionality is unavailable.
- **Edit**: The project is open for editing. In edit mode, you can make changes to the project. When a project is in editing mode, only the user editing the project can make changes. Other users with the correct permissions can view the project, but cannot make changes.

  See Section 4.5.6, "How to Edit a Shared Project" for information on how to set the edit mode.

You can determine the current edit mode for a project in the project information pane as shown in Figure 4–2.

4.4.3 Project Roles

Project roles define who has access to view and make changes to a project. There are three types of project roles defined as follows:

- Owner: When a user creates a project, they become the owner of that project. You can also define another user as the owner of a project. The owner of a project can perform the following:
  - Deploy a project
  - Create a project snapshot
  - Share a project with other users
  - Delete a project
- Editor: An editor can make changes to a project.
- Viewer: A viewer can view a project, but cannot make any changes to it.

4.5 Creating and Working with Projects

The following sections provide information on how to create and use Oracle BPM projects.

4.5.1 How to Access the Project Welcome Page

The project welcome page is displayed by default when you open a project from the application welcome page. When you are editing a component within a project, you can return to the project welcome page by clicking the **Project Home** tab.

4.5.2 How to Create a New Project

Using the project navigator, you can create a new Oracle BPM project. Before creating a new project, you should decide whether to create it based on an existing project template or to create a new project.

New projects are not based on project templates. These projects contain only basic business processes created by process analysts. After you create a new project using Business Process Composer, you can use Oracle BPM Studio to finish the project implementation to publish the project or create project templates.

To create a new project using the project navigator:

The project navigator enables you to create a new project quickly. After creating the project, you can edit additional project properties.

1. Start Oracle Business Process Composer.
2. From the project navigator, click New Project as shown in Figure 4–4.

Figure 4–4 The New Project Dialog

This graphic displays the New Project Dialog.

****

3. Click **Next**.
4. Enter a name for the project, then click **Create Project**.

To create a new project using the main menu:

Creating a new project from the main menu enables you to configure project properties as well as select a project template.

1. Start Oracle Business Process Composer.
2. From the main menu, select **New** then **Project**.
3. Enter a name for the project.
4. Enter the following optional information:
   - **Description**: Provides a description of the project.
   - **Folder**: Enables you to specify a folder in the BPM repository where the project is stored.

- **Use template**: Creates the new project based on a project template. Click Choose to select the project template.

5. Select an optional deployment option from the drop-down list.
6. Click **Finish** to create the new project.

If you created a new project based on a template, the project is created with the required processes and business catalog elements. If you created new project without using a template, you must manually add the required processes.

See Chapter 5, "Working with Processes and the Process Editor" for information on creating and editing processes.

4.5.3 How to Open a Project Using the Application Welcome Page

You can open project directly from the application welcome page by clicking on the name of the project.

4.5.4 How to Open a Project Using the Main Menu

You can open a project using the main menu. This enables you to move directly from one project to another, for example, without having to return to the application welcome page.

To open a project:

1. From the main menu, select **Open**, then **Open Project**.
2. Select the project you want to open, then click **OK**.

4.5.5 How to Share a Project with Other Users

You share projects with other Business Process Composer users using the BPM repository.

To share a project publicly with all users:

1. Open the project you want to share.
2. From the main menu, select **Share**.
3. Select the sharing visibility from the drop-down list.
   - Private
   - Team members only
   - Public
4. Click **Share**.
5. Click **OK**.

To share a project with specific users or groups:

1. Open the project you want to share.
2. From the main menu, select **Share**.
3. Specify the users of groups you want to share the project
   1. Click **Choose**.
   2. From the drop-down list, select the scope:

All:

Users:

Groups:

3. Click **Search**.
4. Select an item from the **Available** column, then click **Move**.
5. Click **OK**.

4. Select a role from the drop-down list.
   - Owner
   - Editor
   - Viewer
5. Click **Share**.
6. Click **Close**.

4.5.6 How to Edit a Shared Project

When you open a shared project, by default it is opened in view mode. If you have permissions to edit a project and the project is not locked by another user, you can edit the project.

To begin editing a project:

To begin editing, click Edit at the top of the project welcome page

Once you have enable edit mode for a project, you can begin making changes to it. See Chapter 5, "Working with Processes and the Process Editor" for more information.

4.5.7 How to Save Changes to a Project

You can save changes to your project as you are editing processes and other project components. Changes are saved directly to the BPM repository.

To save changes to the current project:

1. If you want to continue editing, click **Save** in the process editor toolbar.

   All unsaved changes for each project component are saved. The project continues to be locked and you can continue editing

2. If you want to release the lock on the project, click **Save and Release** in the process editor toolbar.

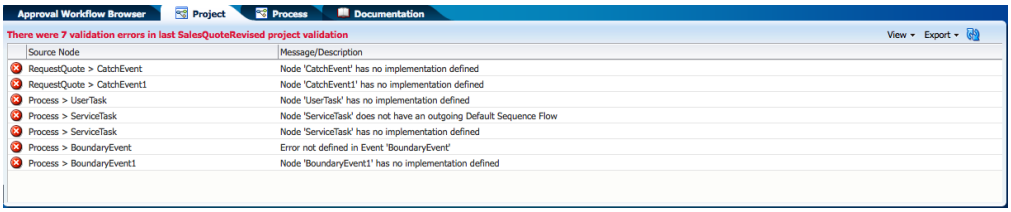
   All unsaved changes for each project component are saved. If project sharing is enabled, other Business Process Composer and Oracle BPM Studio users who have permissions begin editing the project You must switch to edit mode to make changes.

4.5.8 How to Validate a Project

Validating a project enables you to check your project and processes for any errors. Business Process Composer displays these errors in an error browser. Business Process Composer has an error browser for the project and one for each process.

Figure 4–5 shows the types of errors displayed within the project error browser.

Figure 4–5 The Project Error Browser



This graphic displays a tabbed pane called Project. Within the pane is a table that displays a toolbar with three items: the View menu, the Export menu, and Refresh tool.

The table contains two columns labeled: Source Node and Message/Description.

****

To validate a project:

1. Open your project.
2. Ensure that you are editing the project.
3. From the main menu, select **Validate Project**.

After validating your project, any errors found are displayed in the error browsers for the project or process.

> **Note:** You cannot deploy projects that contain errors. You must correct any errors before the project can be deployed.

4.5.9 How to Discard Changes to a Shared Project

While editing project elements, you can revert your changes and return to the most recent published version of a project.

To discard changes made to the current project:

1. From the main menu, select **Cancel**.
2. Click **OK** to confirm discarding changes to the current project.

> **Note:** After discarding your changes they cannot be recovered.

4.5.10 How to Close a Project

To close a project, select **Close Project** from the main menu.

4.5.11 How to View the History of Changes Made to a Project

You can view the history of major changes made to a project within the recent activity browser. This browser displays changes including the following:

- Creating the project
- Modifying resources
- Creating processes or human tasks

To view the history of a project:

1. Access the project welcome page.

   Project changes are displayed in the Recent Activity pane.

2. To view the details of a specific change, click the expand icon next to it.

4.5.12 How to View and Edit Project Properties

You can view and edit project properties in the project information pane of the project welcome page. The project information pane is shown in Figure 4–2.

You can edit the following properties of a project from this pane:

- **Description**: enables you to add an optional description of your project. This is useful when sharing your project with other users.
- **Approval workflow**: enables you to define the approval workflow for the project. See Section 7.4, "Configuring Approval Workflow for a Project" for more information.
- **Event generation:** enables you to configure how sampling points are generated for the project as a whole. Sampling points allow you to generate information about the performance of an flow object within in a running process. The data generated according to this configuration is stored in the Process Analytics Database.

  See Appendix A.1, "Common Properties" for information on setting sampling point generation for individual flow objects. See *Oracle Fusion Middleware Business Process Management User's Guide* for general information about sampling points.

You can view the following project properties from the project information pane:

- Edit mode: Displays the edit mode for the project. See Section 4.5.6, "How to Edit a Shared Project" for information on changing the edit mode of a project.
- Sharing: Displays the sharing configuration of the project.
- Project Language: Displays the project language.

4.5.13 How to Mark a Project as a Favorite

Marking a project as a favorite enables you to identify important or frequently used projects. In the project browser, you can choose to view only projects flagged as favorites.

To mark a project as a favorite, click the **Mark project as favorite** button in the upper right-hand corner.

4.6 Using Guided Business Processes to Create Project Milestones

The following sections describe how to use Guided Business processes and project milestones.

4.6.1 Introduction to Guided Business Processes

Guided Business Processes provide a guided visual representation of a process flow, improving the user experience by providing process participants with an encapsulated hierarchical view of the business process.

Guided Business Processes enable process designers to direct process participants to complete a business process through a set of guided steps associated with the process.

By following the steps outlined in a Guided Business Process, process participants require less training to complete a business process, and the results of the process are more predictable.

4.6.1.1 Introduction to Activity Guides and Milestones

A Guided Business Process is modeled as an activity guide that is based on a business process. The Activity Guide includes a set of Milestones. A milestone is a contained set of tasks that the process participant must complete. A milestone is complete when the user successfully runs a specific set of tasks in the milestone.

Each milestone is a specific set of human workflow tasks. Each human workflow task is itself a task flow that may require the collaboration of multiple participants in various roles. Depending on the nature of the task flows, a participant may save an unfinished task flow and go back to it at a later time.

4.6.2 How to Configure the Activity Guide and Create Project Milestones

Using Business Process Composer, you can configure Guided Business Processes and add milestones to them.

To configure the activity guide for a project:

1. Open your project.
2. From the Quickstart menu, select **Activity Guide**.

   The activity guide editor is displayed.
3. Enter a title for the activity guide.
4. Configure the following optional properties:
   - **Display Mode**: Determines how milestones and tasks within the guided business process display links. If the milestone and tasks use another configuration, then the guided business process configuration is ignored.

     Possible values are:
     - **Always**: Always displays the milestone and task links for all the milestones in this guided business process.
     - **When Instantiated**: Displays the milestone and task links only when one or more of the user tasks in the milestone are instantiated, for all the milestones in the guided business process.
   - **Task Access**: After the task is completed, the guided business process uses this configuration to display the links. If the task mode is active only, the tasks links are grayed out. If the task mode is any state, the tasks links remain enabled and a message appears when you run the task.

     Possible values are:
     - **Active Only**: The link to the task is enabled only when the task is active and the user can update it. When you complete the task the link to the task is grayed out.
     - **Any State**: The link to the task is always enabled after you instantiate the task, even after you complete the task.
   - **Root Process**: Determines the process used for this Activity Guide. You can only define one guided business process per BPM project. This process is the root process.

- **Description**: Provides an optional description for the Activity Guide.

5. Click **Save** in the project toolbar.

To create a new milestone:

1. Click **New Milestone**.
2. Select the milestone you just created from the list.
3. Configure the milestone as necessary.
4. Click **Save** in the project toolbar.

To add a user task to a milestone:

1. Open the process where you want to add a milestone.
2. Right-click the user task you want to add to a milestone.
3. Select a milestone from the list, then click **OK**.

4.7 Defining the Roles Used in a Project

This section describes how to create project roles.

4.7.1 Introduction to Project Roles

Project roles are used within your business processes to model the users or groups that are responsible for performing the work performed by you business process.

Roles enable you to define functional categories that correspond to job functions or responsibilities within your organization. Business Process Composer enables you to create and edit the required roles within your process and assign them to swimlanes. When an project is deployed to runtime, project roles are mapped to the real-world users and groups of your organization. More advanced mapping and configuration of project roles can be performed using BPM Studio or Oracle Business Process Management Workspace.

Project roles are defined for the entire project. This enables them to be shared by all the processes in your project. Within a process, roles are assigned to the horizontal swimlanes.

4.7.2 Working with Project Roles

The following procedures describe how to create and delete project roles.

How to create a project role:

1. Access the project welcome page.
2. Expand **Organization**, then click the **Add** icon.
3. Provide a name for the new role, then click **Add Role**.

How to delete a project role:

1. Access the project welcome page.
2. Expand **Organization**.
3. Select the role you want to delete, then click the **Delete** icon.

5

Working with Processes and the Process Editor

This chapter provides information about creating and using business processes in Oracle BPM. It provides a general introduction to business processes and describes the process editor window. It also provides procedural information for creating and using processes.

See Chapter 6, "Modeling Business Processes with Oracle BPM" for information on using Business Process Management Notation (BPMN) to design a business process.

This chapter includes the following sections:

5.1 Introduction to Business Processes

A business process can be defined as a sequence of tasks that result in a well-defined outcome. Business processes are the core components of process-based business applications created with the Oracle BPM Suite.

Although projects are higher level wrappers that contain all the resources of a business application, the processes within the project determine how the application works. This flow is defined by various BPMN flow objects.

Business processes are generally created by process analysts who determine the business requirements that must be addressed and define the corresponding process flow.

There are different types of business processes that determine how a process behaves in relation to other processes. These are described in Table 5–1.

Note: By default, new business processes are synchronous. After creating a new process you can change the type by editing the process.

Table 5–1 Types of Business Processes

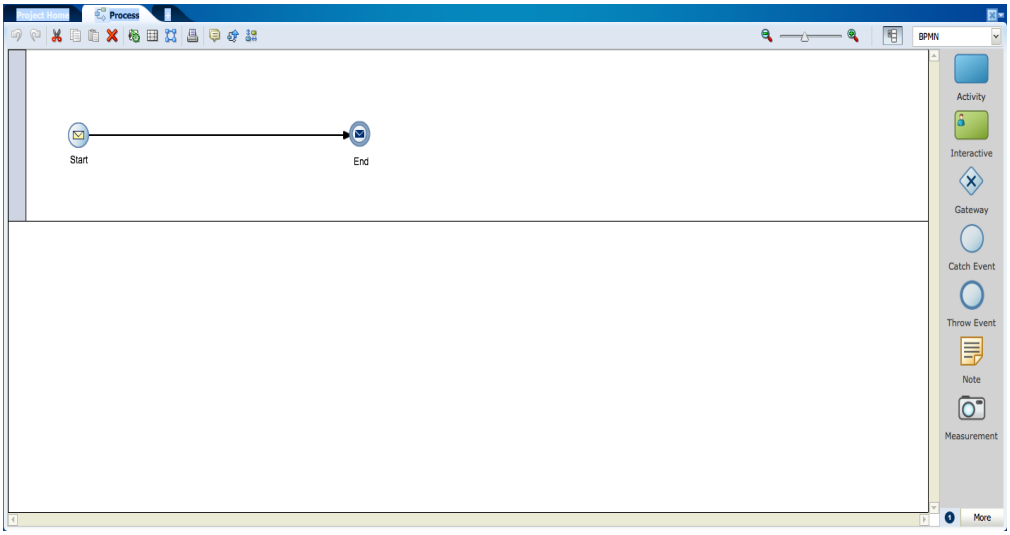
Process Type	Description
Synchronous Service	Synchronous services are processes that can be invoked from other processes or services synchronously. In a synchronous service, the calling process waits until the process completes before continuing.
Asynchronous Service	Asynchronous services are processes that can be invoked from other processes or services asynchronously. In an asynchronous service, the calling process does not wait until the process completes before continuing.
Manual Process	Manual processes are processes that require user interaction. Manual processes begin and end with none start and end events.
Reusable Process	Reusable processes are processes that can called by a BPMN process using the call activity.

5.2 Introduction to the Process Editor

The process editor is a graphical editor that enables you to create business processes by adding BPMN flow objects. You can drag BPMN flow objects from the component palette to the process editor.

Figure 5–1 shows an example of the process editor tab.

Figure 5–1 The Process Editor Window



This text describes the Process Editor Window.

The Process Editor is displays your business process and the component palette. It also contains a tool bar that is contains tools and features related to editing BPMN processes.

****

You can open multiple processes within the same project simultaneously in Business Process Composer. Each process opens in its own tab within the editor window.

The different areas of the process editor are described in the following sections.

5.2.1 Introduction to the Process Editor Toolbar

The process editor window contains a toolbar enabling access to the Business Process Composer features described in Table 5–2.

Table 5–2 Process Editor Menu

Menu Item	Description
Undo	Reverts the last change made to your process.
Redo	Reverses the last undo action you performed.
Cut	Cuts the selected items and copies them the clipboard.
Copy	Copies the selected items to the clipboard.
Paste	Pastes the items currently in the clipboard.
Delete	Deletes the selected elements from the process.
Autolayout	Automatically adjusts the layout of your process.
Toggle grid visibility	Shows or hides a grid in the process editor window.
Snap to grid	Causes flow objects to be centered on the nearest grid axis. Any existing flow objects will automatically be centered. When adding new flow objects, they will automatically be centered when they are added. This button is active only when toggle grid is enabled.
Print	Prints the process using your browser's printer setup.
Edit conversations	Opens the conversations editor. This editor enables process developers to define the interface used to determine the input and output data objects.
Find process usage	Determines what other processes within the current project call the current process.
View collaboration	Switches the process editor to collaboration view.
Zoom slider	Zooms in and out of your process.

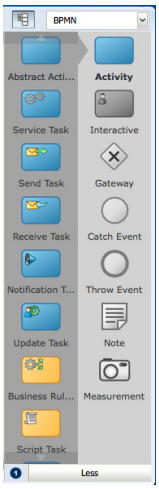
5.2.2 Introduction to the Process Editor Canvas

The process editor canvas is the central area of the process editor window. It enables you to create the graphical representation of your process using BPMN flow objects. In addition to a process flow, the process editor canvas also displays swimlanes.

5.2.3 Introduction to the BPMN Component Palette

The component palette appears within the editor window and enables you to add BPMN sequence flows and business catalog elements to your process. Figure 5–2 shows the component palette.

Figure 5–2 The Component Palette



This graphic displays the Component Palette.

****

The component palette enables you to drag and drop artifacts to the process editor window.

> **Note:** The component palette is greyed-out until you enter edit mode for the project.

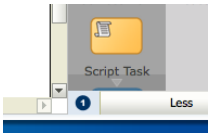
The component palette separates BPMN elements into the following groups:

- Activity
- Interactive
- Gateway
- Catch Event
- Throw Event
- Note
- Measurement

Business Process Composer provides two separate modes for adding flow objects to a process:

- Single object mode: enables you to add individual flow objects one at a time. This mode is indicated by a 1 within a blue circle as shown in Figure 5–3.

Figure 5–3 Single Object Entry Mode



This graphic displays the single object entry mode within the component palette.

****

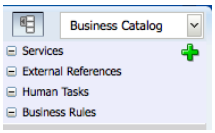
- Multiple object mode: enables you to quickly add multiple flow objects of the same type. This mode is indicated by an "N" within a blue circle.

You can toggle between the two modes by clicking the blue circle.

5.2.4 Introduction to the Business Catalog

This panel enables you to select reusable services from the business catalog. Figure 5–4 shows the business catalog.

Figure 5–4 The Business Catalog



This graphic displays the business catalog.

****

These services are grouped according to the following categories:

- Services
- External references
- Human tasks
- Business rules

> **Note:** External references and business rules cannot be created in Business Process Composer. They are created in Oracle BPM Studio and included as part of a project template or any project shared with Oracle BPM Studio.
>
> You can use Business Process Composer to create services based on Web Services. However, other services based on adapters and other SOA components must be created in Oracle BPM Studio and included within a project template or shared project.

You can use Business Process Composer to assign reusable services from the business catalog to the corresponding flow objects.

See Section 4.1.1.2, "The Business Catalog" for more information.

5.3 Working with Business Processes

The following sections describe how to create, open, and delete business processes.

5.3.1 How to Create a New Business Process

Business processes are created within an Oracle BPM project. You can add one or more processes to your project.

To create a new business process

1. Access the project welcome page.

2. If you are editing a shared project, ensure that you are currently editing the project.
3. Click **Processes**, then click **New Process**.
4. Enter a name for the process, then click **Create**.

The new process appears in the list of processes.

New business processes are created with a start and end event connected by a default sequence flow. By default, both the start and end event have the message trigger type.

See Section 6.2, "Defining the Start and End Point of a Process" for more information.

5.3.2 How to Open a Business Process

After opening an Oracle BPM project, you can open any of the processes it contains. Processes are opened in the process editor window.

To open a business process

1. Access the project welcome page.
2. Click **Processes**.
3. Click the name of the process you want to open.

The process opens in the process editor window. Before you can begin editing the process, you must ensure that you are in edit mode.

5.3.3 How to Delete a Business Process

You can delete processes from your project.

To delete a business process for a project:

1. Open your project.
2. Access the project welcome page.
3. Click **Processes**.
4. Move the cursor over the name of the process you want to delete.
5. Click the delete icon, the click **OK**.

5.3.3.1 What You Need to Know About Deleting a Business Process

When deleting a process you should ensure that there are no remaining references to the deleted process elsewhere in your project. For example, if the deleted process was invoked from another process through a message throw event, you must ensure that you have reconfigured the invoking process so it is no longer referring to the deleted process. An error should be displayed during validation if any remain references to the deleted process still exist.

5.4 Working with Flow Elements

This section describes the basic mechanics of using the process editor to add flow elements to a process. See Chapter 6, "Modeling Business Processes with Oracle BPM" for information on designing your business process using BPMN 2.0.

5.4.1 How to Add a Flow Object from the Component Palette

You can add flow objects to your process by dragging them from the component palette onto the process editor canvas.

To add a flow element from the component palette:

1. Open the process where you want to add flow elements.
2. Ensure you are in edit mode.
3. In the component palette, double-click the type of flow object you want to add.
4. Select the object entry mode you wish to use.

   You can choose to enter a single flow object or multiple flow objects of the same type. See Section 5.2.3, "Introduction to the BPMN Component Palette" for more information.
5. Click and drag the flow object you want to add to the area in process editor canvas where you want to place it.

   The cursor displays the icon associated with the type of flow object.
6. Position the cursor to the point in your process where you want to add the flow object, then click the mouse.

   If you are in multiple object mode, you can continue clicking within the process editor canvas to add additional flow objects of the same type.

> **Note:** If you position the cursor over a sequence flow, Business Process Composer will automatically create incoming and outgoing sequence flows for the new flow object.

5.4.2 How to Cut, Copy or Delete a Flow Object

Within the process editor window, you can cut copy or delete flow objects.

To cut, copy, or delete a flow object:

1. Select the flow object or sequence flow that you want to cut, copy, or delete.
2. Select **Cut**, **Copy**, or **Delete** from the process editor toolbar.

> **Note:** When you cut or delete a flow object that contains incoming and outgoing sequence flow, Business Process Composer automatically connects it to the outgoing sequence flow. However, you may need to manually reconfigure the surrounding flow objects.

5.4.3 How to Paste a Flow Object in a Process

You can paste that you previously cut or copied.

To paste a flow object in a process:

1. Right-click in the area of the process editor canvas where you want to paste a flow object.
2. Select **Paste**.

5.4.4 How to Add a Sequence Flow to a Process

Sequence flows define the order or sequence that the work is performed within a process. For more information see Section 6.6, "Controlling Process Flow Using Sequence Flows"

To add sequence flows to your process:

1. Open your process.
2. Move the cursor over the flow object where you want to create the outgoing sequence flow.
3. Click the **Add Sequence flow** button.

   This button only appears for flow objects that do not have outgoing sequence flows.
4. Move the cursor to the flow object you want to connect to, then left-click.

5.4.5 How to Delete a Sequence Flow

You can delete a sequence flow from a BPMN process.

To delete a sequence flow from a process:

1. In the process editor canvas, right-click the sequence flow you want to delete.
2. Select **Delete**.

5.4.5.1 What You Need to Know About Deleting a Sequence Flow

When you delete a sequence flow from a process, any implementation details you may have configured for the sequence flow are lost.

5.4.6 How to Edit the Properties of a Flow Object

You can edit the basic properties of a sequence flow.

To edit the properties of a flow object:

1. Right-click on the flow object whose properties you want to edit.
2. Select **Properties**.
3. Edit the properties of the flow object.
4. When you are finished editing the properties, click outside the properties dialog window.

   Your changes are automatically saved and the dialog window closes.

5.4.7 How to Assign a Custom Icon to a Flow Object

Business Process Composer enables you to select a custom icon to replace the default BPMN icon of a flow object. You can select from a list of custom icons provided by Oracle BPM.

To assign a custom icon to a flow object:

1. Right-click the flow object, the select **Properties**.
2. Click **Change**, then select the icon you want to use.

3. Click outside the properties window to apply your changes.

5.5 Working with Business Catalog Components

The following sections provide information on working with the business catalog in Business Process Composer. See Section 4.1.1.2, "The Business Catalog" for more information on the business catalog.

5.5.1 How to Assign a Business Catalog Component to a Flow Object

Business Process Composer enables you configure implementation details for a flow object by assigning business catalog components to it.

The flow objects that you can assign business catalog components to are:

- Business rule task
- Service task
- User task
- Message events and the receive task

To assign a business catalog component to a flow object:

1. Open your process.
2. Right-click on the flow object where you want to add the business catalog component.
3. Select **Implementation**.
4. Select Browse, then select the business catalog component from the list.
5. Click **OK**.
6. Click **Apply Changes**.

Note: You must click **Apply Changes** to save the any changes you make to the implementation of a flow object. Even if you save the project, implementation changes are not saved until you click **Apply Changes**.

5.5.2 How to Create New Human Tasks in the Business Catalog

You can create new human tasks within the business catalog. See Chapter 11, "Working with Human Tasks" for more information.

After creating a human task, you can then assign them to the user tasks within your process. See Section 6.3, "Adding User Interaction to Your Process" for information on using human tasks within a BPMN process.

5.6 Working with Draft Processes

Oracle BPM enables you to create and deploy draft processes.

5.6.1 Introduction to Draft Processes

A draft process is a process that has one or more flow objects which do not have their implementation defined. Deploying a draft process enables you to test the parts of

your process that have been completed without having to wait until all flow objects have been implemented. Draft processes are created by marking one or more flow objects within the process as draft.

When you configure a flow object to be a draft, you cannot configure data associations for the flow object. If mark a flow object as draft that you have previously assigned data associations for, the data associations will be lost.

You can define the implementation details of a draft flow object. However, it is not required. Draft flow object with no implementation defined will not generate errors when the project is validated.

When a project containing a draft flow object is deployed, any implementation details that are defined are ignored. For example if your process contains a user task marked as draft, it will not create instances of the associated human task at runtime.

5.6.2 How to Mark a Flow Object as Draft

You can mark a flow objects as draft by editing the flow objects implementation properties.

To mark a flow object as draft:

1. Open your process.
2. Right-click on the flow object, then select **Implementation**.
3. Select the checkbox next to **Is Draft**.
4. Click **Apply Changes**.

5.7 Documenting Your Process

Oracle BPM enables you to create documentation for your process using the documentation editor. You can add documentation for an entire process or for individual flow objects within a process.

Oracle BPM enables you to create two separate types of documentation:

- **End User**: Documentation that is available to end users of your process using the Process Workspace application.
- **Internal (Use Case)**: Documentation used to make your process more understandable to other process analysts and developers who may revise you process later.

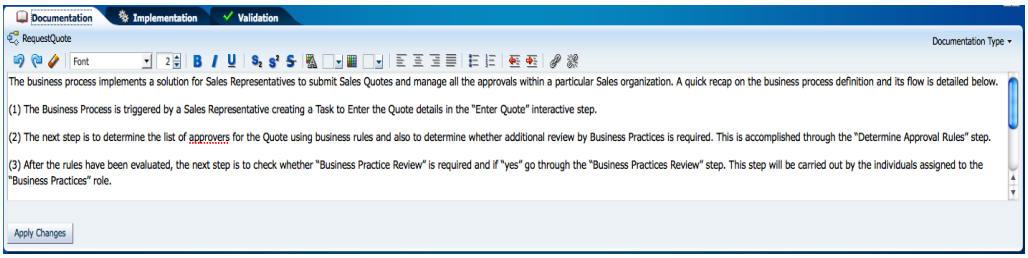
  You can define use-case documentation for each of the activities, events, and gateways within your process.

  Note: You cannot create documentation for sequence flows or measurement marks.

5.7.1 Introduction to the Documentation Editor

The documentation editor contains a toolbar and editor pane that enables you to enter the documentation for your process.

Figure 5–5 The Documentation Editor



This text describes the Documentation Editor

****

5.7.1.1 Inserting Links in Your Documentation

When adding links in your documentation, you should include a full URL as part of the link. For example, you should refer to external links as http://www.oracle.com, instead of just www.oracle.com.

5.7.2 How to Add Documentation to Your Process

Business Process Composer enables you to add documentation to your processes and the flow objects within your processes.

To add documentation to a process:

1. Open your process.
2. Click **Restore Pane** to display the documentation editor. The Restore Pane icon is located in the lower right corner of the Business Process Composer application.
3. Click the **Documentation** tab, then select the tab for the process.
4. Select the type of documentation you want to create from the **Documentation Type** drop-down menu.
5. Enter your documentation in the editor window.
6. Click **Apply** to save your changes.

> **Note:** You should apply your changes before selecting another process or process element. If you navigate away from the documentation editor before applying your changes, they are not saved.

To add documentation to a specific flow object within a process:

1. Open the process where you want to add documentation.
2. Right-click the flow object where you want to add documentation, then select **Properties**.
3. In the Properties dialog box, click **Documentation**.
4. Select the type of documentation you want to create from the **Documentation Type** drop-down menu.
5. Enter your documentation in the editor window.
6. Click **Apply** to save your changes.

Note: You should apply your changes before selecting another process or process element. If you navigate away from the documentation editor before applying your changes, they are not saved.

5.7.3 How to Add Notes to a Process

You can add notes to your process to make them easier to understand.

To add a note to a process:

1. Open your process.
2. Ensure that you are in edit mode.
3. In the BPMN component palette, click and drag the Note icon to the point in your process where you want to add the note.
4. Double-click on the note to edit the text.
5. Enter the text of the note, then click outside of the note to finish.

5.8 Importing and Exporting Process Models

Using Business Process Composer you can import and export process models created in other programs.

5.8.1 Importing Process Models into Oracle BPM

Business Process Composer enables you to import process models and convert them to BPMN notation. You can import process models in the following formats:

- Visio
- Workflow
- XPDL
- Oracle Tutor (files are saved using the .docx extension)

When converting Visio or XPDL processes, you may need to modify the processes before conversion to ensure that they are converted accurately. See Appendix B, "Preparing Processes for Import into BPMN" for more information.

Note: If the original file contains properties and artifacts that are not supported by BPMN, the unsupported elements are not converted and are omitted from the final BPMN process.

For example, if the origin file contains loop characteristics on a regular activity, which is not supported in BPMN, the BPMN process will not contain the loop characteristics after conversion.

To import a process model:

1. From the main menu, select **Import**, then **Import Model**.
2. On your local file system, browse to the file you want to import, then click **OK**.
3. If you are importing a Visio or XPDL file, select one of the following:
   - Create a separate model from each pool

- Merge pools into a single model

This dialog appears even if the original file does not contain multiple pools.

4. Click **OK**.

You can view the newly created BPMN processes from the project welcome page.

5.8.2 Exporting BPMN Processes to Oracle Tutor

Using Business Process Composer you can export BPMN processes to Oracle Tutor. These files are exported as Microsoft Word (.docx) files and contain Oracle Tutor formatting.

To export a BPMN process to Oracle Tutor:

1. From the main menu, select **Export**, then **Export to Word**.
2. Select one of the following:
   - Active process:
   - All open processes:
   - All processes of project:

   The converted processes are downloaded as a .zip file to your local filesystem.
3. Click **Save**.

To view the exported processes, you must extract the .zip file. Each individual process file contains conversion notes for any objects that were altered during conversion.

6

Modeling Business Processes with Oracle BPM

This chapter describes how to use create and model business processes using Business Process Management Notation and Modeling (BPMN) within the Oracle Business Process Management Suite.

This chapter provides specific information on about Oracle's implementation of BPMN 2.0. See Chapter 2, "Overview of Business Process Design" for a general introduction to BPMN using the Sales Quote example project. For general information about BPMN, including the formal specification, see http://www.bpmn.org.

This chapter is organized by the different types of tasks your business process must perform. It includes the following sections:

6.1 Using Swimlanes to Organize Your Process

This section shows you how to organize your process using swimlanes. It also describes how to use roles to determine which members of your business organization are responsible for performing the work of your process-based application.

6.1.1 Introduction to Roles

A key to designing a business process is determining the people and roles required to complete each of the tasks that require user interaction. Within your process, roles are used to model who is responsible for performing the work performed within your

business processes. Roles enable you to define functional categories that represent job functions or responsibilities within your organization.

The roles defined in your process are also referred to as logical roles. When your Oracle BPM project is deployed to the run time environment, these roles are mapped to LDAP roles that correspond to the users in your real-world organization.

Roles are assigned to the horizontal swimlanes that display the roles responsible for completing activities and tasks within your process. Business Process Composer enables you to create and edit the required roles within your process and assign them to swimlanes.

Using Oracle BPM Studio, you can also map roles to specific users using LDAP. Oracle BPM Studio also enables you to create more robust organizational models using organizational units, calendars, and holidays. See the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for more information.

6.1.1.1 Roles in Context

Process analysts are responsible for determining what roles are required when designing a business process.

The Sales Quote example defines the following roles:

- Sales Rep: Sales representatives are responsible for creating and updating a sales quote until is approved by the other roles defined in the project.
- Approvers: Approvers represent users who are responsible for approving the combination of products and pricing structure defined by the sales quote.
- Business Practices: This role represents users who are responsible for viewing and approving the sales quote. Additionally, they have the authority to add additional approvers during the review of the sales quote.
- Contracts: This role represent users who are responsible for approval of the terms specified in the sales quote and also for creating the formal legal documents that can be forwarded to the customer.

See Section 2.2, "Introduction to the Sales Quote Example" for more information on the Sales Quote example project.

6.1.2 Introduction to Swimlanes

Swimlanes are the horizontal lines that run across the process editor. All flow objects must be placed within a swimlane.

Swimlanes can also be used to group flow objects based on the roles defined within your process. Swimlanes that contain user tasks must have roles assigned to them. Swimlanes visually display the role responsible for performing each flow object within your process. Additionally, you can have multiple swimlanes that are assigned to the same role.

Swimlanes can make your process more readable when you must use the same role in different parts of the same process.

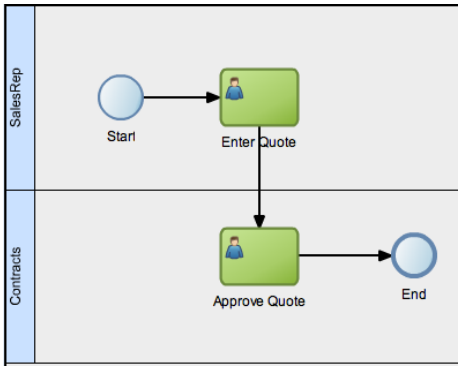
When you create a new process, Oracle BPM Studio and Business Process Composer create a default swimlane. You can add additional swimlanes to your process as necessary. When adding interactive and manual activities to a process, you must assign a role to the swimlane.

Note: You cannot delete a swimlane that contains the only start or end even of a process.

6.1.2.1 Swimlanes in Context

Figure 6–1 shows a simple process split across multiple swimlanes. In this example, the SalesRep role is assigned to the first swimlane. Because the Enter Quote user task appears inside this swimlane, process participants assigned to the SalesRep role are responsible for performing this task.

Figure 6–1 A Simple Business Process Split Across Two Swimlanes



This graphic shows a simple process with two swimlanes: SalesRep and Contracts.

The SalesRep swimlane includes a start event with a sequence flow to a user task labeled Enter Quote.

From the Enter Quote task, a sequence flow extends to the Contracts swimlane to a user tasks labeled Approve Quote.

From the Approve Quote task, a sequence flow extends to an end event.

****

In a real-world business process, the combination of swimlanes and flow objects within them can be complex. The Sales Quote example project, as shown in Figure 2–1, is an example of a more complex process using multiple swimlanes.

6.1.3 How to Add Roles and Swimlanes to Your Process

You can add roles and swimlanes to your BPMN process.

To add a new swimlane to your process:

1. Right-click on a white area of the process editor canvas.
2. Select **Add Lane**.

The new swimlane is created. By default, the swimlane is not assigned a role. You can add a role by editing the swimlane properties.

To add a new swimlane and role to your process by adding a new flow object:

1. Open the process where you want to add a swimlane.
2. From the component palette, select a flow object, then drag and drop it on the process canvas below an existing swimlane.

The new swimlane is created. By default, the swimlane is not assigned a role. You can add a role by editing the swimlane properties.

6.1.4 How to Edit Swimlane Properties

You can edit the properties of a swimlane in the process editor.

To edit swimlane properties:

1. Move the cursor over the role name for the swimlane.
2. Click the **Edit** icon.
3. Determine if you want to use an existing role, or create a new one:

   To assign an existing role to a swimlane:

   1. Click the Use button.
   2. Select a role from the drop-down list.

   To assigning a new role to a swimlane:

   1. Click the **Create** button.
   2. .Enter the name of the new role in the text field.
4. If you want to optionally add a custom icon to a swimlane, click **Change**, then select the icon you want to use.
5. If you want to optionally change the background color of a swimlane:
   1. Click **Implementation**.
   2. In the implementation properties editor, enter the RGB value of the color or select a color from the color palette.
   3. Click **Apply Changes**.

6.1.5 Sharing Roles Between Business Process Composer and BPM Studio

Oracle BPM Studio enables you to integrate roles within complex organization models based on organizational units, calendars and holidays.

When editing a project or creating a project based on a project template in Business Process Composer, you can access the roles defined within the project. However, you cannot view or edit the organizational information defined within the project.

Additionally, you can create new roles using Business Process Composer. These roles are incorporated as part of the overall organization information of the project.

6.2 Defining the Start and End Point of a Process

This section describes the BPMN flow objects used to define the start and end of a process.

6.2.1 Introduction to Start and End Events

Start events are BPMN flow objects that define the starting point of a process. There are different types of start events that determine how process instances are created.

End events, in contrast, define the end point of a process. There are different types of end events that determine what happens when the process instance is completed.

Note: In Oracle BPM, all BPMN processes must have at least one start and one end event.

Because start events define the beginning of a process, they do not have incoming sequence flows. Likewise, end events cannot have outgoing sequence flows.

However, except for the none end and start events, start and end events can have input and output to processes.

6.2.1.1 Specifying the Start Events for Different Types of Processes

When you create a new process Business Process Composer creates a message start and message end event.

You can change these defaults depending on the type of business process you need to create. Table 6–1 shows the default start and end events for each type of process.

Table 6–1 Start and End Events for Each Process Type

Process Type	Default Start and End Event Types
Asynchronous service	Message start and end event
Synchronous service	Message start and end event
Manual process	None start and end event

See Section 5.1, "Introduction to Business Processes" for more information on the different types of processes supported by Oracle BPM.

Subprocesses contain none start and end events by default. These are the required start and end events and cannot be changed.

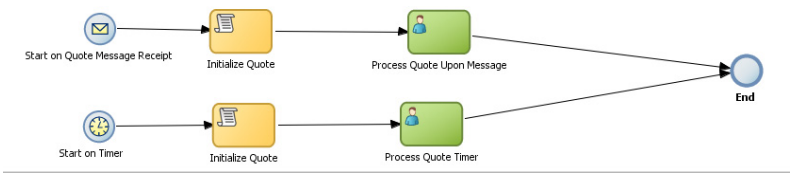
Event subprocesses contain a message start and none end event by default. However, you can change the start event to reflect the type of event you are handling.

6.2.1.2 Using Multiple Start Events in a Process

You can define multiple start points in a BPMN process. Multiple start points enable you to specify multiple ways of creating a process instance, depending on which start event is used.

Figure 6–2 shows an example process that contains both a message start and timer start event.

Figure 6–2 Using Multiple Start Events within a Process



This graphic shows a process with two start events.

The first is a message start event labeled Start on Quote Message Receipt.

From this message start event, a sequence flow extends to a script task labeled Initialize Quote.

From the Initialize Quote task a sequence flow extends to a user task labeled Process Quote Upon Message.

From the Process Quote Upon Message task, a sequence flow extends to an end event.

The second message is a timer start event labeled Start on Timer. From this event a sequence flow extends to a script task labeled Initialize Quote.

From the Initialize Quote task a sequence flow extends to a user task labeled Process Quote Timer.

From the Process Quote Timer task, a sequence flow extends to the same end event mentioned earlier.

****

This process can be started using a message event when called from another process or service. It can also be started based on a time interval if the process instance must be created automatically.

Using multiple start events enables you to have multiple ways of starting a process without having to create two separate processes.

6.2.1.3 Using Multiple End Events in a Process

End events mark the end of a process path. When you have only one end event in your process and the token reaches the end event, the process is stopped when the end event is reached.

> **Note:** Message end events can only be used to terminate processes initiated by a message start event. Additionally, if you have multiple message end events associated with a message start event, each of these message end events must have the same quantity and type of output arguments.

When you are using multiple end events, it is possible for different tokens to take different paths within a process. In typical cases, all parallel paths must reach an end event before the process is completed.

However, in the following special cases, a process instance can be stopped before all process paths have completed:

- Error end event: When an error end event is reached, all process activity is stopped. Like the error throw event, the error end event stops the flow of a process. See Section 6.2.9, "Introduction to the Error End Event" for more information.
- Terminate end event: The terminate end event causes all work on a process to stop immediately. There is no error handling or other clean up of the running process. See Section 6.2.11, "Introduction to the Terminate End Event" for more information.

6.2.2 Defining How a Process Instance is Triggered

Oracle BPM supports the following ways of triggering a process instance:

- Using a message, signal, or timer start event. See the following sections for more information:
  - Section 6.2.4, "Introduction to the Message Start Event"
  - Section 6.2.5, "Introduction to the Signal Start Event"

  - Section 6.2.6, "Introduction to the Timer Start Event"

- Using a none start event followed by a receive task. The receive task must be configured to create a process instance. See Section 6.4.5, "Introduction to the Receive Task" for more information.
- Using a none start event followed by a user task defined with the initiator pattern. See Section 6.3.2, "Introduction to the User Task" for more information.
- Using an event-based gateway that is configured to create a new process instance. See Section 4.6, "Using Guided Business Processes to Create Project Milestones" for more information.

6.2.3 Introduction to the None Start Event

The none start event is used when no instance trigger is specifically defined. Process analysts can use the none start event as a placeholder when the necessary start event of a process is unknown or is defined and implemented later by process developers.

Figure 6–3 shows the default notation for the none start event

Figure 6–3 The None Start Event

The none start event is represented by single circle.

****

None start events are also used to specify the beginning of a process where the process instance is created by another flow object. In general, the none start event does not trigger a new process instance.

However, when used with the following, the none start even does trigger a new process instance:

- Receive task: The receive task must have the Create Instance property set to true.
- User task: The user task implemented with the initiator pattern

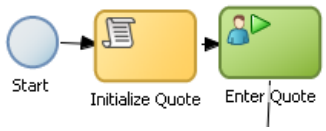
Similar to other start events, the none start event cannot have incoming sequence flows. It can only have default out-going sequence flows.

> **Note:** None events are always used to define the beginning of subprocesses.

6.2.3.1 The None Start Event in Context

Figure 6–4 shows an example of the none start event within the Sales Quote example project. In this example, the none start event defines the start of the process. Additionally, since the process contains a user task implemented with the initiator pattern, the none start event triggers a process instance.

Figure 6–4 The None Start Event within the Sales Quote Example Process



This figure shows an example of the none start event. It shows three separate flow objects: a none start event, a script task, and a user task implemented with the initiator pattern.

****

6.2.3.2 Data Associations

The none start event does not accept process input arguments.

6.2.4 Introduction to the Message Start Event

The message start event triggers a process instance when a message is received. This message can be sent from another BPMN or BPEL process or from a service.

Messages are types of data used to of exchange information between processes. Just as data objects are used to define the data used within a project, messages are used to define the data used between processes or between a process and a service.

Figure 6–5 shows the default notation of the message start event.

Figure 6–5 The Message Start Event

The message start event is represented by a single circle with a yellow envelope icon in the middle.

****

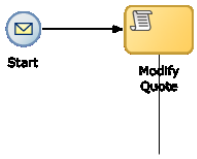
Similar to other start events, the message start event cannot have incoming sequence flows. Message start events require a default outgoing sequence flow.

You can expose a BPMN process as a service that enables other processes and applications to invoke the process. To expose a process as a service, your process must begin with a message start event. Additionally, you must define the input arguments to the process which are the data objects passed to the message start event. See Section 10.5.1, "How to Define the Input Arguments for a Process" for more information.

6.2.4.1 The Message Start Event in Context

Figure 6–6 shows a modified version of the Sales Quote process. Here, the process begins with a message start event that initiates the process instance.

Figure 6–6 The Message Start Event Within the Sales Quote Example Process



This figure shows an example of a message start event. It shows two separate flow objects: a message start event which initiates the process instance script task which is used to initialize the values of data objects passed to the process.

****

6.2.4.2 Using Process Input and Output Arguments

The message start event enables you to specify input and output arguments to a process. These arguments define the message that other processes or services must send to the process during invocation. See Section 10.5, "Defining Process Input and Output" for information on how to configure process input and output arguments.

6.2.5 Introduction to the Signal Start Event

The signal start event is similar to a message start event in that it is based on communication from another process or service. However, the message start event responds to a message sent to a specific process. In contrast, the signal start event is a response to a signal broadcast to multiple processes.

Signals can be broadcast from a BPMN process using the signal throw event. Using a combination of signal throw events and signal start events, you can invoke multiple processes simultaneously.

The signal start and throw events are added to a process by process developers. For information on implementing the signal throw event, see "Introduction to Communicating Between Processes Using Signal Events" in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

Figure 6–7 shows the default notation for the signal start event.

Figure 6–7 The Signal Start Event

The signal start event is represented by a single circle with a triangle in the middle.

****

6.2.5.1 The Signal Start Event in Context

The signal start and throw events are added to a process and implemented by process developers.

6.2.6 Introduction to the Timer Start Event

The timer start event triggers the creation of a process instance based on a specific time condition. You can configure the timer start event to trigger a process instance based on the following:

- A specific date and time. For example, a process could be triggered on December 31, at 11:59 P.M.
- A recurring interval. For example, a process could be triggered every 10 hours, 5 minutes, 32 seconds.

Figure 6–8 shows the default notation for the timer start event.

Figure 6–8 The Timer Start Event

The timer start event is represented by two concentric circles with a clock in the middle.

****

6.2.7 Introduction to the Error Start Event

The error start event is used as the start event of an inline handler. Using inline handlers you can define a separate process flow to handle errors that occur within your process.

Figure 6–10 shows the default notation for the none end event.

Figure 6–9 The Error Start Event

The error start event is represented by a circle with a lightning bolt in the center.

****

Note: Note: Error start events can only be used within inline handlers. They cannot be used within normal process flows.

You can define multiple inline handlers to handle error conditions. However, you cannot define multiple inline handlers that use the same exception.

6.2.8 Introduction to the None End Event

The none end event is used to mark the end of a process path. When a token reaches a none end event, it is consumed. If there are no other tokens within the process instance, the instance is complete.

The none event is used when your process is not required to perform any action after it completes. It can also be used as a placeholder by process analysts, to be changed later during implementation by a process developer.

Figure 6–10 shows the default notation for the none end event.

Figure 6–10 The None End Event

The none end event is represented by single circle.

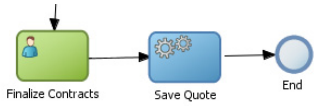
****

The none end event is always used to mark the end of a subprocess and event subprocess.

6.2.8.1 The None End Event in Context

Figure 6–11 shows an example of the none end event within the Sales Quote example. In this example, the Sales Quote service task is used to save information about the sales quote to a database.

Figure 6–11 The None End Event Within the Sales Quote Example



This figure shows an example of the none end event. It shows three separate flow objects: a user task, a service task, and the none end event.

****

Because no other work can be performed when the token reaches the end of a process, a none end event is used. After all process tokens reach the none end event, the process instance completes.

6.2.9 Introduction to the Error End Event

The end error event is used when the end of a process is the result of an error condition.

Errors end events are typically used with the error boundary event. The error boundary event is used to change the process flow based on a specific error. This flow usually ends using an error end event. See Section 6.8.3, "Introduction to the Error Catch Event" for more information on using the error intermediate event.

Figure 6–12 shows the default notation for the error end event.

Figure 6–12 The Error End Event

The error end event is represented by two concentric circles with a lightning bolt in the middle.

****

For information implementing the error end event, see the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

6.2.10 Introduction to the Message End Event

The message end event is used to send a message to another process or service when the process is completed. The message end event is always used with either a message start event or message catch event.

> **Note:** When creating a process that has multiple end events, you must ensure that any tokens that reach a message end event were created by a message start event. For example, you cannot use a message end event to end a process instance initiated by a timer start.

Figure 6–13 shows the default notation for the message end event.

Figure 6–13 The Message End Event

The message end event is represented by single circle with an envelope in the middle.

****

For information on how to configure process output arguments using message end events, see Section 10.5.3, "How to Define the Output Arguments for a Process".

6.2.11 Introduction to the Terminate End Event

The terminate end event is used to immediately stop a process. When a terminate end event is reached, the process stops immediately. There is no error handling or additional cleanup performed.

6.3 Adding User Interaction to Your Process

Many business applications require interaction from process participants within your organization. This interaction can be as simple as entering information into a form or can involve multiple work flows and multiple users.

This section describes the BPMN flow objects that are used to model how process participants interact with your business processes. It contains the following sections:

- Section 6.3.1, "Introduction to Human Workflow"
- Section 6.3.2, "Introduction to the User Task"
- Section 6.3.3, "Introduction to the Manual Task"

6.3.1 Introduction to Human Workflow

Many end-to-end business processes require manual interaction with the process. For example, users may be needed for approvals, exception management, or performing activities required to advance the business process.

Oracle Human Workflow provides comprehensive support for user participation by providing the following features:

- Manual interactions with processes, including assignment and routing of tasks to the correct users or groups

- Deadlines, escalations, notifications, and other features required for ensuring the timely performance of a task.
- Organization, filtering, prioritization, and other features required for process participants to productively perform their tasks.
- Reports, reassignments, load balancing, and other features required by supervisors and business owners to manage the performance of tasks.

For more information, see "Getting Started with Human Workflow" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

6.3.1.1 Introduction to Human Tasks

Human tasks are a component of Oracle Human Workflow. Human tasks enable you to interleave manual interactions with connectivity to systems and services within an end-to-end process flow. Human tasks are responsible for handling all interactions with users or groups participating in the business process. They do this by creating and tracking tasks for the appropriate users in the organization. Users typically access tasks through a variety of clients, including the worklist application, e-mail, portals, or custom applications.

Human tasks enable process developers to define how process participants interact with process-based applications created using Oracle BPM Suite and Oracle SOA Suite.

Using human tasks, process developers can define the interface and workflow for end user interaction by creating the following:

- Roles and assignments
- Deadlines and escalations
- Presentations

Human tasks are reusable services that can be used within other processes that require the same UI. Human tasks are created using Oracle BPM Studio.

6.3.2 Introduction to the User Task

The user task represents a part of your process where a process participant is required to perform work. This can be a simple interaction, such as entering a form, or part of a more complicated workflow that requires input from multiple process participants.

Figure 6–14 shows the default notation for the user task.

Figure 6–14 The User Task

The user task is represented by green rectangle with an icon in the middle. This figure shows a human icon representing the generic user task.

****

In the Oracle BPM Suite, process participants interact with your business application using the Process Workspace. The specific user interface elements, including the screens and panels that process participants see, are created using human tasks.

When designing a process, process analysts often add the user task to a process diagram. Process developers then create the necessary human tasks and implement them as part of creating the overall process-based business application.

When a token reaches a user task, the corresponding human task is performed. The token waits until the human task is completed before continuing to the next flow object.

See Section 5.5.1, "How to Assign a Business Catalog Component to a Flow Object" for procedures on how to assign elements from the business catalog to a user task.

For information on how implementing user tasks, see "Using Human Tasks" in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

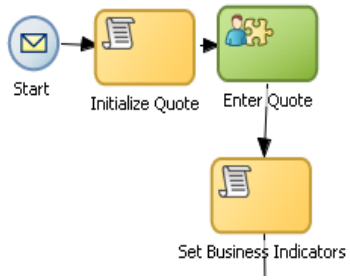
Similar to other flow objects, the user task may contain incoming and outgoing data associations.

User tasks may also contain incoming and default outgoing sequence flows.

6.3.2.1 The User Task in Context

In the Sales Quote example, the Enter Quote task, shown in Figure 6–15, represents entering information about the quote.

Figure 6–15 The Enter Quote User Task



This graphic shows a start event with a sequence flow extending to a script task labeled Initialize Quote.

From the initialize quote task, a sequence flow extends to a user task labeled Enter Quote.

From the Enter Quote task, a sequence flow extends to a task labeled script Set Business Indicators.

****

After the user enters information about the sales quote, the process flow passes through the outgoing sequence flow to the next flow object in the process.

6.3.2.2 Using Interactive Activities

Oracle BPM Studio enables you to add interactive activities to a process directly from the component palette. Interactive activities are short cuts based on the task routing and approval features of Oracle Human Workflow. See "Getting Started with Human Workflow" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite* for more information.

Table 6–2 shows the interactive activities that are available in Business Process Composer component palette.

Table 6–2 Interactive Activities

Pattern	Description
Complex	Uses a complex routing flow that is defined within the human task.
Management	Uses the management chain pattern where the assignee is set to the management chain pattern for the process participant belonging to the group or role assigned to the swimlane.
FYI	Bases assignment on the participant, role, or group defined in the swimlane. Similar to the user interactive activity, but the FYI activity does not wait until completion before continuing.
Group	Uses the group vote pattern. The assignee for the task is automatically set to the role or group associated with the lane. This interactive activity can only be added to swimlanes that are assigned to roles or groups.
Initiator	The initiator pattern is used to create a process instance.

6.3.2.3 Using the User Task in Project Templates

Using Oracle BPM Studio, you can add human tasks to the business catalog. Process analysts can use these in Business Process Composer when working with projects created from project templates.

When adding the user task to a project template to be used within Business Process Composer, follow these guidelines:

- Process developers must create any required human task services within Oracle BPM Studio before using the template in Business Process Composer. Human tasks can be implemented within a user task or this implementation can be performed when editing the project based on the project template.
- However, if the human task service is being implemented for a user task that is sealed within a project template, you must also perform the implementation before using the project template in Business Process Composer.

6.3.3 Introduction to the Manual Task

The manual task represents a task performed by process participants that is outside the scope of Oracle BPM. Manual tasks are used as placeholders within your process to show work that is not managed by the BPMN service engine at run time. Additionally, manual tasks do not appear in the Process Workspace application.

> **Note:** Manual tasks are not managed by Oracle BPM. The Oracle BPM run time does not track the start and completion of the manual task.

Figure 6–16 shows the default notation for the manual task.

Figure 6–16 The Manual Task

The manual task is represented by green rectangle with a single hand in the middle.

****

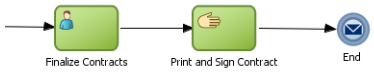
Manual tasks can only have one default incoming and one default outgoing sequence flow.

Unlike most BPMN flow objects, the manual task does not enable you to manipulate data objects. Data objects associated with the previous flow element are passed through as is to the next flow element.

6.3.3.1 The Manual Task in Context

In the context of the Sales Quote example, a manual task can be added for printing and signing a copy of a formal contract as shown in Figure 6–17.

Figure 6–17 Example of a Manual Task



In this graphic, a sequence flow extends to a user task labeled Finalize Contracts.

From the Finalize Contracts task, a sequence flow extends to a manual task labeled Print and Sign Contract.

From the Print and Sign Contract task a sequence flow extends to a message end event.

****

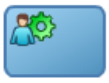
In this example, signing the formal contract is something that you may want to explicitly show as part of your business process. However, because it is not managed by the BPMN Service Engine, a manual task is used.

6.3.4 Introduction to the Update Task

The update task is used to perform operation on one or more Human Tasks. For example, you can use the update task perform actions like reassigning a task to another user.

Figure 6–18 shows the default notation for the update task.

Figure 6–18 The Update Task



The update task is represented by a blue rectangle containing the icon for a person and a green gear.

****

For more information on using the update task see *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

6.4 Communicating With Other Processes and Services

Oracle BPM enables you to define interactions across business processes within a process-oriented application. The following sections describe the BPMN flow objects used to model communication between processes.

This section describes how to use these flow objects to create process models using Business Process Composer. For information on how to implement these flow objects within a process-based application, see the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

6.4.1 Introduction to the Service Task

The service task enables you to communicate with other processes and services. Process analysts can add the service task when they know that a process must invoke an external service or process.

Process developers can then implement the necessary services. You can use the service task to invoke the following:

- Other BPMN processes
- BPEL processes
- SOA service adapters
- Mediators that are exposed as services

See Section 5.5.1, "How to Assign a Business Catalog Component to a Flow Object" for information on for procedures on how to assign elements from the business catalog to a service task.

The service task has similar behavior to the send and receive task pair and the message throw and catch event pair. The primary difference is that the service task is used to invoke processes and services synchronously. When the service task invokes a process or service, the token waits at the service task until a response is returned. After the response is received, the token continues to the next sequence flow in the process.

See "Using Service Tasks to Invoke Synchronous Operations in Services and BPMN Processes" in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for more information on how to implement the service task with these types of processes and services.

Figure 6–19 shows the default notation for the service task.

Figure 6–19 The Service Task

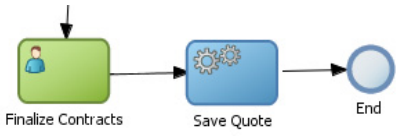
The service task is represented by blue rectangle with two gears in the middle.

****

6.4.1.1 The Service Task in Context

Figure 6–20 shows an example of the service task used to save the finalized sales quote to a database.

Figure 6–20 The Service Task within the Sales Quote Example Process



This graphic shows a user task labeled Finalize Contract with a sequence flow extending to a service task labeled Save Quote.

The Save Quote task has a sequence flow extending to an end event.

****

6.4.1.2 Implementing Reusable Services in Project Templates

Oracle BPM enables you to incorporate reusable services in project templates. These services are components of the business catalog.

6.4.2 Introduction to the Notification Task

The notification task is similar to the service task. It uses a predefined service to perform different types of notification. You can use expressions to determine the users or groups who will receive notifications generated by the notification task.

For more information on implementing the notification task see *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

These different types of notification are:

- **IM**: Send an instant message to a user or group. Figure 6–21 shows the default notation of the IM notification task.

Figure 6–21 The Instant Message Notification Task

The instant message notification task is represented by a blue rectangle with a megaphone icon on the left-hand side. Next to the megaphone is a text balloon.

****

- **Email**: Sends an email a user or group. You can also include email attachments. Figure 6–22 shows the default notation of the email notification task

Figure 6–22 The Email Notification Task

The email notification task is represented by a blue rectangle with a megaphone icon on the left-hand side. Next to the megaphone is a yellow letter icon.

****

- **SMS**: Sends an SMS message to a user. Figure 6–23 shows the default notation of the SMS notification task

Figure 6–23 The SMS Notification Task

The SMS notification task is represented by a blue rectangle with a megaphone icon on the left-hand side. Next to the megaphone is a cellular phone icon.

****

- **Voice**: Sends a voice mail to a user. Figure 6–24 shows the default notation of the voice mail notification task.

Figure 6–24 The Voicemail Notification Task

The voice mail notification task is represented by a blue rectangle with a megaphone icon on the left-hand side. Next to the megaphone is a microphone icon.

****

- **User**: Sends a notification to a user based on the available notification types. For example, if a user has both a voice mail and email configured, the notification task will send both. Figure 6–25 shows the default notation of the User notification task.

Figure 6–25 The User Notification Task

The user notification task is represented by a blue rectangle with a megaphone icon on the left-hand side.

****

6.4.3 Introduction to the Call Activity

The call activity allows you to call a reusable process from within the current process. The process being called becomes a child process of the calling process. When calling a reusable process, the call activity of the parent process waits until the child process completes before continuing.

Figure 6–26 shows the default notation for the call activity.

Figure 6–26 The Call Activity

The call activity is represented by an empty rectangle.

****

Data objects of the parent process are not automatically available to the reusable process. Data objects must be passed to and from the child process using argument mapping of the call activity.

6.4.3.1 Reusable Processes

Oracle BPM supports a type of process called reusable processes. In BPMN terminology, this is sometimes referred to as a reusable subprocess. Reusable processes allow you to create processes that can be called from other BPMN processes.

Reusable processes allow you to create processes that can be called from other BPMN processes. For example all your processes may need to charge a credit card, so you can create a charge credit card reusable subprocess

Reusable processes have the following characteristics:

- Must start with one none start event
- Can contain multiple end events.
- Can only be called by other BPMN processes.

6.4.4 Introduction to the Send Task

The send task sends a message to a system or process outside the current process. After this message is sent, the task is complete and running of the process continues to the next task in the process flow.

The send task is frequently paired with the receive task to invoke a process or service and receive a response in return. The send and receive tasks are used to invoke processes and services asynchronously. If you are invoking a process or service synchronously, use the service task.

Note: The send and receive tasks perform functions similar to the throw and catch message events. However, you cannot use the send task to invoke a process that is initiated with a message start event.

Figure 6–27 shows the default notation for the send task.

Figure 6–27 The Send Task

The send task is represented by blue rectangle with a yellow envelope in the middle. The envelope has an outgoing arrow to represent the send action.

****

For information on implementing the send task to invoke a process or service, see "Using Send and Receive Tasks to Define a Synchronous Operation in a BPMN Process" in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

6.4.4.1 The Send Task in Context

See Chapter 6.4.6, "Using the Send and Receive Tasks to Communicate Between Processes" for information on using the send and received tasks to communicate between processes.

6.4.5 Introduction to the Receive Task

In contrast to the send task, the receive task waits for a message from a system or process outside the current process. After this message is received, the task is complete and running of the process continues to the next task in the process flow.

Figure 6–28 shows the default notation for the receive task.

Figure 6–28 The Receive Task

The receive task is represented by blue rectangle with a yellow envelope in the middle. The envelope has an incoming arrow to represent the receive action.

****

For information on implementing the send task to invoke a process or service, see "Using Send and Receive Tasks to Define a Synchronous Operation in a BPMN Process" in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

6.4.5.1 The Receive Task in Context

See Section 6.4.6, "Using the Send and Receive Tasks to Communicate Between Processes" for information on using the send and receive tasks to communicate between processes.

6.4.5.2 Starting a Process with the Receive Task

You can use the receive task to trigger the start of a process. This is useful when you want to invoke a process from another process using a send task.

To start a process using the receive task, the following conditions must be met:

- The receive task is preceded by an none start event.
- Your process does not contain any other start events.
- The Create Instance property is enabled.

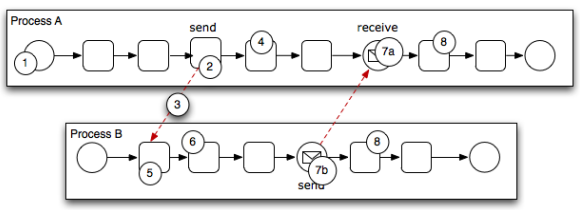
The following section describes how to use the send and receive tasks to communicate between processes.

6.4.6 Using the Send and Receive Tasks to Communicate Between Processes

You can use the send and receive tasks to invoke another BPMN process and receive messages back from it. Processes that begin with a receive task and contain a send task are exposed as services that can be used by other process and services within an Oracle BPM application.

Figure 6–29 outlines the basic behavior when using send and receive tasks to invoke a process and receive a response.

Figure 6–29 Using the Send and Receive Tasks to Communicate Between Processes



This illustration is described in the text.

****

The following steps outline a possible scenario when using the send and receive tasks to communicate between processes.

1. Process A is invoked.
2. A token of Process A reaches the send task.
3. The send task invokes Process B.

   This is defined by the implementation for the send task.
4. The token of process A proceeds to the next flow object in the process.
5. The receive task initiates a process instance of Process B.

   The receive task must have the Create Instance property defined. See Section 6.4.5.2.
6. The newly created token proceeds through process B.
7. Depending on the specific behavior of your process, the following scenarios may occur:
   a. If the token of Process A reaches a receive task paired with a send task from Process B, the token of Process A waits until a response is received. After the response is received, the token of Process A continues to the next flow object.
   b. If the token of Process B reaches a send task paired with a receive task in Process A, Process B sends a response to Process A. The token of Process B continues to the next flow object.
8. Both processes continue running. You can use subsequent send and receive pairs to define subsequent communication between the two processes.

6.4.7 Introduction to the Message Throw Event

The message throw event enables you to send a message to another process or service.

Figure 6–30 shows the default notation for the message throw event.

Figure 6–30 The Message Throw Event

The message throw is represented by two concentric circles with a blue envelope in the middle.

****

The throw message event can be used to invoke the following types of processes and services:

- Other BPMN processes
- BPEL processes
- SOA service adapters
- Mediators that are exposed as services

Process analysts may add message throw events to a process to define where a process must invoke another process or service. However, process developers are typically responsible for implementing the connectivity with other processes. Additionally, they

are typically responsible for creating and implementing the services invoked by the message throw event.

For information on how to implement message throw events, see "Communicating With Other BPMN Processes and Services Using Message Events" in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

Message throw events are often used to invoke other BPMN processes by calling the message start event of another process. See Section 6.2.4, "Introduction to the Message Start Event" for more information.

Message throw events are also frequently used with message catch events to receive a response from the process or service invoked. However, they are always used asynchronously. After the message throw event sends a message to another process or service, the token immediately moves to the next flow object of the process.

If your process receives a response synchronously, use the service task to invoke the process or service. See Section 6.4.1, "Introduction to the Service Task" for more information.

Note: The send and receive tasks perform functions similar to the throw and catch message events. However, you cannot use the message throw event to invoke a process that is initiated with a message receive task.

6.4.8 Introduction to the Message Catch Event

The message catch intermediate event enables you to receive a message from another process or service.

Figure 6–31 shows the default notation for the message catch event.

Figure 6–31 The Message Catch Event

The message catch is represented by two concentric circles with a yellow envelope in the middle.

****

The message catch event is frequently used with the message throw event to communicate with another BPMN process. See Section 6.4.9 for information on how message throw events with message catch event.

For information on how to implement message throw events, see "Communicating With Other BPMN Processes and Services Using Message Events" in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

6.4.9 Using Message Throw and Catch Events to Communicate Between Processes

You can use combinations of throw and catch events to invoke and communicate with other BPMN processes. When using a throw event to invoke another process, the following conditions must be met:

- The process being invoked must be an asynchronous process. Although you can use a message throw event to invoke a synchronous process, there is no mechanism for catching messages synchronously from the process.

If you invoke a synchronous process use the service task. See Section 6.4.1, "Introduction to the Service Task" for more information.

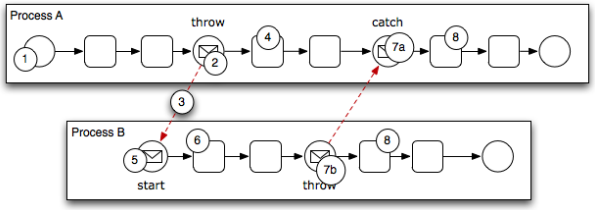
- The first time you use a message throw event it must be paired to the message start event of the other process. This is required to trigger the process instance. After the instance has been triggered, you can use subsequent message throw events that are caught by the second process.

Processes that begin with a message start and end with a message end event are exposed as services that can be used by other process and services within an Oracle BPM application.

Processes invoked from another process are not considered child processes. This is important to consider when designing processes that use the terminate end event as a process end point. For example, a terminate event in the calling process does not stop processes invoked with a message throw event.

Figure 6–32 shows the basic behavior when using message throw and catch events to invoke a process and receive a response.

Figure 6–32 Using Message Throw and Catch Events Between Processes



This figure is described in the text.

****

The following steps outline a possible scenario when using the message throw and catch events to communicate between processes.

1. Process A is invoked.
2. The token of Process A reaches a message throw event that is configured to invoke Process B.
3. The message throw event sends a message to the message start event of Process B.
4. The token of Process A proceeds to the next flow object.
5. The message start event triggers an instance of Process B.
6. The newly created token proceeds through Process B.
7. Depending on the behavior of your process, the following scenarios may occur:
   a. If the token of Process A reaches a catch event paired with a throw event from Process B, the token of Process A waits until the message is received. After the message is received, the token of Process A continues to the next flow object.
   b. If the token of Process B reaches a throw event paired with a catch event in Process A, Process B throws a message to Process A. The token of Process B continues to the next flow object.
8. Both processes continue running. You can use subsequent catch and throw event pairs to define subsequent communication between the two processes.

6.5 Adding Business Logic Using Oracle Business Rules

This section describes how to use the business rule task to incorporate Oracle Business Rules within your business processes. See Chapter 8, "Using Oracle Business Rules" for information on working with Oracle Business Rules using Business Process Composer.

6.5.1 Introduction to Oracle Business Rules

Business rules are statements that describe business policies or describe key business decisions.

6.5.2 Introduction to the Business Rule Task

The business rule task enables you to incorporate Oracle Business Rules within your process.

Figure 6–33 shows the default notation for the business rule task.

Figure 6–33 The Business Rule Task

The business rule task is represented by a rectangle with a gear icon in the center. The gear has two green arrows pointing outward.

****

There are two primary use cases for incorporating Oracle Business Rules within your business process.

- Using structural rules

  Structural rules enable you to perform calculations used within your business process. For example, you can use a business rule to calculate a credit score.

- Using operative rules

  Operative rules are used to make changes to the flow of your process. A typical use of an operative rule is to perform a check of the rule conditions within the rules catalog. Then, as part of the output data association, assign a value to a data object using an expression.

  In this scenario, the business rule task is immediately followed by a gateway that is used to branch the process path according to the value of the data object.

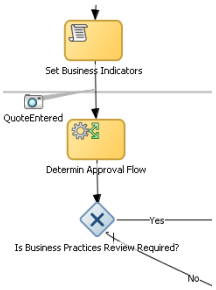
  See Section 6.5.2.1, "The Business Rule Task in Context" for information on how an operation rule is used within the Sales Quote example.

See Section 5.5.1, "How to Assign a Business Catalog Component to a Flow Object" for information on for procedures on how to assign elements from the business catalog to a business rule task.

6.5.2.1 The Business Rule Task in Context

Figure 6–34 shows an example of the business rule task within the Sales Quote example.

Figure 6–34 The Business Rule Task within the Sales Quote Example Process



This illustration is described in the text.

****

6.6 Controlling Process Flow Using Sequence Flows

This section describes how to use sequence flows to define the behavior of your business process.

6.6.1 Introduction to Sequence Flows

Sequence flows define the order or sequence that work is performed within a process. Sequence flows connect the flow objects within your process and determine the path a process token follows through your process.

Incoming sequence flows are the sequence flows that lead into a flow object. Outgoing sequence flows are the sequence flows that determine the process path out of a flow object.

Most flow objects contain both incoming and outgoing sequence flows. Exceptions to this are start and end events. Start events can only contain outgoing sequence flows. End events can only contain incoming sequence flows. Additionally, event subprocesses do not have either incoming or outgoing sequence flows.

6.6.2 Introduction to Unconditional Sequence Flows

Unconditional sequence flows represent the typical path between two flow objects. Default sequence flows are displayed as a line with and arrow as shown in Figure 6–35.

Figure 6–35 The Unconditional Sequence Flow

This illustration is described in the text.

****

Most flow objects can contain only one default out going sequence flow. Only parallel gateways can contain multiple unconditional sequence flows which represent the parallel paths of your process.

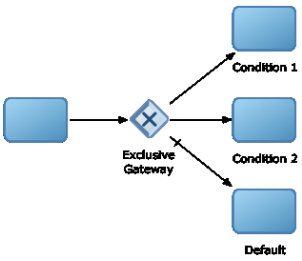
Exclusive, inclusive, and conditional gateways cannot have unconditional outgoing sequence flows. These gateways use conditional and default sequence flows to determine the flow of your process.

6.6.3 Introduction to Conditional Sequence Flows

Conditional sequence flows are used to control the flow of a process based on certain conditions. Like unconditional sequence flows, conditional sequence flows are displayed by an arrow lined arrow.

Figure 6–36 shows two outgoing conditional sequence flows and a default sequence flow.

Figure 6–36 Conditional and Default Sequence Flows



This graphic shows a process flow that passes to an exclusive gateway. From the exclusive gateway, there are two arrowed lines representing conditional sequence flows that connect to two rectangles representing the conditional parts of a process flow. There is also an arrowed line with tic mark representing the default flow of the process path that is followed when none of the conditions evaluate to true.

****

Not all flow objects can use outgoing conditional sequence flows. Only the following types of gateways can have outgoing conditional sequence flows:

- Exclusive gateways
- Inclusive gateway (split)

The conditions used within a conditional sequence flow are defined using expressions. See Section 10.4, "Introduction to Expressions" for information on using the expression editor to define expressions.

6.6.4 Introduction to Default Sequence Flows

Like conditional sequence flows, default sequence flows are used as outgoing sequence flows to exclusive, inclusive, and conditional gateways. Default sequence flows represent the path your process will take out of these gateways when none of the conditions evaluate to true.

Default sequence flows are represented by an arrowed line with a tic mark on one end as shown in Figure 6–36.

6.7 Controlling Process Flow Using Gateways

This section describes how to use gateways to control process flow and behavior.

6.7.1 Introduction to Gateways

Gateways are flow elements that define the flow of your process. Gateways determine the path a token takes through a process. They define control points within your process by splitting and merging paths.

When possible, gateways are used for paths that are exceptions to or deviate from the default path of the process.

6.7.1.1 Split-Merge Pairs

The following gateways require a split-merge pair:

- Parallel gateway
- Inclusive gateway
- Complex gateway

When you add one of these gateways to a BPMN process, Oracle BPM Studio automatically creates the split and merge flow objects.

Although the merge portion of the gateway is required, you do not have to ensure that all paths out of the split return to the merge.

Although it is possible to have process paths that split at a gateway without merging through the gateway, this is not usually good practice. For more details on the merge behavior of gateways, see the following sections for each gateway type.

Note: If you delete the merge gateway from a process, the corresponding split gateway is also deleted.

6.7.2 Introduction to the Exclusive Gateway

The exclusive gateway enables you to split your process into two or more paths. However, the process only continues down one of these paths even if multiple outgoing sequence flows are present. Exclusive gateways can have conditional outgoing sequence flows and must have at least one default outgoing sequence flow.

You can define expressions that are used to determine if your process continues down a conditional sequence flow. See Section 10.4, "Introduction to Expressions" for more information.

If your process has multiple outgoing sequence flows for an exclusive gateway, you can define the order in which they are evaluated. The order of evaluation is configured in the properties of the exclusive gateway.

If you have an exclusive gateway where more than one conditional evaluates to true, the process will continue down the first conditional sequence flow determined by this order.

Unlike other gateways, the exclusive gateway does not require a corresponding merge to be explicitly defined in your process after splitting.

Figure 6–37 shows the default notation for the exclusive gateway.

Figure 6–37 The Exclusive Gateway

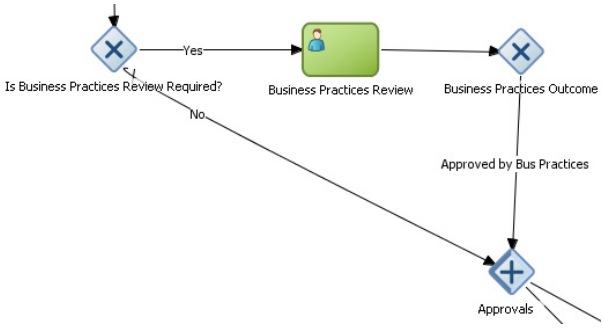
The exclusive gateway is represented by a diamond-shaped icon with an X in the middle.

****

6.7.2.1 The Exclusive Gateway in Context

Figure 6–38 shows an example of the exclusive gateway used within the Sale Quote example. Here, the exclusive gateway is used to evaluate whether a review of business practices is required.

Figure 6–38 The Exclusive Gateway within the Sales Quote Example Process



This figure shows an example of the exclusive gateway within the Sales Quote example. It contains an exclusive gateway that splits the process path in two, one path represents yes, the other no.

****

This evaluation is determined by the expression defined for the outgoing conditional sequence flow. If this evaluates to true, then the process flow proceeds down the Yes path. If it evaluates to false, then the process flow proceeds down the path of the default outgoing sequence flow.

6.7.2.2 Splitting and Merging Exclusive Gateways

When a token reaches an exclusive gateway the outgoing conditional sequence flows are evaluated until one of them evaluates to true. You can define the specific order in which these are evaluated by configuring a property for the exclusive gateway.

Based on this configuration, when the first conditional sequence flow evaluates to true, the token moves down this outgoing sequence flow to the next flow object. If you have multiple outgoing conditional sequence flows, you can determine the order in which they are evaluated.

If none of the outgoing conditional sequence flows evaluate to true, then the token moves down the default outgoing sequence flow. Therefore, you must define a default outgoing sequence flow for the exclusive gateway.

The exclusive gateway can also merge incoming sequence flows. However, there is no synchronization with other tokens that may be coming from other paths within the process flow.

> **Note:** If other tokens arrive at an exclusive gateway merge, then they are also passed through as is. If you are synchronizing tokens or perform evaluations on incoming sequence flows, you should use a different type of gateway.

6.7.3 Introduction to the Inclusive Gateway

The inclusive gateway enables you to split your process into two or more paths. Unlike the exclusive gateway, however, a token may flow down one or more of these paths depending on how the outgoing conditional sequence flows are evaluated.

You can have multiple outgoing conditional sequence flows for an inclusive gateway split. You must define at least one default sequence flow.

Figure 6–39 shows the default notation for the inclusive gateway split.

Figure 6–39 The Inclusive Gateway (Split)

The inclusive gateway split is represented by a diamond-shaped icon with a circle in the middle. It is shaded on the left-hand side to represent the split of the inclusive gateway.

****

Figure 6–40 shows the default notation for the inclusive gateway merge.

Figure 6–40 The Inclusive Gateway (Merge)

The inclusive gateway merge is represented by a diamond-shaped icon with circle in the middle. It is shaded on the right-hand side to represent the merge of the inclusive gateway.

****

6.7.3.1 Splitting and Merging Inclusive Gateways

The inclusive gateway splits a process similar to the exclusive gateway, but enables tokens to proceed down multiple outgoing sequence flow. When a token arrives at an inclusive gateway, the expressions of its conditional sequence flows are evaluated.

Next, a token is generated for each of the conditional sequence flows that evaluates to true. A token is generated for the default sequence flow only if none of the conditional sequence flows evaluates to true.

These tokens are joined at the merge of the inclusive gateway. When a token reaches the merge gateway, it waits until all of the tokens generated by the split have reached the merge. After all of the tokens have reached the merge of the inclusive gateway, the merge is complete, and the token continues to the next sequence flow after the gateway.

6.7.4 Introduction to the Parallel Gateway

The parallel gateway enables you to split your process into two or more paths when you want your process flow to follow all paths simultaneously. The parallel gateway is useful when your process must perform multiple tasks in parallel.

Figure 6–41 shows the default notation for the parallel gateway split.

Figure 6–41 The Parallel Gateway (Split)

The parallel gateway split is represented by a diamond-shaped icon with a plus sign in the middle. It is shaded on the left-hand side to represent the split of the parallel gateway.

****

Figure 6–42 shows the default notation for the parallel gateway merge.

Figure 6–42 The Parallel Gateway (Merge)

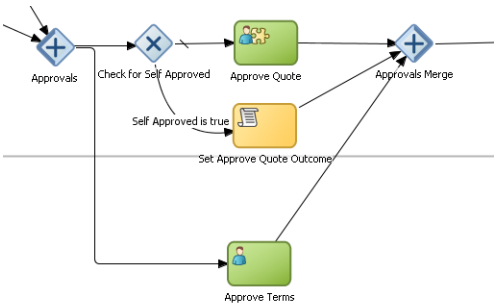
The parallel gateway merge is represented by a diamond-shaped icon with a plus sign in the middle. It is shaded on the right-hand side to represent the merge of the parallel gateway.

****

6.7.4.1 The Parallel Gateway in Context

The Sales Quote example uses a parallel gateway during the approval stage of the process. Figure 6–43 shows how the parallel gateway is used to perform two process paths simultaneously.

Figure 6–43 Example of a Parallel Gateway



This figure shows an example of the parallel gateway. It contains the split and merge halves of the parallel gateway. Between them are two parallel paths represented by the Approve Quote and Approve Terms users tasks.

****

In this example, two different process paths are executed at the same time.

6.7.4.2 Splitting and Merging Parallel Gateways

When a token reaches a parallel gateway, the parallel gateway creates a token for each outgoing sequence flow. The split of the parallel gateway does not evaluate outgoing sequence flows.

You can also use the parallel gateway to merge process paths split by the parallel gateway. The merge of the parallel gateway waits for a token to arrive from each of the

incoming sequence flows. After all tokens arrive, only one token is passed to the outgoing sequence flow.

Note: Design your process so that a token arrives for each incoming sequence flow for the merging parallel gateway. If you do not, your process can freeze if the merge is expecting tokens that do not arrive.

6.7.5 Introduction to the Complex Gateway

The complex gateway splits a process similar to an inclusive gateway. However, it enables you to define an activation condition that determines if the instance can continue even if not all of the tokens have arrived at the complex gateway merge.

For example, you can configure a complex gateway to continue after two or more tokens have arrived. If only two out of the possible conditions in the inclusive gateway evaluate to true the process instance continues to the next activity. However because the inclusive gateway immediately evaluates all the conditional sequence flows, all of the flow objects in these process paths are also run.

Figure 6–44 shows the default notation for the complex gateway split.

Figure 6–44 The Complex Gateway (Split)

The complex gateway is represented by a diamond with an asterisk in the middle.

****

Figure 6–45 shows the default notation for the complex gateway merge.

Figure 6–45 The Complex Gateway (Merge)

The complex gateway is represented by a diamond with an asterisk in the middle.

****

6.7.6 Introduction to the Event-based Gateway

The even-based gateway enables you to branch your process flow based on the possibility that an event may occur. Depending on the context, this may be one of several types of events.

The event-based gateway enables you to anticipate the possibility that several types of events may occur at a specific point in your process. It is similar to the exclusive gateway, but instead of choosing a path based on expressions, the event-based gateway chooses a path based on the occurrence of an event within your process.

For example, in an order processing process, you may reach a point in your process when there is no stock currently available. The process may need to wait until stock is available, but cannot wait indefinitely. By using an event-based gateway, your process can wait for a message saying new stock has been received (using a message catch event) or it can continue if no message is received after a certain amount of time has passed (using a timer event).

Figure 6–46 shows the default notation for the event-based gateway.

Figure 6–46 The Event-based Gateway

The event-based gateway is represented by a diamond-shaped icon with hexagon in the middle.

****

The event-based gateway is different from other gateways in that decisions about process flow are based on an event rather than data-specific conditions.

The event-based gateway is composed of the following:

- The event-based gateway
- Two or more target events. These can be of the following types:
  - Message catch events

    When initiating a process using a message catch event, the process must be invoked using a message throw event.
  - Timer catch events

    Generally only one timer event is used following an event-based gateway.
  - Receive tasks

    You can use the receive task to initiate a process instance following an event-based gateway. However the process must be invoked from a send task within the calling process.

> **Note:** You cannot mix message events and receive tasks within the same event-based gateway.

The target elements can only have incoming sequence flows from the event-based gateway. They cannot have sequence flows from other parts of the process.

Although the event-based gateway enables you to plan that multiple events may occur in your process, within the process instance, only one event is triggered. When the first event in the event-based gateway is triggered, the path that comes after that event is followed.

By default, when you add an event-based gateway to a process, it is created with a timer and message catch event.

> **Note:** If you delete an event-based gateway, any outgoing sequence flows are also deleted. The associated events are not deleted.

6.7.6.1 Starting a Process with an Event-Based Gateway

You can also use an event-based gateway at the beginning of a process to create a new process instance. This is similar to having multiple start events within a process.

To enable an event-based gateway to create a new process instance, you must ensure the following:

- You have enabled the Initiate property of the event-based gateway.
- There are no incoming sequence flows to the event-based gateway.

Although the event-based gateway can be used to create a new process instance, it does not accept data input from another process. Any data that must be passed to the process instance must be configured using the target events.

6.8 Controlling Process Flow Using Intermediate Events

This section describes intermediate events and describes how to use them to control the flow and behavior of your process.

6.8.1 Introduction to Intermediate Events

Unlike start and stop events, intermediate events occur during the flow of your process.

There are two types of intermediate events:

- Normal flow events

  Normal flow events occur within the typical flow of your process.

- Boundary events

  Boundary events trigger an interruption with your process. Boundary events are associated with flow objects and can be configured to interrupt their usual behavior.

  Boundary events behave similar to sequence flows in that they are used to determine the path a process takes between flow objects.

  Boundary events can be divided into two types: interrupting and non interrupting.

6.8.2 Introduction to the Timer Catch Event

Timer catch events enable you to control the flow of your process using a time condition. Possible uses of the time catch event include:

- Creating a delay before running an activity
- Configuring a deadline for an activity
- Configuring a deadline for a process
- Triggering additional activities after an elapsed time

Figure 6–47 shows the default notation for the timer catch event.

Figure 6–47 The Timer Catch Event

The timer catch event is represented by a two concentric circles with a clock icon in the middle.

****

You can use timer events as boundary events on an activity. Timer events can be defined as either interrupting or non interrupting boundary events.

When an interrupting timer event fires, the token leaves the main process flow to follow the flow the timer event defines. The flow that an interrupting timer event can return directly to the main process flow.

When a non interrupting event fires, a copy of the token is created and passes through the flow the timer event defines. The flow that a non interrupting event defines cannot return to the main process flow.

6.8.3 Introduction to the Error Catch Event

Error catch events are intermediate events used to handle an error that occurs within your process flow.

> **Note:** You can also use inline handlers to handle error conditions that occur within your process.

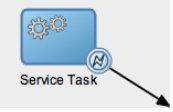
Error catch events are always used as boundary events and can be attached to the following:

- Service Tasks
- Call Activities
- User Tasks
- Send Tasks
- Receive Tasks
- Script Tasks
- Rules Tasks
- Subprocesses

Error catch events are always interrupting, meaning that they interrupt the usual flow of a process.

Figure 6–48 shows the default notation for the error catch event attached as a boundary event on a service task.

Figure 6–48 The Error Catch Event as a Boundary Event on a Service Task



This figure shows a service task with the error catch event as a boundary event. The service task is represented by a rectangular box containing two gears in the center. The error catch event is represented by two concentric circles with a lightning bolt in the center.

****

When a service or process fails with an error, the error catch event is triggered. This causes the process flow to follow the path of the outgoing sequence flow of the error catch event.

You can use this flow to define how you handle the error. This is handled in two ways:

- The process flow returns to the main process flow. Any work that must be performed is handled within the error process flow before returning to the main flow.

  Note: If the boundary event is non-interrupting, the boundary flow cannot return to the main flow.

- The process flow continues to an end event. The process is stopped immediately. Process control is returned to the service or process that initiated the process.

6.9 Using Subprocesses and Inline Handlers to Organize Your Process

You can use subprocesses and inline handlers to organize your processes. This enables you to group functional areas together and also makes your processes more readable.

6.9.1 Introduction to Subprocesses

Subprocesses allow you to group BPMN flow objects together to make your process more readable. In Oracle BPM, subprocesses are embedded subprocesses. Subprocesses are contained as part of the parent subprocess. Subprocesses must begin with a start none event and must end with a none end event.

Subprocesses can be expanded or collapsed. Figure 6–49 shows how a collapsed subprocess appears within a process.

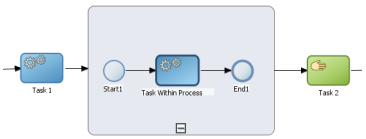
Figure 6–49 Example of a Collapsed Subprocess

This figure shows an example of a collapsed subprocess. It shows three flow objects in a straight line: A service task, a subprocess, and a user task. Each is connected by a sequence flow. The collapsed subprocess displays a plus icon that expands the subprocess.

****

Figure 6–50 shows how an expanded subprocess appears within a process. When a subprocess is expanded, you can edit the flow objects within it. You can also click and drag the edge of the subprocess window to make the window larger or smaller.

Figure 6–50 Example of an Expanded Subprocess



This figure shows an example of an expanded subprocess. It shows three flow objects in a straight line: A service task, a subprocess, and a user task. Each is connected by a sequence flow. The expanded subprocess displays the flow objects contained within the subprocess. It also displays a minus icon that can be used to collapse the subprocess.

****

Similar to other types of processes, subprocesses have can contain start and end events and contain a separate process flow. A subprocess must begin with a none start event and end with a none end event. Subprocesses cannot contain swimlanes.

Subprocesses also behave similar to activities. They can have incoming and outgoing sequence flows. They also contain data associations that define the data objects used within the subprocesses.

Subprocesses can also contain timer, message, and error boundary events.

If necessary, your process can contain nested subprocesses. However, use nested subprocesses only when necessary to make your process more readable.

6.9.1.1 Subprocesses and Sequence Flows

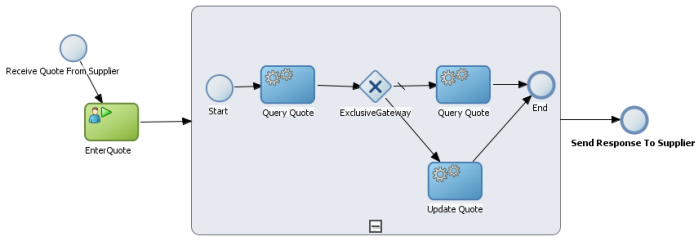
The flow objects within a subprocess cannot have sequence flows that connect to flow objects outside the subprocess.

Similar to other flow objects, subprocesses have incoming and outgoing sequence flows.

6.9.1.2 Subprocesses in Context

Figure 6–51 show an example of a subprocess. In this example, a subprocess is used to group the service task used to process a sales quote.

Figure 6–51 Example of a Subprocess



This graphic shows a start event labeled Receive Quote From Supplier, which has a sequence flow extending to a User task labeled EnterQuote.

A sequence flow extends from the EnterQuote task to the expanded subprocess, illustrated as a rounded rectangle containing a process.

From the subprocess, a sequence flow extends to an end task labeled Send Response to Supplier.

****

6.9.1.3 Looping Subprocesses

You can configure a subprocess to repeat numerous times within the context of a process flow. This is something that a process analyst should consider when designing a process, but the implementation is performed by process developers.

> **Note:** Looping cannot be configured using Oracle Business Process Composer. You must used Oracle BPM Studio to configure looping. See the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for more information.

6.9.2 Introduction to Inline Handlers

Inline handlers are types of subprocesses that allow you to model conditions that happen outside of a normal process flow.

Inline handlers can have the following start events:

- Error start
- Timer start
- Message start

When you add an inline handler to your process, by default it is created with a message start event. You can change the type of start event if necessary.

Note: Inline handlers can only contain one start event. However, you can have multiple inline handlers within your process.

6.10 Changing the Value of Data Objects in Your Process

This section describes how to use the script task to change the values of data objects within your process. See Chapter 10, "Working with Data Objects and Expressions" for general information about data objects.

6.10.1 Introduction to the Script Task

The script task is used to change values of data objects within your process. The script task is used when you want to show this change explicitly within your business process or when you must change the values of data objects outside of another flow object. It is often used to set initial values of data objects at the beginning of a process.

Figure 6–52 shows the default notation for the script task.

Figure 6–52 The Script Task

The script task is represented by a yellow rectangle with a scroll icon in the middle.

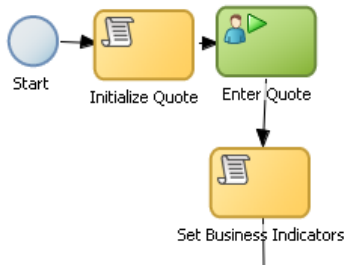
****

Script tasks are added to a process by process developers who are responsible for defining the behavior of data objects within a process and process-based application.

6.10.1.1 The Script Task in Context

Figure 6–53 shows two examples of the script task used at the beginning of the Sales Quote example. The Sales Quote example uses a script task to set initial values for data objects when the process instance is created and to set values for several business indicators.

Figure 6–53 The Script Task within the Sales Quote Example



This graphic shows a start event with a sequence flow extending to a script task labeled Initialize Quote.

From the initialize quote task, a sequence flow extends to a user task labeled Enter Quote.

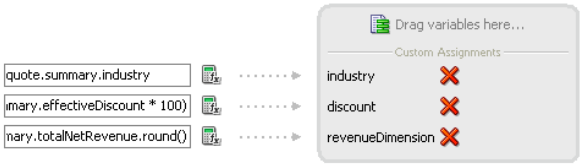
From the Enter Quote task, a sequence flow extends to a script task labeled Set Business Indicators.

****

Project data objects are data objects that you define in a project. All processes within a project have access to the data object defined, though the value changes according to the process using them. In addition, the engine stores the value of those marked as business indicators to the process analytics databases if the project is configured to use them.

Figure 6–54 shows the data associations used to set initial values for the business indicators.

Figure 6–54 Data Associations Used by the Set Business Indicators Script Task

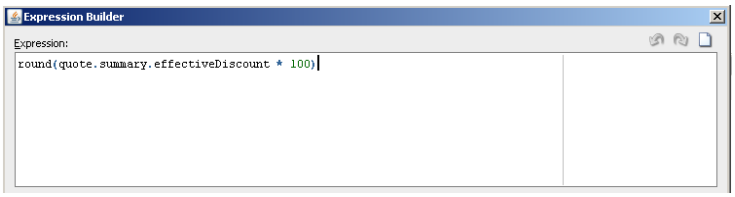


This figure shows how the data associations editor appears for the Set Business Indicators script task in the Sales Quote example. It contains two columns. On the left is a list of data objects representing the input arguments to the script task. On the right are the three data objects that these are mapped to.

****

As with other flow objects that accept data associations, you can use expressions to change the values of data objects. Figure 6–55 shows how an expression is used to change the value of the discount project variable.

Figure 6–55 Expression Used to Change the Value of Discount Project Variable



This figure shows an example of the expression editor show the expression used to define the value of the discount project variable. The value of the expression is:

round(quote.summary.effectiveDiscount * 100).

****

Note: If you implemented the script task using transformations using BPM Studio, they will not be visible when opening the BPM project in Business Process Composer.

6.11 Measuring Process Performance Using Measurement Marks

You can measure process performance using measurement marks. Measurement marks enable you to measure a business indicator of type measure at a certain point in the process or in a section of the process.

For more information on using measurement marks and the Process Analytics database, see "Using Process Analytics" in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

A measurement mark stores the following data into the Process Analytics databases:

- The value of the process default measures
- The value of the measure business indicators associated with that measurement mark
- The value of the dimensions defined in the process

You can use one measurement mark to measure multiple business indicators.

When storing the value of a measure business indicator, the BPMN service engine also stores the value of the dimensions you defined in your process. Later on, when you build the dashboards to monitor your process, you can use these dimensions to group the values into different categories. For example, in the Sales Quote example you might want to view the total number of quotes approved by region.

The types of measurement marks you can define are:

- Single measurement
- Interval start
- Interval stop

6.11.1 How to Add a Measurement Mark to a Process

You can add measurement marks to your business processes by dragging the from the component palette to the process editor canvas.

To add a single measurement mark to a process:

1. Open the BPMN process.
2. In the **Component Palette**, expand the **Artifacts** section and select from following:
   - Start measurement mark
   - End measurement mark
   - Single measurement mark (Snapshot)
3. Place the measurement mark near the sequence flow where you want to measure a business indicator. When the sequence flow turns blue, drop the measurement mark.
4. Right-click the measurement mark and select **Properties**.
5. In the **Name** field, enter a name to identify the measurement mark.
6. In the Business Indicators section, select a business indicator from the list of available business indicators and move it to the Selected list using the arrows between the two lists.

   Note: You can measure multiple business indicators in the same measurement mark.

   Note: If you do not select a business indicator, then this measurement mark only stores the value of the default business indicators. If you want to add a business indicator without leaving the Measurement Mark Properties dialog, then click the **New** button under the **Selected** list.

7. Click **OK**.

7

Working with the Project Life Cycle

This chapter describes some of the advanced features related to BPM projects and how to work with them within the development life-cycle.

This chapter includes the following sections:

- Section 7.1, "Importing and Exporting Projects"
- Section 7.2, "Using BPM Project Templates"
- Section 7.4, "Configuring Approval Workflow for a Project"
- Section 7.5, "Deploying a Project"

7.1 Importing and Exporting Projects

Business Process Composer enables you to import and export BPM projects as .exp files. This enables you to share projects directly with other Business Process Composer and BPM Studio users directly through the files system without having to use the BPM repository.

7.1.1 How to Import a Project from Your Local File System

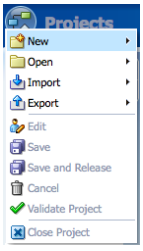
You can import a BPM project that was previously exported and saved as a .EXP file. The imported project is stored in the BPM repository.

To import a project:

1. Access the application welcome page.
2. From the main menu, select **Import**, then select **Import Project**

   The main menu is accessible from the upper left-hand corner of the Business Process Composer user interface as shown in Figure 7–1.

Figure 7–1 The Application Composer Main Menu



This graphic displays the Application Composer Main Menu.

****

3. Click **Browse**, then select the project file you want to import.
4. Click **OK**.
5. Enter a name for the project.

   After you select a project file to import, the name field is automatically propagated.
6. Enter an optional description.
7. If you want to optionally select a folder within the BPM repository where the imported project will be created, click **Browse**, then select the folder.
8. Click **OK** to import the project into the BPM repository.

7.1.2 How to Export a Project to Your Local File System

Projects exported from Business Process Composer can be imported into Oracle BPM Studio. Exporting a project to your local file system enables you to share projects without using the Oracle BPM MDS repository.

For information on importing a project into Oracle BPM Studio see the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

To export a project as an EXP file:

1. Open the project you want to export.
2. From the main menu, select **Export**, then select **Export Project**.
3. Choose a location on your local file system and click **Save**.

Your exported project is saved as a.EXP file on your local file system.

To export a project as an Oracle Tutor file:

1. Open the project you want to export.
2. From the main menu, select Export, then select **Export Oracle Tutor**.
3. Select one of the following:
   - **Active process**: Selects the currently active process. This option is only displayed if a process has already been opened and focus is currently on that process.
   - **All open processes**: Exports only the processes that are currently open.
   - **All processes of project**: Exports all of the processes of the current project.
4. Click **OK**.

The file is saved as a ZIP file on your file system.

7.2 Using BPM Project Templates

The following sections provides information how to create and use Oracle BPM projects.

7.2.1 Introduction to Project Templates

Project templates enable business users to quickly create custom Oracle BPM applications and deploy them to runtime without assistance from developers. Using Oracle Business Process Composer business users can create new BPM projects based on project templates. These projects contain BPMN process flows and can be deployed directly to run time.

Project templates enable you to incorporate reusable components and services including Human Tasks, Business Rules, and Adapters. These services are stored as part of the business catalog.

Business Process Composer users who have administrative privileges can also import project templates directly from their local file system. See Chapter 12, "Performing Administrative Tasks" for more information.

Using Oracle BPM Studio process developers can convert a normal project to a template and publish it to the Oracle BPM repository. After a template is available in the repository, you can create new projects based on the template using Business Process Composer.

See the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for more information on creating process templates.

Project templates are based on normal Oracle BPM projects and generally have all the required implementation and services defined in the business catalog. However, they often do not have all of the required services assigned to the necessary flow objects. After creating a new project based on a project template, a business analyst can assign business catalog components to the necessary flow objects.

The specific services required for each activity are defined by the editor policies of the project template. After the process analyst incorporates the required services, the project can be deployed to Oracle BPM run time.

7.2.1.1 Introduction to Edit Policies

Project templates also allow you to define edit policies for processes and flow objects within a process. Edit policies determine what parts of a process can be changed or edited when creating a new project based on a project template. Edit policies are defined for the entire process. However, you can also define edit policies for individual flow objects.

See the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for information on defining edit policies in a process template.

Edit policies allow the creator of a project template to define what elements of a process can and cannot be changed when a project is created from a template.

> **Note:** You cannot change the edit policy settings of processes and elements using Business Process Composer.

7.2.1.1.1 Process Level Edit Policies Within a project template, each process contains an edit policy which determines the changes you can make to the process using Business Process Composer.

Table 7–1 describes the process level edit policies.

Table 7–1 Process Level Edit Policies

Edit Policy	Description
Activity Sealed	User cannot make any changes to the activities within the process.
Flow Sealed	The overall flow of the process cannot be changed. A user can edit specific implementation details, but cannot change the process flow

7.2.1.1.2 Component Level Edit Policies Within a process, there are also edit policies that apply to the flow objects within a process. Component level edit policies can be configured for the flow objects within a process.

Table 7–2 describes the edit policies values that can be configured for component level edit policies.

Table 7–2 Component Level Edit Policies

Edit Policy	Description
Sealed	The flow object cannot be modified
Can modify implementation	The user may redefine this flow object if necessary.
Must implement	The user is required to assign a component from the business catalog to this flow object for it to function correctly.
Use process permission	Uses the default edit policy defined by the process.

7.2.1.2 Introduction to Using Data Objects and Variables in Project Templates

A project template can define the data objects used within a project. These can be the Oracle BPM default types or complex data objects created by process template developers within Oracle BPM Studio.

When editing a project based on a project template in Business Process Composer, you can add and create new data objects as necessary. However, you can only create new data objects based on types that are previously defined in the project template. You cannot create new types of complex data objects.

You can use any of the data objects defined in a project template in data associations and expressions. See Section 10, "Working with Data Objects and Expressions" for more information.

7.2.2 Creating a Project Based on a Project Templates

Using Business Process Composer you can create new projects based on project templates.

To create a new project from a project template:

1. Launch Oracle Business Process Composer.
2. From the main menu select **New**, then select **New Project**.
3. Enter a name for you project
4. Select **Use Template**, then click **Choose**.
5. Select the template you want to use.
6. Optionally, choose the folder where you want to store the new project.

7. Click **Next**
8. From the drop-down list, choose the type of approval routing you want to configure for the project.

> **Note:** You can change the type of approval routing after you create the new project.

9. Click **Choose**, then select **Users** or **Groups** from the drop-down menu in the browser.
10. Click **Search** to see a list of available users or groups.
11. Select an item from the **Available** list, then click **Move selected items to other list**.
12. Click **OK**.
13. Click **Finish** to create the new project.

7.3 Using Project Snapshots

The following sections describe how to use project snapshots.

7.3.1 Introduction to Project Snapshots

A project snapshot is a read-only copy of a project at a particular moment. Since snapshots are read-only, they cannot be modified or opened for editing. You can view the contents of a project snapshot as well as export and deploy a project based on a snapshot.

7.3.2 Working with Project Snapshots

The procedures in this section describe how to create and manage project snapshots.

7.3.2.1 How to Create a New Project Snapshot

You can create a new project snapshot from the project welcome page.

To create a new project snapshot:

Users with owner and editor permissions on a project can create new snapshots.

1. Go to the project welcome page, then expand **Snapshots**.
2. Click **New**.
3. Enter a name for your snapshot, then click **Create Snapshot**.

The snapshot appears in the list of snapshots defined for this project, including the date the snapshot was created and the user ID of the snapshot creator.

7.3.2.2 How to View the Contents of a Project Snapshot

Viewing the contents of a project snapshot enables you to view and compare previous version of a project with the current one.

To view the contents of a project snapshot:

1. Go to the project welcome page.
2. Expand **Snapshots**, then click the name of the snapshot you want to view.

Within the snapshot view, you can view the state of the processes, rules, and human tasks associated with the project.

7.3.2.3 How to Return to the Active Version of a Project

To return to the active version of a project from a project snapshot, click the **Back to Active Version** button at the top of the project welcome page.

7.3.2.4 How to Delete a Project Snapshot

A user who is granted the editor role can delete the snapshots they created. A user who is granted the owner or an administrator role can delete any snapshot created by any user.

1. Open your project.
2. From the project welcome page, expand **Snapshots**.
3. Select the snapshot you want to delete from the list, then click **Delete**.
4. Click **Yes** to confirm that you want to delete the project snapshot.

   Once you delete a project snapshot, it cannot be recovered.

7.3.2.5 How to Export a Project Snapshot

1. Open your project.
2. View the project snapshot you want to export.
3. From the main menu, select **Export**, then **Export Project**.
4. Choose a location on your local file system and click **Save**.

   The exported project snapshot is saved as a .EXP file on your file system.

7.3.2.6 How to Deploy a Project Snapshot

1. Open your project.
2. View the project snapshot you want to deploy.
3. From the main menu, select **Deploy Project**.
4. Provide the information as shown in Figure 7–2.
5. Click **Deploy**.

7.4 Configuring Approval Workflow for a Project

The following sections describe how to configure the approval workflow for a project.

7.4.1 Introduction to Approval Workflow

Approval workflow defines the approval process that must be followed before an Oracle BPM project can be deployed to run time using Business Process Composer. You can define the approval workflow when you create a new project, or you can configure it later. You can view the approval workflow defined for a project in the project properties box of the project welcome page.

Table 7–3 describes the different types of approval workflow available in Oracle BPM.

Table 7–3 Approval Routing

None	No approval workflow is specified. Any user with the correct permissions can deploy the project directly to Oracle BPM run time.
Simple	Specifies only one approver in the workflow.
Simple Sequential	Specifies a sequential list of approvers. All approvers must take action in order to approve deployment of the project.
Simple FYI	Sends notification that a project is being deployed, but approval workflow does not wait for approvers to process the task.
Parallel	Specifies a list of approvers who must approve the project deployment concurrently and in parallel.

7.4.2 Working with Approval Workflow

The following sections describe how to configure approval workflow for a project and how to perform approvals as part of the workflow.

7.4.2.1 How to Configure Approval Workflow for a Project

Business Process Composer enables you to configure the type of approval workflow when you create a new project. You can also configure approval workflow after the project has been create.

To configure approval workflow for a project:

1. Go to the project welcome page.
2. Ensure that you are editing the project.
3. Click **Edit** next to the approval workflow in the project information area.
4. Select the type of approval workflow, then click **Apply**.
5. Click **Choose**, then select **Users** or **Groups** from the drop-down menu in the browser.
6. Click **Search** to see a list of available users or groups.
7. Select an item from the **Available** list, then click **Move selected items to other list**.
8. Click **OK**.
9. Click **Apply**.

7.5 Deploying a Project

You can use Business Process Composer to deploy a project to the Oracle BPM run time. Deployment is available only within the same environment where the Business Process Composer application is installed.

7.5.1 Who Can Deploy Projects?

Users who are granted project owner permissions can use Business Process Composer to deploy projects directly to Oracle BPM run time.

7.5.2 How to Deploy a Project to Run Time

Project owners can deploy projects directly to Oracle BPM run time. The following procedures define how to deploy a project when approval workflow is not enabled.

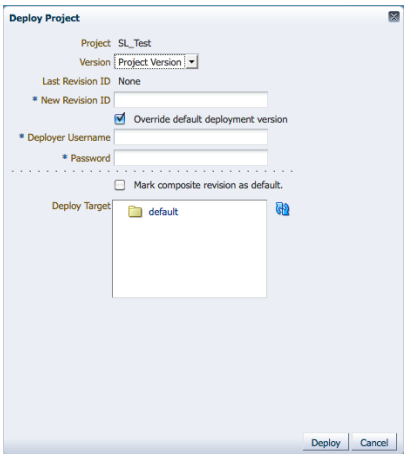
To deploy a project to run time:

1. From the main menu select **Deployment**, then select **Deploy Project**.**Deploy Project**.

   Business Process Composer validates the project. If there are no errors in the project, the deployment process continues.

2. Provide the required information in the **Deploy Project** dialog shown in Figure 7–2.

Figure 7–2 The Deploy Project Dialog



This graphic is described in the text.

****

The properties in the Deploy Project dialog are described in the following table.

Table 7–4 The Deploy Project Dialog Properties

Project	Displays the name of the project
Version	Use to select the version of the project you want to deploy. This can be the current version of the project or a project snapshot.
Last Revision ID	Displays the revision ID of the most recent deployed version of the project.
New Revision ID	Specifies a revision ID for the deployed application. This ID can must be of the form: n0[.n1[.n2[.n3[.n4]]]][-milestone-name[milestone-number]/_ patch-number.
Override the default deployment version	Overrides the default deployment version.
Deployer Username	Used for deploying the Oracle BPM project to run time.
Password	Specifies the password corresponding to the Deployer Username defined above.

Table 7–4 (Cont.) The Deploy Project Dialog Properties

Add MDS Connection for *oramds* Protocol	This option may be available, depending on how your server is configured. Select to enable a connection to a database using the *oramds* protocol. This is sometimes required when connecting to the database used for the Oracle BPM repository. If this is required, your system administrator must provide the following connectivity information for the database: ■ **Hostname**: specifies the database hostname or IP address. ■ **Port**: specifies the database port. ■ **SID**: specifies the SID of the database. ■ **Username**: specifies the database username used to connect to the Oracle MDS database. ■ **Password**: specifies the password for the database user.
Mark composite revision as default	Defines the current revision as the default revision.
Deploy target	Enables you to select the folder in Oracle MDS where the SOA composite application is deployed.

3. Click **Deploy**.

   Business Process Composer deploys the project to runtime. This may take a few minutes.

4. After the deployment is complete, click **OK**.

The project is deployed to Oracle BPM run time. The project is available from the list of deployed projects in the project browser.

7.5.3 How to Deploy a Project Using an Approval Workflow

Business Process Composer enables you to specify an approval workflow. This workflow defines the users who must approve a project before the project is deployed to Oracle BPM run time.

To deploy a project using approval workflow:

1. From the main menu select **Deploy Project**.

   Business Process Composer validates the project. If there are no errors in the project, the project appear in the Approval Workflow browser.

   Based on the type approval workflow defined for the project, required approvers must approve the deployment using Process Workspace. See the *Oracle Fusion Middleware Business Process Management User's Guide* for more information.

   You can monitor the approval status and progress using the Approval Workflow browser.

2. After the deployment has been approved open the Approval Workflow browser, then click **Deploy**.

7.5.4 How to Edit a Deployed Project

You can use Business Process Composer to open deployed Oracle BPM projects. Opening a deployed project enables you to edit the Oracle Business Rules contained in the project and deploy your changes back to Oracle BPM run time.

Note: In order to edit a deployed project, you must be granted the SOA Designer role.

See Section 8.4, "Editing Oracle Business Rules at Run Time" for more information on editing Oracle Business Rules at run time.

To open a deployed project:

1. Launch Business Process Composer.
2. From the main menu, select **Open a Deployed Project**.
3. Select a project from the Project navigator.
4. Click Refresh to ensure you see the latest contents of the Oracle BPM repository.
5. Click **OK**.

7.5.5 How to Generate a Project SAR File

You can generate a project as a SAR file from Business Process Composer. Your system administrator can use this file to deploy a project using the Oracle Enterprise Manager administration console.

To generate a project SAR file:

1. From the main menu, select **Deployment**, then select **Generate Project SAR file**.

   Business Process Composer validates the project. If the project contains errors, these are displayed in the project validation tab.
2. Select the project version you want to use to generate the SAR file. This can be the current version of the project or a project snapshot.
3. Enter a revision ID.
4. If required, select the following options:
   - Override the default deployment version
   - **Add MDS database connection for the oramds protocol**: Select to enable a connection to a database using the oramds protocol. This is sometimes required when connecting to the database used for the Oracle BPM repository.
5. Click **OK**.
6. If the project contains no errors, click OK to save the SAR file to your local file system.

7.5.6 How to Generate a Deployment Plan

A deployment plan is an XML configuration file that is used when deploying a BPM project from Oracle BPM Studio. Business Process Composer will generate any unexpected errors when generating the XML file of the deployment plan.

Note: You should validate your project before creating a deployment plan. Business Process Composer does not perform any validation when generating the deployment plan.

To generate a deployment plan:

1. From the main menu, select **Deployment**.
2. Select **Generate Deployment Plan**
3. Select a location on your local file system, then click **OK**.

8

Using Oracle Business Rules

This chapter describes how to use the Oracle Business Rules editor that is part of Business Process Composer. It contains a general introduction to Oracle Business Rules and provides tasks for working with them.

> **Note:** You cannot create new business rules or rules dictionaries using Business Process Composer. You can edit rules dictionaries that were defined as part of the business catalog of a project template.

This chapter includes the following sections:

8.1 Introduction to Oracle Business Rules

This section provides a brief introduction to Oracle Business Rules.

Business rules are statements that describe business policies or describe key business decisions. For example, business rules include:

- Business policies such as spending policies and approval matrices.
- Constraints such as valid configurations or regulatory requirements.
- Computations such as discounts or premiums.
- Reasoning capabilities such as offers based on customer value.

For example, a car rental company might use the following business rule:

```
IF
Rental_application.driver age < 21
THEN
modify Rental_application(status: "Declined")
```

An airline might use a business rule such as the following:

```
IF
Frequent_Flyer.total_miles > 10000
THEN
```

```
modify Frequent_Flyer (status : "GOLD")
```

A financial institution could use a business rule such as:

```
IF
Application_loan.income < 10000
THEN
modify Application_loan (deny: true)
```

These examples represent individual business rules. In practice, you can use Oracle Business Rules to combine many business rules or to use more complex tests.

Oracle Business Rules allow process analysts to change policies that are expressed as business rules, with little or no assistance from a process developers. Applications using Oracle Business Rules support continuous change that enables the applications to adapt to new government regulations, improvements in internal company processes, or changes in relationships between customers and suppliers.

Business rules follow an if-then structure and consists of two parts:

- If part: a condition or pattern match.
- Then part: a list of actions.

Alternatively, you can express rules in a spreadsheet-like format called a decision table.

8.1.1 Introduction to Rule Conditions

The rule IF part is composed of conditional expressions, rule conditions, that refer to facts. For example:

```
IF Rental_application.driver age < 21
```

The conditional expression compares a business term (Rental_application.driver age) to the number 21 using a less than comparison.

The rule condition activates the rule whenever a combination of facts makes the conditional expression true. In some respects, the rule condition is like a query over the available facts in the Rules Engine, and for every row returned from the query the rule is activated.

8.1.2 Introduction to Rule Actions

The rule THEN part contains the actions that are run when the rule is fired. A rule is fired after it is activated and selected among the other rule activations using conflict resolution mechanisms such as priority. A rule might perform several kinds of actions. An action can add facts, modify facts, or remove facts. An action can run a Java method or perform a function which may modify the status of facts or create facts.

Rules fire sequentially, not in parallel. Note that rule actions often change the set of rule activations and thus change the next rule to fire.

8.1.3 Introduction to Decision Tables

A Decision Table is an alternative business rule format that is more compact and intuitive when many rules are needed to analyze many combinations of property values. You can use a Decision Table to create a set of rules that covers all combinations or where no two combinations conflict.

8.1.4 Introduction to Facts and Bucketsets

In Oracle Business Rules, facts are the objects that rules reason on. Each fact is an instance of a fact type. You must import or create one or more fact types before you can create rules.

In Oracle Business Rules a fact is an asserted instance of a class. The Oracle Business Rules run time or a developer writing in the RL Language uses the RL Language assert function to add an instance of a fact to the Oracle Business Rules Engine.

In Rules Designer you can define a variety of fact types based on, XML Schema, Java classes, Oracle RL definitions, and ADF Business Components view objects. In the Oracle Business Rules run time such fact type instances are called facts.

You can create bucketsets to define a list of values or a range of values of a specified type. After you create a bucketset you can associate the bucketset with a fact property of matching type. Oracle Business Rules uses the bucketsets that you define to specify constraints on the values associated with fact properties in rules or in Decision Tables. You can also use bucketsets to specify constraints for variable initial values and function return values or function argument values.

8.1.5 Introduction to Rulesets

A ruleset is an Oracle Business Rules container for rules and Decision Tables. A ruleset provides a namespace, similar to a Java package, for rules and Decision Tables. In addition you can use rulesets to partially order rule firing.

8.1.6 Introduction to Decision Functions

A decision function provides a contract for invoking rules from Java or SOA (from an SOA composite application or from a BPEL process). The contract includes input fact types, rulesets to run, and output fact types.

8.1.7 Introduction to Decision Points

Oracle Business Rules SDK (Rules SDK) provides APIs that let you write applications that access, create, modify, and run rules in Oracle Business Rules dictionaries (and all the contents of a dictionary). The Rules SDK provides the Decision Point API to access and run rules or Decision Tables from a Java application.

8.1.8 Introduction to Dictionaries

A dictionary is an Oracle Business Rules container for facts, functions, globals, bucketsets, links, decision functions, and rulesets. A dictionary is an XML file that stores the application's rulesets and the data model. Dictionaries can link to other dictionaries. Oracle JDeveloper creates an Oracle Business Rules dictionary in a.rules file. You can create as many dictionaries as you need. A dictionary may contain any number of rulesets.

8.2 Introduction to the Business Process Composer Rules Editor

The Business Process Composer rules editor enables you to view and edit a rules dictionary. Rules dictionaries are displayed in a tabbed window similar to the process editor and data association editor.

This window is divided into two main areas:

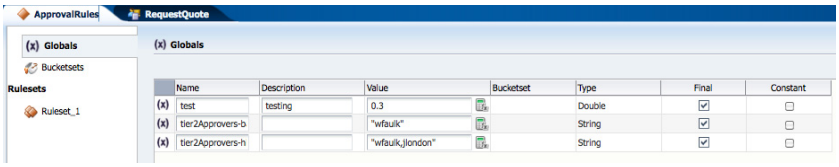
- A panel containing tabbed panes for globals, bucketsets, and rulesets.

- An editor panel showing detailed information for each tab.

Figure 8–1 shows the Globals tab displaying information for the globals contained in the Sales Quote example project.

> **Note:** This tab only displays globals that were marked as final when the rules dictionary was created.

Figure 8–1 The Globals Tab of the Oracle Business Rules Editor



This image displays two tabbed panes labeled Approval Rules and Request Quote. The Approval Rules tab is selected.

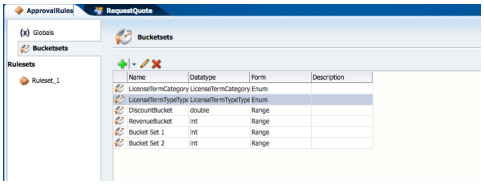
This tab contains three horizontal tabs labeled: Globals, Bucketset, and Ruleset. The Globals tab is selected.

This tab contains a table whose columns are labeled: Name, Description, Value, Bucketset, Type, Final, and Constant.

****

Figure 8–2 shows the Bucketsets tab displaying details for the bucketsets contained in the Sales Quote example project.

Figure 8–2 The Bucketsets Tab of the Oracle Business Rules Editor



This image displays two tabbed panes labeled Approval Rules and Request Quote. The Approval Rules tab is selected.

This tab contains three horizontal tabs labeled: Globals, Bucketset, and Ruleset. The Bucketset tab is selected.

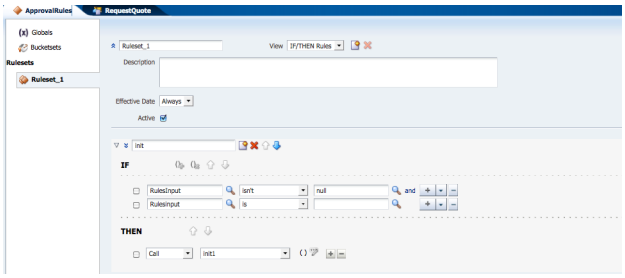
The Bucketset tab contains a table whose columns are labeled: Name, Datatype, Form, and Description.

It also contains a toolbar with icons for Add, Edit, and Delete.

****

Figure 8–3 shows the **Rulesets** tab displaying details for the rulesets contained in the Sales Quote example.

Figure 8–3 The Rulesets Tab of the Oracle Business Rules Editor



This image displays two tabbed panes labeled Approval Rules and Request Quote. The Approval Rules tab is selected.

This tab contains three horizontal tabs labeled: Globals, Bucketset, and Ruleset. The Ruleset tab is selected.

****

8.3 Viewing and Editing Business Rules in Business Process Composer

The following sections provide specific procedures for viewing and editing Oracle Business Rules using Business Process Composer.

8.3.1 How to Open a Business Rule

Oracle Business Rules can be included as part of the reusable business catalog, enabling you to use business rules when editing Oracle BPM projects created from project templates.

To open a business rule from the Project Navigator:

1. From the project home page, expand **Rules**, then click the name of the rule you want to open.

The business rule appears in the rules editor. If you want to edit the rule, ensure that the project is in edit mode.

8.3.2 How to Add a Bucketset

Using Business Process Composer you can add new bucketsets to a rules dictionary.

To add a new bucketset:

1. Open the rules dictionary where you want to edit the bucketset.
2. Select the **Bucketsets** tab. This displays a table listing the bucketsets in the dictionary as shown in Figure 8–2.
3. Click the **Add Bucketset** drop-down list, then select the type of bucket set you want to create.
   - List of values
   - List of ranges
4. Select the bucketset from the list, then click **Edit Bucketset**.
5. Edit the bucket set as required, then click **OK**.

8.3.3 How to Edit an Existing Bucketset

In Business Process Composer, selecting the **Bucketsets** tab shows you a table listing the bucketsets in the dictionary. To edit a bucketset, select the appropriate row and click the **Edit** icon. Depending on the type of the bucketset, Range, Enum, or LOV, this displays a corresponding Edit bucketset page.

You can create a Range Bucketset by clicking Add in the menu bar and selecting a type. This adds a new row in the Bucketsets table. Adding a bucket automatically adds an end point for a range bucket and a value for an LOV bucket based on the data type. You can modify the newly added bucket end point or value. Note that the alias is modified when an end point or value is changed.

To delete a bucketset, select a row and click **Delete**.

To edit a bucketset:

1. Open the rules dictionary where you want to edit the bucketset.
2. Select the Bucketsets tab. This displays a table listing the bucketsets in the dictionary as shown in Figure 8–2.
3. Select the appropriate bucketset row and click the Edit Bucketset icon.
4. Use the Bucketset Editor to edit the appropriate fields in the bucketset.
5. Click OK to confirm the changes.

8.3.4 How to View Globals in the Oracle Rules Dictionary

When you open a rules dictionary, Business Process Composer displays the Globals tab. The Globals tab only shows final global variables (global variables with Final option selected).

You cannot create or delete global variables. From the Globals tab, in edit mode you can edit the Name, Description, and Value fields. For the Value field, you can use the expression builder to set the value.

To view globals:

1. Open the rules dictionary where you want to view globals.
2. Select the Globals tab. This displays a table listing the globals defined for this rules dictionary as shown in Figure 8–1.

8.3.5 How to Add a Rule to a Ruleset

Using Business Process Composer you can edit, add, and delete rules in a ruleset.

To add a rule to a ruleset:

1. Open the rules dictionary containing the ruleset where you want to add a rule.
2. Click the tab of the ruleset you want to edit. This displays a table listing the rulesets defined for this dictionary as shown in Figure 8–3
3. Click the **New Rule** icon.
4. Enter a name for the new rule.
5. Click the **Show Advanced Settings** icon.

6. In the IF area, use the controls, icons, and selection boxes, including the Left Value expression icon, drop-down list for an operator, and Right Value expression icon to modify the condition.
7. In the THEN area for the rule, next to the rule action click Add Action.
8. Click Save in the editor toolbar to save the changes to the rule dictionary.

8.4 Editing Oracle Business Rules at Run Time

You can use Business Process Composer to open deployed Oracle BPM projects. Opening a deployed project enables you to edit the Oracle Business Rules contained in the project and deploy your changes back to Oracle BPM run time.

> **Note:** When editing a deployed project, you can only edit the Oracle Business Rules for that project. You can view other project resources, but cannot edit them.

To open a deployed project:

1. From the **Project** menu select **Open a Deployed Project**.

   If you are currently editing a project, your changes are automatically saved.
2. Select **Deployed Projects**, then select the project you want to open from the project list.
3. Expand Repository, then select the deployed project you want to open.
4. Click **Ok**.
5. In the Project Navigator, expand **Business Rules** then expand the rules dictionary where whose Oracle Business Rules you want to edit.
6. Click **Edit** in the rules editor.
7. **Edit** the rules as required, then click **Save**.
8. Click **Validate** to ensure the changes you made to the business s made are valid.
9. Click **Commit** to commit the changes to Oracle BPM run time.
10. Click **Yes**.
11. From the **Project** menu, select **Close Project**.

8.5 Assigning a Rule to a Business Rules Task

The business rules task is an Oracle BPMN element that enables you to incorporate Oracle Business Rules within a process model.

When editing a project based on a project template containing business rules in the business catalog, you can assign business rules to business rules task.

To assign a business rule to a business rules task:

1. Open the process containing the business rules task you want to edit.
2. Right-click the business rules task, then select **Properties**.
3. Select the Implementation tab.

4. Click **Change**. The displays the business rules browser containing a table of available business rules.
5. Double-click a rule from the table.
6. Click **OK**.

Part III

Advanced Business Process Composer Functionality

This part describes advanced functionality of Business Process Composer that is targeted towards process developers who need to make changes to the implementation details of a BPM project.

This part contains the following chapters:

9

Advanced Business Process Composer Functionality

This chapter describes functionality of Business Process Composer related to technical implementation of a BPMN process. The functionality described in this chapter is generally performed by a process developer.

This chapter includes the following sections:

9.1 Working with Services

The following sections describe how to create services using Business Process Composer.

9.1.1 How to Create New Services in the Business Catalog

Business Process Composer enables you to create new services in the business catalog. These services are based on standard Web Services.

Services based on web services are defined using a Web Services Description Language (WSDL) file. A WSDL file is an XML file used to describe how web services are implemented. When creating a new service using Business Process Composer, you can specify a WSDL file stored local on your machine or one that is available.

Process developers are responsible for providing a WSDL file or URL location.

To create a new service:

1. Open the project where you want to create a new service.
2. From the main menu, select **New** then **New Service**.
3. Provide the following information:
   - **Name**: Defines the name of the service as it appears in the business catalog.
   - **Type**: Defines the type of service being created.
   - **WSDL**: The source of the WSDL used to create the new service can be one of the following:
     - **URL**: Specifies the remote URL of the WSDL.
     - **File**: Specifies either a WSDL or ZIP file on your local file system.

If using a ZIP file, it can include only WSDL and XSD files. The WSDL file should have valid references.

- **Port Type**: Port type specifies the service used. A WSDL file exposes one or more port types.
- **Callback Type**: Determines the call back type used for this web service. This is only applicable to asynchronous services.
- **Transaction Participation**:
- **Version**:

4. Click **OK**.

9.2 Defining Conversations

The following sections describe how to edit conversations using Business Process Composer. For complete information see *Oracle BPM Modeling and Implementation Guide*.

9.2.1 Introduction to Conversations

Conversations group the message exchange between two or more processes. The message exchange between processes is called collaboration. Within a project you can define multiple conversations that you can reuse amongst the processes in that project.

Collaboration diagrams allows you to view the process flow together with the interactions your process has with other participants in the conversation.

Your BPM project defines a conversation by default. If you do not want to define multiple conversations you must use this default conversation to gather all the message exchange amongst the processes in your project.

You can only define one default conversation per project. However you can modify your project to use a different default conversation than the one it uses by default.

The different types of conversations allow you to specify the different types of interaction your process can establish with other processes or services. The following list describes the different types of conversations:

- **Define Interface**: use this type to define the operations that other services and processes can invoke to interact with a BPMN process.
- **Use Interface:** use this type to configure your process to use an interface from a component in the Business Catalog.
- **Process Call:** use this type to invoke another BPMN process.
- **Service Call**: use this type to invoke a service defined in your BPM project.

9.2.2 Working with Conversations

The following sections describe how to define and configure conversations using Business Process Composer.

9.2.2.1 How to define a conversation

1. Open your process.
2. Click the **Edit Conversation** button in the process editor toolbar.
3. Click the Add Conversation button.

4. Provide a name for the conversation.
5. Select the type of conversation you want to define.
6. Click **OK**.

9.2.2.2 How to set the default conversation

1. Open your process.
2. Click the **Edit Conversation** button in the process editor toolbar.
3. Select a conversation from the list, then click the **Edit** button.
4. Click the **Default Conversation** checkbox.
5. Click **OK**.

9.2.2.3 How to define a conversation for a BPMN flow object

1. Open your process
2. Right-click one of the following types of BPMN flow objects:
   - Message events (throw and catch)
   - Send and receive tasks
   - Service tasks

   You can only define conversations these BPM flow objects.
3. Select **Implement**.
4. In the **Implementation** tab, click the **Browse** button next to the **Conversation** text field.
5. Select a conversation from the list, then click **OK**.

9.2.2.4 How to view a collaboration diagram

To view the collaboration diagram for a process, click the **View Collaboration** button in the process editor toolbar.

Note: In collaboration view, you cannot make changes to the process. To return to normal process editing, click the **View Collaboration** button again.

10

Working with Data Objects and Expressions

This chapter describes how to use data objects and expressions within Oracle Business Process Composer. Data objects and expressions enable you to define the data used within your process and determine how it is handled.

This chapter includes the following sections:

10.1 Introduction to Data Objects

Data Objects are variables used to define the type of information used by your business process. They are also used to store the value of this information.

Oracle BPM supports two types of data objects:

- **Basic data objects**

  Basic data objects define the basic types of variables that you can use within your processes and projects. Basic data objects can be used explicitly within your process or they can be combined into complex data objects.

  Table 10–1 lists the types of basic data objects supported by Oracle BPM.

Table 10–1 Simple Data Objects

Type	Description
Bool	Represents the logical values true or false.
Int	Represents an integer. For example: 23, -10, 0
Int(64)	Represents a long integer value.
Decimal	Represents a number than can be expressed in decimal notation. For example: 3.14, 62, 0.023.
Real	Represents a floating-point numeric value. For example: 2e-1, 2.3E8.

Table 10–1 (Cont.) Simple Data Objects

Type	Description
String	Represents a sequence of characters. For example: "This is a string."
Time	Represents a specific time expressed as: year-month day hour:minute:second. For example: 1995-02-03 13:30:28-08:00
Interval	Represents a duration of time expressed as a number in years, months, days, hours, minutes, and seconds. For example: 1d3h30m.
Binary	Used to store binary data, including images or videos.

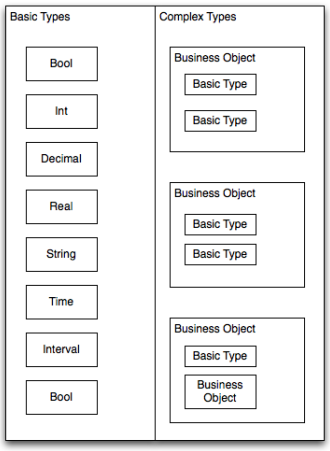
- **Complex data objects**

  Complex data objects enable you to group types of data. Complex data types are defined using business objects.

  Business objects enable you to create data structures based on basic data objects. For example, you can create a complex data object called employee that contains different data types for employee name, id, and salary.

  Figure 10–1 shows the relationship between basic data objects, complex data objects, and business objects.

Figure 10–1 Relationship Between Basic and Complex Data Objects



This graphic shows a rectangle divided into two sections. The left-hand section is labeled Basic Types. The right-hand section is labeled Complex Types.

The left-hand section contains eight rectangles with a label for each of the basic types described in the previous section.

The right-hand section contains three boxes labeled Business Object. Each of these boxes contain rectangles labeled Basic Type or Business Object.

****

  Business Process Composer enables you to create or edit business objects. Additionally, you can create new complex based on business objects defined in the business catalog. However, you cannot create new data types.

10.1.1 Introduction to Process and Project Data Objects

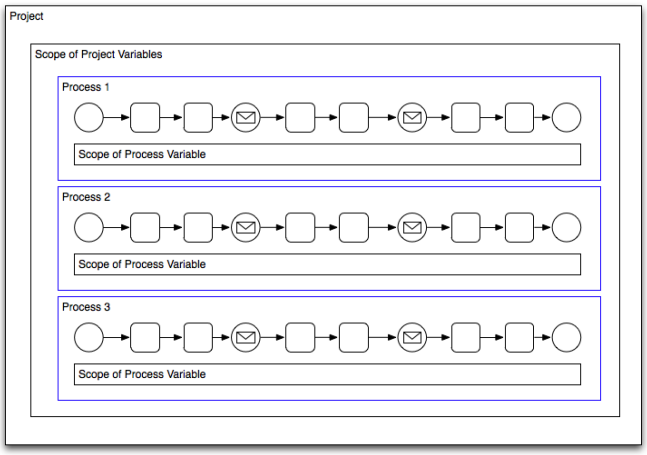
Process data objects are data objects that are defined for a specific process. Similarly, project data objects are defined for an entire project.

Process data objects can be created from both basic and complex data objects. Project data objects can only be created from basic data types. This determines the scope of the variable.

Therefore, you can only use process data objects within the process where they are created, while project data objects are applicable to the entire project. This is known as variable scope.

Figure 10–2 shows the difference in scope between project and process variables.

Figure 10–2 Scope of Process and Project Data Objects



This figure shows the scope of project and process data objects.

****

Process data objects allow you to define data objects that are only used within a single process. When designing a process-based application, if you know that a data object is only used within a process, it is better to define it as a process data object to conserve system resources.

Project data objects allow you to share data between processes. For example, the Purchase Order process and the Request Approval process may both track the value of the employee that created the request, or the priority of the request.

Project data objects ensure that all processes in a project use the same data. Each process must assign and update the value of this data.

The main benefit of defining project data objects is that after publishing your project you can configure Oracle Business Process Management Workspace views to display the values of those variables. This is possible only if you use project data objects.

> **Note:** Although project data objects allow you to define data objects that are used by all processes in a project, they are not global data objects. Each process within your project uses its own version of the data object. Project data objects are not used to share data between processes.

10.1.2 Using Data Objects in New BPM Projects

When working with new projects created in Business Process Composer, you can create project and process data objects only for flow objects that support data associations. New projects created in Business Process Composer enable process analysts to define types of data objects used between process. These data objects can be incorporated by process developers as part of the implementation created in Oracle BPM Studio.

10.1.3 Using Data Objects in Projects Based on Project Templates

Project templates allow you to create reusable components from the business catalog. In addition to services such as business rules and human tasks, you can also create business objects.

When creating an Oracle BPM project based on a project template, you can use the data objects defined in the business catalog within data associations. Also, you can create new project data objects using Business Process Composer.

Project variables can only be simple data objects based on created from basic data types.

Note: You cannot create new types of complex data objects in Business Process Composer.

10.1.4 Introduction to Data Associations

Data associations are used to pass the information stored in data objects in the following contexts:

- To and from another process or service invoked from a BPMN process
- To and from a Human Task service
- To and from an Oracle Business Rule
- To and from a script task. This BPMN flow object is used to pass data objects through data associations.

Table Figure 10–2 lists the flow objects where you can define data associations. It also lists the objects implemented.

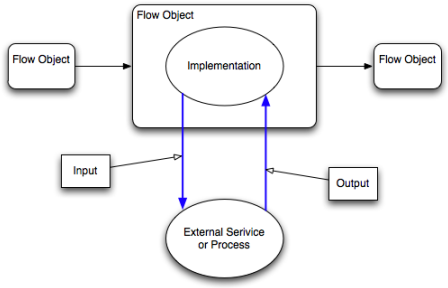
Table 10–2 Flow Objects that Accept Data Associations

Flow Objects	Implementation
Message start and end events	Services and other BPMN processes
Message throw and catch events	Services and other BPMN processes
Send and receive tasks	Services and other BPMN processes
Script tasks	Do not contain an implementation, are used to pass data objects through data associations.
User tasks	Oracle Human Tasks
Business rule tasks	Oracle Business Rules
Service Tasks	Services and BPMN processes

Data associations are used to define the input and output from a flow object to an external service or process. Figure 10–3 shows the relationship between a flow object, its corresponding implementation, and external processes or services.

The blue arrows represent the input and output arguments to the external process or service. These are defined using data associations.

Figure 10–3 Relationship between a Flow Object, Implementation and an External Service or Process

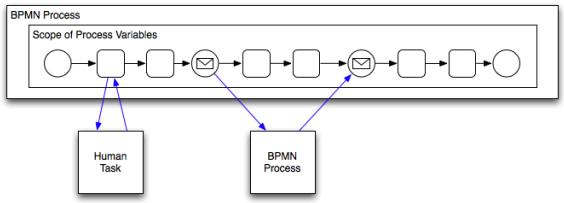


This figure shows the relationship between a flow object and its implementation.

****

It is important to note that although the inputs and outputs are defined in the data associations for a flow object, the defined values that are passed to the implemented systems and services. These systems and services are external to the process as shown in Figure 10–5.

Figure 10–4 Data Associations between a process



This graphic contains a long horizontal rectangle labeled BPMN process. Within the rectangle is a generic representation of a BPMN process containing squares and circles representing flow objects

Below the long rectangle are two squares labeled Human Task and BPMN process. These are connected to the BPMN process by blue arrows representing the data associations between the BPMN process and the implementation represented by the two squares.

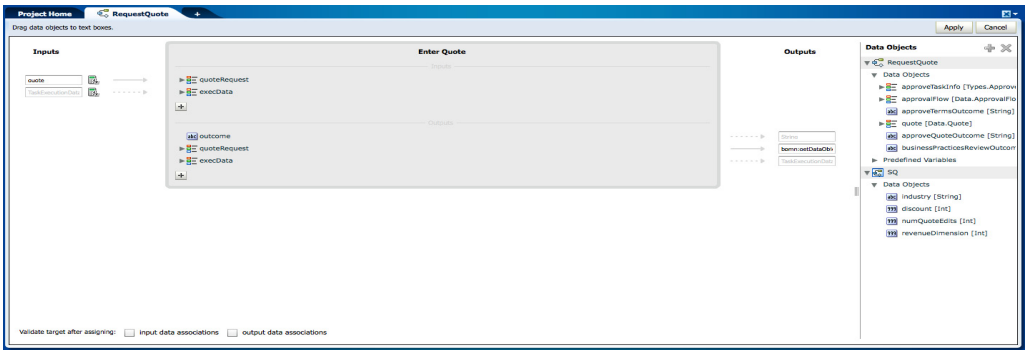
****

You can use expression to evaluate and change the input and output values

10.1.5 Introduction to the Data Associations Editor

The data associations editor enables you to configure the input and output values passed between a flow object and a its implementation.

Figure 10–5 The Data Associations Editor



This graphic is described in the text.

****

Table 10–3 describes the different areas of the data associations editor.

Table 10–3 The Data Associations Editor User Interface

UI Area	Description
Inputs	Contains text boxes that display the data objects assigned as inputs to the service or process implemented in the flow object. Next to each text box is an icon that launches the expression editor
Flow Object Interface	Lists the expected input arguments for the service or process implemented. The flow object interface also contains an expandable list of the data objects supplied as input and output. Within the flow object area, you can expand complex data objects to map to specific basic data objects within a complex data object.
Outputs	Contains text boxes that display the data objects assigned as outputs from the service or process implemented in the flow object.
Data Objects	Displays a list of all the data objects. This list is divided between process and project data objects.

10.2 Working with Data Objects and Data Associations

The following sections describe how to create and delete data associations and configure data associations.

10.2.1 How to Create a Data Object

Using Business Process Composer, you can create process and project data objects.

To create a data object:

1. Ensure that you are editing your project.
2. Open your process.
3. Right-click on a flow object, then select **Data Associations**.

   For a list of flow objects that allow you to configure data associations, see Table 10–2.
4. Select a processes or the project from the expandable list.

5. Click **Add New Data Object**.
6. Enter a name for your data object, then select a type from the drop-down menu.
7. Click **Create**

10.2.2 How to Delete a Data Object

You can delete a data object from a process or project.

To delete a data object:

1. Open the process where you want to delete a data object. If you want to delete a project data object, you can open any process in the project.
2. Right-click a flow object that supports data associations, then select **Data Associations**.

   See Table 10–2 for the list of sequence flows that allow data associations.
3. In the **Data Objects** column, expand the process or project containing the data object you want to delete, then select the data object.
4. Click the **Delete** icon

> **Note:** You can delete simple and complex data objects. However, you cannot delete simple data objects used within complex data objects.

10.2.2.1 What You Need to Know About Deleting Data Objects

After deleting a data object, you must ensure that all references to it are removed. This includes any data associations and expressions that use the data object. If you do not remove references to the deleted data object, the project does not validate.

10.2.3 How to Configure Data Associations for a Flow Object

You can configure data associatons for flow objects.

To configure a data association for a flow object:

1. Open the process where you want to configure data associations.
2. Right-click a flow object that enables data associations, then select **Data Associations**.

   See Table 10–2 for the list of sequence flows that allow data associations.
3. From the data objects column on the right, select the data object you want to map as an input argument.
4. Click and drag the data object to an input text field.

10.3 Working with Business Indicators and Counter Marks

This section describes how to create business indicators and counter marks using Business Process Composer.

10.3.1 Introduction to Business Indicators and Counters

Business Indicators are project data objects you use to store the value of the key performance indicators of your process. Although Oracle BPM allows you to create

business indicators using different types of data objects, within Business Process Composer you can only create business indicators used as counter marks.

Counters keep track of the number of times an instance completes a certain activity. You must use them with counter marks. The counter variable does not store the actual value, its value is always 1. The value that specifies the number of times an instance completes an activity is updated directly in the Process Analytics databases. To monitor the value of a counter business indicator, you must create a dashboard based on a counter mark that is configured to track this counter business indicator.

10.3.2 Introduction to Counter Marks

Counter marks enable you to update the value of the counter business indicators defined for your process. A counter mark may update multiple counter mark business indicators. When a token arrives at an activity that has a counter mark defined, the BPM Service Engine updates the value of its associated counters in the Process Analytics databases. Each time the BPM Service Engine updates a counter business indicator, it adds one unit to the current value.

Note: The actual value of the counter variable is stored in the Process Analytics databases. You must not use the counter variable in your process to perform any calculations because its default value never changes. The value of the counter variable is always equal to 1.

You can use counter marks for the following:

- Auditing: The number of activities the instance completed combined with other performance measurements are important information for auditing the process.
- Identifying performance issues: You can use a counter to identify performance issues within your process. Your process might be taking longer than expected because the instances are following a different path than expected or because the loop in an activity is running more times than it should. You can identify these situations by comparing the actual number of completed activities to the number you expected.
- Identifying the process path the instance followed: You can mark different paths using different counter business indicators. When the instance reaches the end of the process, the path the instance followed has the greatest number of completed activities.

Typically you define one counter business indicator for each of the process paths you want to monitor. Then you add counter marks in all the activities that are part of that process path. Finally you associate the counter business indicators that correspond to the paths that activity is part of, to the counter mark.

10.3.3 How to Add a New Counter Mark to a Process

You can add new counter marks to activities and tasks within your process.

To add a new counter mark to a process

1. Right-click on the task or activity where you want to add a counter mark.
2. Select **Implement**.
3. If required, create a new business indicator.
    1. Click the **Add** button.

2. Provide a name for the business indicator.
3. Click **OK**.

4. In the list of business indicators, select the check box next to the indicators you want to use for this flow object.
5. Click **Apply Changes**.

10.3.4 How to Delete a Counter Mark

You can delete the counter marks you have defined for a project.

To delete a counter mark:

1. Right-click on the activity or task where you want to edit a counter mark.
2. Select **Data Associations**.
3. In the list of Data Objects, expand the name of the project. This contains a list of all the project data object.
4. Select the counter mark you want to delete.
5. Click **Remove Data Object**.
6. Click **Yes**.
7. Click **Apply**.

10.4 Introduction to Expressions

Expressions allow you to perform calculations on data objects.Using Business Process Composer, you can define and edit expressions in the following contexts:

- Conditional sequence flows
- Complex gateways
- Timer events
- Data associations
- Notification tasks

Expressions do not allow you to directly reassign the values to data objects. However, you can use expressions to change the values passed to and from the implementation of a sequence flow. See Section 10.1.4, "Introduction to Data Associations" for more information.

10.4.1 Types of Expressions

Oracle Business Process Composer supports the following types of expressions:

- Simple
- Plain text
- XML Literal

10.4.2 Simple Expressions

Simple expressions are defined using a basic expression language supported by Oracle BPM.

10.4.2.1 Operator Types

Simple expressions support the following operator types:

- Arithmetic Operators
- Unary Operators
- Equality and Relational Operators
- Conditional Operators

You can use these operators to write expressions and conditions to define your process flow. Generally these expressions perform their calculations based on the data objects in your process. You can write expressions and conditions using the value of the data objects, but you cannot modify their value.

The following examples of expressions use operators:

- totalAmount - discount
- activationCount > 3
- unitsSold <= 1200

Table 10–4, Table 10–5, Table 10–6, and Table 10–7 describe the supported operators in the simple expression builder.

Table 10–4 Arithmetic Operators

Operator	Name	Description
+	Addition	Adds numeric data types. Concatenates Strings.
-	Subtraction	Subtracts numeric data types.
*	Multiplication	Multiplies numeric data types.
/	Division	Divides numeric data types.
rem	Remainder	Calculates the remainder of a division in which the divisor does not exactly divide the dividend.
( )	Precedence	Indicates the order of evaluation of an arithmetic expression.

Table 10–5 Unary Operators

Operator	Name	Description
+	Plus	Has no effect in the value of the numeric operand. Use it to indicate explicitly that a certain value is positive.
-	Minus	Negates an arithmetic expression.
*	Not	Logical complement operator. Negates the value of a boolean expression.

Table 10–6 Equality and Relational Operators

Operator	Name	Description
= or ==	Equal	Returns true is the first operand equals the second operand.
!=	Not Equal	Returns true is the first operand is not equal to the second operand.
>	Greater Than	Returns true if the first operand is greater than the second operand.

Table 10–6 (Cont.) Equality and Relational Operators

Operator	Name	Description
>=	Greater Than or Equal to	Returns true if the first operand is greater than or equal to the second operand.
<	Less Than	Returns true if the first operand is less than the second operand.
<=	Less Than or Equal to	Returns true if the first operand is less than or equal to the second operand.

Table 10–7 Conditional Operators

Operator	Name	Description
and	Conditional And	Returns true if both operands evaluate to true.
or	Conditional Or	Returns true if either operand evaluates to true.

10.4.2.2 Operator Precedence

Operator precedence indicates the order in which the compiler evaluates them. You can change operator precedence in an expression by using parenthesis.

In Oracle BPM the operator precedence is:

- Addition, Subtraction
- Multiplication, Division, Remainder
- Plus and Minus
- Less than, Greater Than, Less Than or Equal to, Greater Than or Equal to
- Equal, Not Equal
- Not
- Conditional And
- Conditional Or

10.5 Defining Process Input and Output

When you add operations to a BPMN process, you are defining points in the process that other processes or services can use to communicate with it. The communication between processes and other processes or services generally requires an input and returns an output. The flow events that you use you to define the BPMN process operations enable you to define input and output arguments. These input and output arguments define the process input and output.

10.5.1 How to Define the Input Arguments for a Process

When you create a process that begins with a message start event, you must define the input arguments to that are passed to that process.

To define the input arguments for a process:

1. Add a message start event to your process.
2. Right-click on the message start event, the select **Properties**.
3. Click the **Implementation** tab.

4. Select **Define Interface**.
5. Click the **Add** icon.
6. Determine the type of data object.
7. Click **OK**.

10.5.2 How to Define Data Associations for a Message Start Event

After you have defined the input arguments to your process, you must map them to data objects in your process.

To define data associations for a message start event:

1. Select the message start event of your process.
2. Click **Data Associations**.
3. Drag the data objects from the list on the right-hand side to the text boxes listed under **Outputs**.
4. Click **Apply**.

10.5.3 How to Define the Output Arguments for a Process

When you create a process that contains message end events, you must define the output arguments for each end event. These are the output arguments for the process.

To define the output arguments for a process:

1. Add a message end event to your process.
2. Right-click on the message end event, then select **Properties**.
3. Click the **Implementation** tab.
4. Select **Define Interface**.
5. Click the **Add** icon.
6. Determine the type of data object.
7. Click **OK**.

10.5.4 How to Define Data Association for a Message End Event

After you have defined the output arguments for your process, you must map them to the data objects in your process.

To define data associations for a message end event:

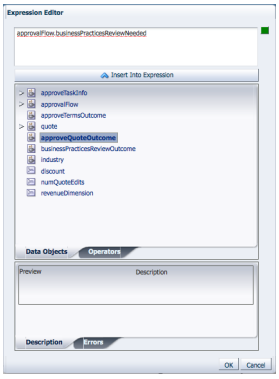
1. Select a message end event in your process.
2. Click **Data Associations**.
3. Drag the data objects from the list on the right-hand side to the text boxes listed under **Inputs**.
4. Click **Apply**.

10.6 Introduction to the Expression Editor

The expression editor provides a simple way of creating expressions by allowing you to select data objects and operators from a list and insert them into your expression. You can also enter the expression manually if necessary.

Figure 10–6 shows the expression editor user interface.

Figure 10–6 The Oracle Business Process Composer Expression Editor



This graphic displays the expression editor. It is divided into three main areas.

At the top is a text field that contains the expression. Below the text field is a button labeled Insert Into Expression.

The center section contains two tabbed panes labeled Data Objects and Operators. The Data Objects tab is selected. This tab displays all the data objects defined for the current project.

The bottom section contains a table that displays the name and description of the currently selected object in the center section.

****

Table 10–8 The Expression Editor User Interface

Area	Description
Expression field	Contains the text of the expression. You can edit this field directly, or use the Insert Into Expression tool.
Insert Into Expression	Inserts the selected data object or operator into the expression.
Data Object and Operator Chooser	Contain tabbed panes that allow you to select the data object or operator you want to insert into the expression.
Description tab	Provides a description of the selected operator.
Errors	Displays errors in the current expression.

10.7 Working with Expressions

The following sections describe how to define expressions using Business Process Composer.

10.7.1 How to Define a Simple Expression for a Conditional Sequence Flow

Using Business Process Composer, you can create and edit expressions for conditional sequence flows. Conditional sequence use expression to determine the flow of your process.

To define an expression for a conditional sequence flow:

1. Open your process.
2. Ensure that the project is in edit mode.
3. Click the edit icon for the conditional sequence flow you want to edit.
4. Click **Implementation**.
5. Click **Edit**.

   The expression editor window displays.
6. Add any required data objects and operators.

   To add a data object to an expression:

   a. Select the **Data Objects** tab.

   b. Select a data object from the list.

      If you add a basic data object that is part of a complex data objects, expand the complex data object and select the basic data object.

   c. Click **Insert Into Expression**

   To add an operator to an expression:

   a. Select the **Operators** tab.

   b. From the expandable list, select the operator you want to add.

   c. Click **Insert Into Expression**.
7. Click the **Error** tab, then verify that there are no errors in your expression.
8. Click **OK**.
9. Click **Apply Changes** in the **Implementation** tab.

10.7.2 How to Define a Simple Expression in Data Associations

Using Business Process Composer, you can create and edit expressions for Data associations. Data associations use expression to alter the values data objects passed as inputs. and outputs.

To define an expression within a data association input or output:

1. Open your process.
2. Ensure that the project is in edit mode.
3. Right-click a flow object within your process then select **Data Associations**.
4. Click **Launch Expression Builder**.
5. Add any required data objects and operators.

   To add a data object to an expression:

   a. Select the **Data Objects** tab.

   b. Select a data object from the list.

If you add a basic data object that is part of a complex data objects, expand the complex data object and select the basic data object.

c. Click **Insert Into Expression**

To add an operator to an expression:

a. Select the **Operators** tab.

b. From the expandable list, select the operator you want to add.

c. Click **Insert Into Expression**.

6. Click the **Error** tab, then verify that there are no errors in your expression.
7. Click **OK**.

11

Working with Human Tasks

This chapter describes how to create and edit human tasks using Oracle Business Process Composer.

This chapter includes the following sections:

11.1 Understanding Human Tasks

Human Workflow is a component of the Oracle SOA and BPM suites that allow you to define how users interact with your business applications. Human Workflow also provides a runtime environment for managing this interaction.

Human tasks are the specific component of Oracle Human Workflow that allows you to define the user interaction of your application. Oracle Business Process Composer enables you to create and edit human tasks.

> **Note:** Not all properties of human tasks can be created or edited using Oracle Business Process Composer. To create and edit these properties, you must use Oracle JDeveloper. See the *Oracle BPM Modeling and Implementation Guide* for more information.

Human tasks enable you to specify the following:

- Routing
- Assignment
- Deadlines
- Notification
- Presentations
- Task data

These components are described in the following sections.

11.1.1 Introduction to Routing and Participants

Human tasks enable you to determine the order in which users perform different tasks within your application. This order is the routing of the human task. Participants are

the users or groups that are responsible for performing each task. You can use the human task editor specify the routing flow and participant for a user task.

11.1.1.1 Participant Types

Human tasks support the following patterns for common routing scenarios:

- Single approver

  This is the simple case where a participant maps to a user, group, or role. See Section 11.1.2, "Introduction to Participant Assignment" for more information.

  For example, a vacation request is assigned to a manager. The manager must act on the request task three days before the vacation starts. If the manager formally approves or rejects the request, the employee is notified with the decision. If the manager does not act on the task, the request is treated as rejected. Notification actions similar to the formal rejection are taken.

- Parallel

  This participant indicates that a set of people must work in parallel. This pattern is commonly used for voting.

  For example, multiple users in a hiring situation must vote to hire or reject an applicant. You specify the voting percentage that is needed for the outcome to take effect, such as a majority vote or a unanimous vote.

- FYI

  This participant also maps to a single user, group, or role, just as in single approver. However, this pattern indicates that the participant just receives a notification task and the business process does not wait for the participant's response. FYI participants cannot directly impact the outcome of a task, but in some cases can provide comments or add attachments.

  For example, a regional sales office is notified that a candidate for employment has been approved for hire by the regional manager and their candidacy is being passed onto the state wide manager for approval or rejection.

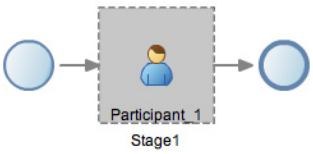
- Serial

  This participant indicates that a set of users must work in sequence. While working in sequence can be specified in the routing policy by using multiple participants in sequence, this pattern is useful when the set of people is dynamic. The most common scenario for this is management chain escalation, which is done by specifying that the list is based on a management chain within the specification of this pattern.

11.1.1.2 Routing Types

Figure 11–1 show a basic routing that contains only one participant.

Figure 11–1 Basic Routing with a Single Participant



This figure shows a basic routing with a single participant. On the far left is a circle that indicates the beginning of the flow.

In the center is a grey box containing an icon in the form of a person. It is labelled Participant 1.

On the right is another circle that indicates the end of the flow.

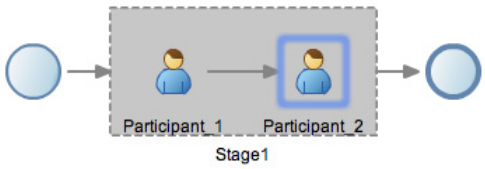
****

However, human tasks support more complex types of routing that give you more control. These routing types are:

- Sequential

  In sequential routing different participants act on a task sequentially. Figure 11–2 shows an example of sequential routing. In this example, Participant_1 is the first participant in the routing flow. After Participant_1 finishes acting on a task, the task is passed to Particpant_2. After Participant_2 finishes acting on the task, the human task is completed.

Figure 11–2 Example of Sequential Task Routing



This graphic displays a sequential task routing with two participants. On the far left is a circle that indicates the beginning of the flow.

In the center is a grey box containing two icons in the form of a person. The one on the left is labelled Participant 1. The one on the right is labeled Participant 2.

On the right is another circle that represents the end of the flow.

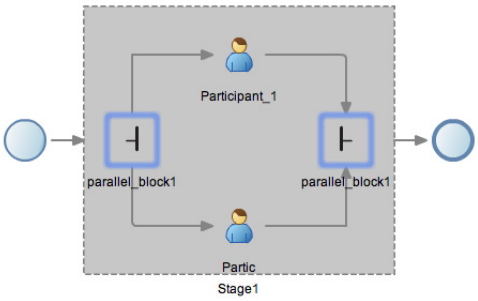
****

- Parallel

  In parallel routing, participants act on a task simultaneously. Figure 11–3 shows an example of parallel routing. In this example, both Participant_1 and Participant_2 act on the task at the same time.

  Parallel routing allows you to define an outcome for the routing that determines how the task flow continues to the next participant after the parallel block. See Section 11.1.1.3, "Outcome" for more information.

Figure 11–3 Example of Parallel Task Routing



This graphic displays a parallel task routing with two participants. On the far left is a circle that indicates the beginning of the flow.

In the center is a grey box containing two icons in the form of a person. The one at the top is labelled Participant 1. The bottom one is labeled Participant 2.

On the right is another circle that represents the end of the flow.

****

11.1.1.3 Outcome

Outcome specifies possible outcome arguments of the Human Task. Oracle BPM Worklist displays the possible outcomes you select as the available tasks to perform at run time.

You can specify a voted-upon outcome that overrides the default outcomes selected in the Default Outcomes list. This outcome takes effect if the required percentage is reached. Outcomes are evaluated in the order listed in the table as shown in.

See Section 11.3.4, "How to Configure the Outcome for Parallel Routing" for information on how to configure outcome.

11.1.2 Introduction to Participant Assignment

Participant assignment is the process of mapping human task participants to the people in your organization that will use your application. This is done by mapping each participant to one of the following:

- Users

  You can assign individual users to act upon tasks. For example, you may assign users jlondon or jstein to a particular task. Users are defined in an identity store configured with the SOA Infrastructure. These users can be in the embedded LDAP of Oracle WebLogic Server, Oracle Internet Directory, or a third party LDAP directory.

  As with users, groups are defined in the identity store of the SOA Infrastructure.

- Groups

  You can assign groups to act upon tasks. Groups contain individual users who can claim and act upon a task. For example, users jcooper and fkafka may be members of the group LoanAgentGroup that you assign to act upon the task.

- Application Roles

You can assign users who are members of application roles to claim and act upon tasks.

See Section 11.3.5, "How to Assign Users, Groups, or Roles to a Participant" for procedures on assigning users, groups, or roles to a human task participant.

11.1.3 Introduction to Duration

Duration defines how long a human task can remain idle before some other action is performed. Duration can be defined globally for the human task or for each human task participant.

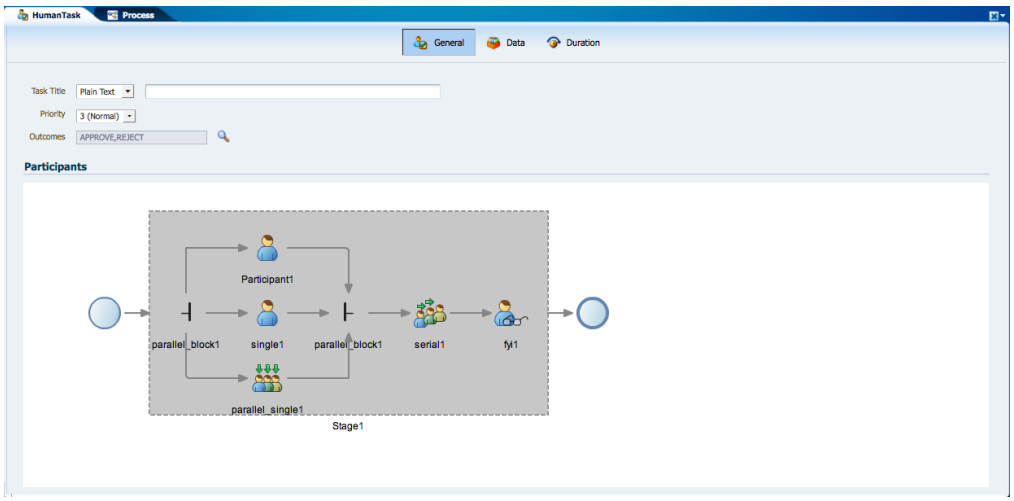
See Section 11.3.7, "How to Define the Duration for a Human Task" for information on defining duration globally for a human task.

See Section 11.3.6, "How to Define the Duration for a Participant" for information on defining duration for a human task participant.

11.2 Introduction to the Human Task Editor

Oracle Business Process Composer provides a human task editor that enables you to create and edit human tasks. The human task editor consists of a tabbed pane as shown in Figure 11–4.

Figure 11–4 The Human Task Editor



This graphic displays a tabbed pane containing the human task editor. At the top of the editor are three tabs labeled General, Data, and Duration.

Directly below the tabs are three fields labeled: Task title, Priority, and Outcomes.

Below the fields is an area of the editor labeled Participants. The participant editor displays multiple participants and routing flows.

****

Within the tabbed pane, there are three tabs that enable you to configure different properties of a human task:

- General: Enables you to edit basic properties of a human task. It also allows you to define the participants and routing.

- Data: Enables you to configure task data.
- Duration: Enables you to configure the duration globally for the human task.

11.3 Working with Human Tasks

The following sections describe how to create and edit human tasks.

11.3.1 How to Create New Human Task

Oracle Business Process Composer enables you to create new human tasks. These are stored in the business catalog and can be incorporated within your business processes.

To create a new human task:

1. Open the project where you want to create a new human task.
2. Ensure the project is in edit mode.
3. From the project welcome page, select **Human Tasks**.
4. Click **New Human Task**.
5. Provide a name for the new human task, then click **Create**.

The new human task is displayed in list of human tasks.

Human tasks are available in the business catalog. You can incorporate it within a BPMN process by assuaging it to a user task. See Section 6.3.2, "Introduction to the User Task" for more information.

11.3.2 How to Open a Human Task

You can open an existing human task for viewing or editing.

To open an existing human task:

1. Open the project containing the human task you want to view or edit.
2. Ensure the project is in edit mode.
3. From the project welcome page, select **Human Tasks**.
4. Click the name of the human task you want to open.

The human task is opened in the human task editor. See Section 11.2, "Introduction to the Human Task Editor" for more information.

11.3.3 How to Add Participants to a Human Task

Using the human task editor, you can add additional participants to a human task.

To add participants to a human task:

1. Open the human task.
2. Select the **General** tab.
3. In the **Participants** editor, select an existing participant.
4. Click the **Add** icon, then select one of the following routing types:
   - Sequential
   - Parallel

See Section 11.1.1.2, "Routing Types" for more information.

5. Select one of the following participant types:
   - Single
   - FYI
   - Serial
   - Parallel

   See Section 11.1.1.1, "Participant Types" for more information.

The new participant is added to the human task routing.

11.3.4 How to Configure the Outcome for Parallel Routing

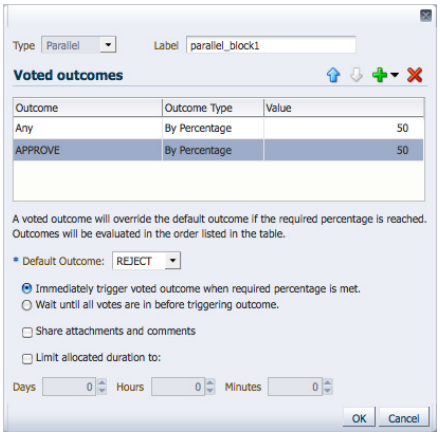
You can configure the routing outcome for parallel blocks.

To configure the outcome for parallel routing:

1. Open your human task.
2. Select the **General** tab.
3. In the **Participants** editor, double-click on one of the intersections of the parallel block.

   The Voted Outcomes editor appears as shown in Figure 11–5.

Figure 11–5 Voted Outcomes Editor



This graphic displays an editor window that contains various fields and properties. These are displayed within the text.

****

4. Click the Add icon, the select one of the following outcomes:
   - Approve
   - Reject
5. In the table, enter a value for the outcome.
6. Select a value for Default Outcome from the drop-down list.
7. Click **OK**.

11.3.5 How to Assign Users, Groups, or Roles to a Participant

You can assign users, groups or roles to a human task participant. See Section 11.1.2, "Introduction to Participant Assignment" for more information. about assignment.

To assign a user, group, or role to a participant using names and expressions:

1. Open your human task.
2. Select the **General** tab.
3. Double-click the participant.
4. Select names and expressions from the drop down list.
5. Click Add, then select one of the following:
   - Add User
   - Add Group
   - Add Application Role
6. Under **Data Type**, select one of the following:
   - By Name
   - By Expression
7. To assign a name to the participant:
   a. Click the **Search** icon in the **Value** column.
   b. Click **Search** to display a list of available users or groups.
   c. Select the user or group you want to add in the **Available** column.
   d. Click **Move Selected Items to Other list**.
   e. Click **OK**.
8. Click **OK**.

To assign a user, group, or role to a participant using lane participants:

1. Open your human task.
2. Select the **General** tab.
3. Double-click the participant.
4. Select **Lane Participants** from the drop down list.
5. Click Add, then select one of the following:
   - Previous lane participant
   - Current lane participant
6. Click **OK**.

To assign a user, group, or role to a participant using parametric roles:

Using Business Process Composer, you can configure a human task to use parametric roles.

Note: Although you can configure a human task to use parametric roles, you cannot create or configure the parametric roles using Business Process Composer. These must be created and assigned to the human task using Process Workspace or Oracle BPM Studio.

1. Open your human task.
2. Select the **General** tab.
3. Double-click the participant.
4. Select Parametric Role from the drop down list.
5. Click **OK**.

11.3.6 How to Define the Duration for a Participant

You can define the duration for each participant within a human task.

Additionally, you can define duration for the entire human task. See Section 11.3.7, "How to Define the Duration for a Human Task" for more information.

To create task data for a human task:

1. Open the human task where you want to assign a user, group, or role to a participant.
2. Select the **Data** tab.
3. Click the Add button.
4. Select one of the following:
   - string
   - int
   - boolean
   - TaskExecutionData

   The task data appears in Task Data table.
5. Enter a name for the new task data in the table.

11.3.7 How to Define the Duration for a Human Task

You can define the duration globally for a human task.

Additionally, you can define duration for each process participant within the human task. See Section 11.3.6, "How to Define the Duration for a Participant" for more information.

To define a duration for a human task:

1. Open the human task where you want to define the duration.
2. Select the **General** tab.
3. Double-click the participant.
4. From the Add drop-down menu, select one of the following:
   - Add User

- Add Group
- Add Application Role

5. Click **OK**.

11.3.8 How to Create Task Data for a Human Task

Using the human task editor, you can define the data types used within the human task. This data is used to store the information entered during user interaction.

To create task data for a human task:

1. Open the human task where you want to create task data.
2. Select the **Data** tab.
3. Click the Add button.
4. Select one of the following:
   - String
   - Int
   - Int(64)
   - Decimal
   - Real(32)
   - Real
   - Bool
   - Time Interval
   - TaskExecutionData

   The task data appears in Task Data table.
5. Enter a name for the new task data in the table

11.3.9 How to Specify the Presentation of a Human Task

Presentations define the user interface of your application. Oracle Business Process Composer does not allow you to view or edit the presentation of a human task. However, you can specify the connectivity information used to access the presentation at runtime.

> **Note:** This information should be provided by a process developer or administrator.

To specify the presentation of a human task:

1. Open the human task whose presentation you want to specify.
2. Select the **Data** tab.
3. Expand **Presentation**.
4. Enter the following:
   - Hostname: Specifies the hostname of the server where the presentation is deployed.

- HTTP Port: Specifies the port of the server where the presentation is deployed.
- HTTPS Port: Specifies the secure port of the server where the presentation is deployed.
- URI: Specifies the Uniform Resource Indicator of the presentation.

5. Click **OK**.

12

Performing Administrative Tasks

This chapter describes how to perform administrative using Oracle Business Process Composer.

> **Note:** The procedures described in this chapter can only be performed by users who have been granted the Project Administrator security role.

This chapter includes the following sections:

12.1 Introduction to Business Process Composer Administration

Business Process Composer enables you to assign administrator privileges to a user. Administrators can perform the following tasks:

- Assign global roles to users.
- Delete shared projects and project templates.
- Release locks on shared projects.
- Configure permissions on projects.
- Import project templates.

> **Note:** Administrators cannot perform tasks on private projects. Only the project owner has access to private projects.

12.2 How to Assign Global Roles

In addition to assigning project permissions for individual projects, BPM Composer enables administrators to assign permissions globally. The following global roles are available:

- **SOA Designer**: The SOA Designer role is a security role shared with the SOA infrastructure. Business Process Composer uses this role to enable users. See Section 8.4, "Editing Oracle Business Rules at Run Time" for more information.

  and groups to edit Oracle Business Rules at run time.

- **Project Administrator:** Users or groups assigned this role can perform the following actions on all shared projects:
  - Remove projects
  - Deploy projects
  - Define project permissions
  - Create project snapshots

To assign a global role:

1. Login to Business Process Composer as a user with administrator privileges.
2. Click **Administration**.
3. Select **Role Mapping**.
4. Select a role from the drop-down list.
5. Click **Choose**:
   a. Select from the following:

      All:

      Users:

      Groups:

   b. Click **Search** to view a list of all available users or groups.
   c. In the **Available** column, select the users or groups you want to add.
   d. Click **Move Selected Item to Other List**.
   e. Click **OK**.

12.3 How to Delete a Project or Project Template

Administrators can delete shared projects or project templates.

To delete a shared project:

1. Login to Business Process Composer as a user with administrator privileges.
2. Click **Administration**.
3. Select **Project Management** or **Template Management**.
4. Select a project or project template from the table.
5. Click **Delete**.
6. Click **Yes** to confirm that you want to delete the project.

12.4 How to Configure Sharing for a Project

Administrators can configure sharing for shared BPM projects.

To configure sharing on a project:

1. Login to Business Process Composer as a user with administrator privileges.
2. Click **Administration**.
3. Select **Project Management**.
4. Select a project from the table.
5. Click **Share**:
   a. Select one of the sharing visibility:

      Team members only: shares the project only with team members of the project owner.

      Public: shares the project with all Business Process Composer users.

   b. Click **Choose** to specify the users or groups who have access to the project.
   c. Select one of the following roles:

      Owner:

      Editor:

      Viewer:
6. Click **Share**.
7. Click **Close**.

12.5 How to Release the Lock on a Shared Project

Administrators can release the locks on shared projects that have been locked by another user.

To release the lock on a shared project:

1. Login to Business Process Composer as a user with administrator privileges.
2. Click **Administration**.
3. Select **Project Management**.
4. Select the project whose lock you want to release.
5. Click **Release Lock**.

> **Note:** If you release the lock of a project, any changes made by the user who originally locked the project will be lost.

12.6 How to Import a Project Template

Administrators can importa a project template from a local file system.

To import a project template:

1. Login to Business Process Composer as a user with administrator privileges.
2. Click **Administration**.
3. Select **Template Management**.
4. Click **Import Template**.

5. Click Browse to locate the project template on your local file system.
6. Provide a name for the project template.
7. Click **OK**.

A

BPMN Flow Object Property Reference

This appendix provides information for each of the BPMN flow object properties. It contains these topics:

A.1 Common Properties

This section describes common properties shared by multiple BPMN flow objects.

A.1.1 Basic Properties

Table A–1 lists the properties shared by all activities and gateways. These properties appear in the properties popup.

Table A–1 Basic Properties for Activities and Gateways

Property	Description
Name	Defines the name of this flow object. This becomes the name of the flow object within your process.
Description	Provides an optional description of this flow object. Adding a description can make your process more readable.
Type:	Determines the type of flow object. When editing a project you can change the type of flow object by selecting from the drop-down menu.
Icon:	Displays the icon used for this flow object. You can click **Change** to select a different icon.

A.1.2 Implementation Properties

Table A–2 lists the implementation properties that are shared by multiple BPMN flow objects.

Table A–2 Implementation Properties

Property	Description
Is Draft	.When selected, specifies that the flow object is a draft.
Sampling Point	Use to configure sampling points for this flow object. ■ **Inherit Process Default**: Select to use the default sampling configuration defined at the process level. ■ **Generate**: Select to generate sampling point data for this activity. This will override the default configuration defined at the process or project level. ■ **Do Not Generate**: Select to not generate sampling point data. This is primarily used for performance reasons. Sampling points enable you to generate information about the performance of an flow object within in a running process. The data generated according to this configuration is stored in the Process Analytics Database. Sampling point generation specified at the project level is applied to all of the processes within the project. See Section 4.5.12, "How to View and Edit Project Properties" for procedures for setting sampling point for a project. However, you can override project-level settings within a process. Likewise, sampling point generation specified at the process level is applied to all of the flow objects within the process. You can also override process-level settings within each flow object. Overriding sampling point generation at the project or process level is usually done to improve performance. For example, if your project contains a process that contains a great number of activities and you are not interested in obtaining process metrics for this process, you might choose to set its sampling point configuration so that the process does not generate sampling points. Likewise, if you are interested in measuring only one process within your project, you might choose to set the project not to generate sampling point and configure that particular process to generate sampling points. By default, the project sampling configuration is set to generate sampling points only for interactive activities.

A.2 Interactive Properties

This section describes the properties of the interactive activities.

A.2.1 Interactive Activities

The interactive activities represent parts of your process where a process participant is required to perform work. See Section 6.3.2, "Introduction to the User Task" for more information. This section lists properties for the following interactive activities:

- User Task
- FYI
- Management
- Group

- Complex
- Initiator

The properties popup of the interactive activities contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

Table A–3 describes the properties that can be edited from the **Implementation** editor.

Table A–3 Interactive Activities - Common Properties

Property	Description
Human Task	Defines the name of the human task assigned to this user task. You can select a list of human tasks in the business catalog.
Pattern	Displays the pattern used for this human task.
Re-initiate	Restarts the approval process from the beginning.

A.2.2 Manual Task

The manual task represents a task within a process that is performed by process participants that is outside of the scope of Oracle BPM.

See Section 6.3.3, "Introduction to the Manual Task" for more information.

The properties popup of the manual task contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.3 Activity Properties

The following sections describe the properties of each BPMN activities supported by the Oracle BPM Suite.

A.3.1 Service Task

The service task enables you to communicate with other processes and services.

See Section 6.4.1, "Introduction to the Service Task" for more information.

The service task contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.3.1.1 Implementation Properties

Table A–4 describes the properties that can be edited from the **Implementation** properties when you select the Service option.

Table A–4 Service Task Properties (When Process Call is Selected)

Property	Description
Conversation	Determines the type of conversation: Determines the type of conversation: ■ Default: enables you to configure the service conversation using only the process and target node. ■ Advanced: enables you to configure the conversation by defining a specific interface.
Name	Enables you to define the interface for the conversation. (This option is only available when Advanced is selected.)

Table A–4 (Cont.) Service Task Properties (When Process Call is Selected)

Property	Description
Process	Determines the BPMN process called by the service task. The process must be another process within the same BPM project.
Operation	Determines the specific node (flow object) within the BPMN process that is called by this service task.

Table A–5 Service Task Properties (When Service Call is Selected)

Property	Description
Conversation	Determines the type of conversation: ■ Default: enables you to configure the service conversation using only the process and target node. ■ Advanced: enables you to configure the conversation by defining a specific interface.
Name	Enables you to define the interface for the conversation. (This option is only available when Advanced is selected.)
Service	Determines the service called by this service task. This service must be defined in the business catalog of the BPM project.
Operation	Determines the operation called by the service task.

A.3.2 Send Task

The send task sends a message to a system or process outside the current process.

See Section 6.4.4, "Introduction to the Send Task" for more information.

The send task contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.3.2.1 Implementation Properties

Table A–6 describes the implementation properties of the send task when **Define Interface** is selected.

Table A–6 Send Task Properties (When Define Interface is Selected)

Property	Description
Default	Defines the interface using only argument definitions that are passed from the service task to the service being called. You can configure the required arguments by clicking the **Add** button. Valid values for argument definitions are: ■ **Name**: defines the name of the argument. ■ **Type**: defines the data type of the argument.
Advanced	Enables you to select the operation name in addition to defining the argument definitions. ■ **Operation Name**:
Asynchronous	Indicates that the interface is called asynchronously.

Table A–6 (Cont.) Send Task Properties (When Define Interface is Selected)

Property	Description
Synchronous	Indicates that the interface is call synchronously. You can also define the following properties: ■ **Reply To**: ■ **Throw Error**: Indicates that the send task uses an error handler when a problem occurs. ■ **Error**: Determines the error called when a problem occurs. This error must be defined in the business catalog.

Table A–7 describes the implementation properties available when **Initiates** is selected.

Table A–7 Send Task Properties (When Use Interface is Selected)

Property	Description
Name	Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Reference	
Operation	Determines the operation invoked by the send task.
Error	Determines the error called when a problem occurs. This error must be defined in the business catalog.

Table A–8 describes the implementation properties when **Continues** is selected.

Table A–8 Send Task Properties (When Process Call is Selected)

Property	Description
Name	Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Process	Defines the BPMN process this send task invokes.
Target Node	Determines the flow object within the BPMN process that is called by the service task.

Table A–9 describes the implementation properties when **Continues** is selected.

Table A–9 Send Task Properties (When Service Call is Selected)

Property	Description
Name	Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Service	Determines the service this send task invokes
Operation	Determines the operation invoked by the send task.

A.3.3 Receive Task

The receive task waits for a message from a system or process outside the current process.

See Section 6.4.5, "Introduction to the Receive Task" for more information.

The receive task contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.3.3.1 Implementation Properties

Table A–10 describes the properties of the receive task when Define Interface is selected.

Table A–10 Receive Task (When Define Interface is Selected)

Property	Description
Name	Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Arguments definition	Lists the arguments required to invoke the operation the receive task exposes. These are the arguments passed to the process from the invoking process or service.
Operation Name	.Defines the operation invoked by the received task.

Table A–11 describes the implementation properties when **Use Interface** is selected.

Table A–11 Receive Task Properties (When Use Interface is Selected)

Property	Description
Name	Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Reference	
Operation	Defines the operation invoked by the receive task.

Table A–12 describes the implementation properties when **Service Call** is selected.

Table A–12 Receive Task Properties (When Service Call is Selected)

Property	Description
Name	.Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Service	Displays the service this receive task invokes.
Operation	Determines the name of the operation this receive task invokes.

A.3.4 Business Rule Task

The business rules task enables you to incorporate Oracle Business Rules within your process.

See Section 6.5.2, "Introduction to the Business Rule Task" for more information.

The business rule task contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.3.4.1 Implementation Properties

Table A–13 describes the properties that can be edited from the **Implementation** editor.

Table A–13 Business Rule Task Properties

Property	Description
Rule	Determines the business rule assigned to this business rules task.
Operation	Specifies the decision function of the rule specified above.

A.3.5 Script Task

The script task is used to change values of data objects within your process

See Section 6.10.1, "Introduction to the Script Task" for more information.

The script task contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of theseS properties.

A.3.6 Call Activity

The call activity allows you to call a reusable process from within the current process.

See Section 6.4.3, "Introduction to the Call Activity" for more information.

The call activity contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.3.6.1 Implementation Properties

Table A–14 describes the properties that can be edited from the **Implementation** properties.

Table A–14 Call Activity Properties

Property	Description
Process	Specifies the BPMN process invoked by the call activity.

A.3.7 Subprocesses

Subprocesses allow you to group BPMN flow objects together to make your process more readable.

See Section 6.9.1, "Introduction to Subprocesses" for more information.

Subprocesses contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.3.7.1 Implementation Properties

Table A–15 describes the properties that can be edited from the Implementation editor.

Table A–15 Subprocesses Properties

Property	Description
Loop characteristics	Determines how many time the subprocess will repeat. This property is read-only Oracle Business Process Composer.

A.3.8 Inline Handlers

Inline handlers are types of subprocesses that allow you to model conditions that happen outside of a normal process flow.

See Section 6.9.2, "Introduction to Inline Handlers" for more information.

Inline handlers contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.4 Gateway Properties

The following sections describe the properties for each BPMN gateway.

A.4.1 Exclusive Gateway

The exclusive gateway enables you to split a process into two or more paths.

See Section 6.7.2, "Introduction to the Exclusive Gateway" for more information.

The exclusive gateway contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

Table A–16 describes the properties that can be edited from the **Outflows Order**.

Table A–16 Exclusive Gateway Properties

Property	Description
Order	Enables you to determine the order in which outgoing sequence flows are evaluated. The first condition that evaluates to true determines the path the process follows.

A.4.2 Inclusive Gateway

The inclusive gateway enables you to split your process into two or more paths.

See Section 6.7.3, "Introduction to the Inclusive Gateway" for more information.

The inclusive gateway contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.4.3 Parallel Gateway

The parallel gateway enables you to split your process into two or more paths when you want your process flow to follow all paths simultaneously.

See Section 6.7.4, "Introduction to the Parallel Gateway" for more information.

The parallel gateway contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.4.4 Complex Gateway

The complex gateway splits a process similar to an inclusive gateway. However, it enables you to define a condition that determines if the instance can continue even if not all of the tokens have arrived at the complex gateway merge.

See Section 6.7.5, "Introduction to the Complex Gateway" for more information.

The complex gateway contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

Table A–17 describes the properties that can be edited from the **Implementation** editor.

Table A–17 Complex Gateway Properties

Property	Description
Activation Condition	Enables you to define a condition that specifies when the gateway releases the tokens that arrive to it. Each time a new token arrives to the complex gateway the BPMN Service Engine evaluates this condition. If the condition evaluates to true, then the complex gateway releases all the tokens that arrived until that moment.

A.4.5 Event-Based Gateway

The even based gateway enables you to branch your process flow based on the possibility that an event may occur.

See Section 6.7.6, "Introduction to the Event-based Gateway" for more information.

The event-based gateway contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

Table A–18 describes the properties of the event-based gateway.

Table A–18 Event-Based Gateway Properties

Property	Description
Instantiate	Causes the event-based gateway to create a new process instance.

A.5 Event Properties

The following sections describe the properties for each type of events supported by Oracle BPM.

A.5.1 The None Start Event

The none start event is used when no instance trigger is specifically defined. See Section 6.2.3, "Introduction to the None Start Event" for more information.

A.5.2 The Message Start Event

The message start event triggers a process instance when a message is received.

See Section 6.2.4, "Introduction to the Message Start Event" for more information.

A.5.2.1 Implementation Properties

Table A–19 describes the basic properties that can be edited from the **Implementation** editor.

Table A–19 Message Start Properties

Property	Description
Type	This property is read-only for message start events. Message start events can only initiate a conversation between two processes.

Table A–19 (Cont.) Message Start Properties

Property	Description
Implementation	Enables you to determine how the receive task defines a conversation with the send task that invokes it. ■ **Not Implemented**: No implementation is specified. ■ **Define Interface**: Enables you to define how the process is exposed as a service to other BPMN processes and services. Argument definition: Defines the arguments required by the receive task. These are the arguments passed to the process from the invoking process or services. Type: Defines whether the process is invoke synchronously or asynchronously. Operation Name: Defines the name of the operation for this receive task. Other processes and services that invoke this receive task use this operation name. ■ Use From Catalog: Enables you to select an interface defined in the business catalog. Name: Defines the name of the interface. Operation: Determines the operation within the interface used by the receive task.

A.5.3 The Timer Start Event

The timer start event triggers the creation of a process instance based on a specific time condition. See Section 6.2.6, "Introduction to the Timer Start Event" for more information.

A.5.3.1 Implementation Properties

Table A–20 describes the properties that can be edited from the **Implementation** editor.

Table A–20 Timer Start Event Properties

Property	Description
Due Type	.Determines whether the timer start event creates a process instance based on a specific date or after a specific interval has passed. ■ Date: Enables you to specify the date and time when the timer start event creates a new process instance. The time and date are specified according to the following format: day/month/year hour:minute PM/AM ■ Interval: Enables you to specify the interval that the timer even waits to create a new process instance. This is specified in months, days, hours, seconds according to the following format: <*number*>M,<*number*>d,<*number*>h,<*number*>s
Expression Mode	Enables you to define either the date or interval using an expression.

A.5.4 The Signal Start Event

The signal start event is similar to a message start event in that it is based on communication from another process or service.

See Section 6.2.5, "Introduction to the Signal Start Event" for more information.

A.5.4.1 Implementation Properties

Table A–21 describes the properties that can be edited from the **Implementation** editor.

Table A–21 SIgnal Start Event Properties

Property	Description
Event	Defines the event used to trigger the signal start event. Events are defined in the business catalog.

A.5.5 The Error Start Event

The error start event is used as the start event of an inline handler.

See Section 6.2.7, "Introduction to the Error Start Event" for more information.

A.5.5.1 Implementation Properties

Table A–22 describes the properties that can be edited from the **Implementation** editor.

Table A–22 Error Start Event Properties

Property	Description
Exception	.Defines the error exception implemented by the error start event. This is stored in the business catalog.
Catch all Business Exceptions	Select to allow the catch event to catch any business exception.
Catch all System Exceptions	Select to allow the catch event to catch any system exception

A.5.6 None Catch Event

The none catch event is used as a place holder in your process.

The none catch event contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.5.7 Message Catch Event

The message catch event enables you to receive a message from another process or service.

See Section 6.4.8, "Introduction to the Message Catch Event" for more information.

The message catch event contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.5.7.1 Implementation Properties

Table A–23 describes basic implementation properties of the receive task.

Table A–23 Message Catch Properties

Property	Description
Type	Determines how the conversation of the message catch event is implemented. A conversation defines the sequence of a group of message events that communicate with other processes or services. A message event can start a conversation with another process or service, or continue a conversation initiated by a previous message event. Possible values are: ■ Initiates: Invokes another BPMN process or service. ■ Continues: Continues the conversation of a process that was previously invoked.

Table A–24 describes the implementation properties when **Initiates** is selected.

Table A–24 Message Catch Properties (When Initiates is Selected)

Property	Description
Implementation	Enables you to determine how the message catch event defines a conversation with the process or service that invokes it. ■ **Not Implemented**: No implementation is specified. ■ **Define Interface**: Enables you to define how the process is exposed as a service to other BPMN processes and services. Argument definition: Defines the arguments required by the message catch event. These are the arguments passed to the process from the invoking process or services. Type: Defines whether the process is invoked synchronously or asynchronously. Operation Name: Defines the name of the operation for this message catch event. Other processes and services that invoke this message catch use this operation name. ■ Use From Catalog: Enables you to select an interface defined in the business catalog. Name: Defines the name of the interface. Operation: Determines the operation within the interface used by the message catch event.

Table A–25 describes the implementation properties when **Continues** is selected.

Table A–25 Message Catch Properties (When Continues is Selected)

Property	Description
Initiator Node	Allows you to select the message event in the current process that precedes this message catch event.

A.5.8 Timer Catch Event

Timer catch events enable you to control the flow of your process using a time condition.

See Section 6.8.2, "Introduction to the Timer Catch Event" for more information.

The the timer catch event contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.5.8.1 Implementation Properties

Table A–26 describes the properties that can be edited from the **Implementation** editor.

Table A–26 Timer Catch Event Properties

Property	Description
Due Type	.Determines whether the timer catch event creates a process instance based on a specific date or after a specific interval has passed. ■ Date: Enables you to specify the date and time when the timer catch event is triggered. The time and date are specified according to the following format: day/month/year hour:minute PM/AM ■ Interval: Enables you to specify the interval that the timer even waits trigger the event. This is specified in months, days, hours, seconds according to the following format: <*number*>M,<*number*>d,<*number*>h,<*number*>s
Expression Mode	Enables you to define either the date or interval using an expression.

A.5.9 Error Catch Event

Error catch events are intermediate events used to handle an error that occurs within your process flow.

The error boundary catch event contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.5.9.1 Implementation Properties

Table A–27 describes the properties that can be edited from the **Implementation** editor.

Table A–27 Error Catch Event Properties

Property	Description
Exception	Defines the error exception implemented by the error catch event. This is stored in the business catalog.
Catch all Business Exceptions	Select to allow the error catch event to catch any business exception.
Catch all System Exceptions	Select to allow the error catch event to catch any system exception

A.5.10 Message Throw Event

The message throw event enables you to send a message to another process or service.

The message throw event contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.5.10.1 Implementation Properties

Table A–28 describes basic implementation properties of the send task.

Table A–28 Message Throw Event Properties

Property	Description
Type	Defines how the conversation of the message throw event is implemented. A conversation defines the sequence of a group of message events that communicate with other processes or services. A message throw event can start a conversation with another process or service, or continue a conversation initiated by a previous message event. Possible values are: ■ Initiates: Invokes another BPMN process or service. ■ Continues: Continues the conversation of a process that was previously invoked.

Table A–29 describes the implementation properties available when **Initiates** is selected.

Table A–29 Message Throw Event Properties (When Initiates is Selected)

Property	Description
Implementation	The implementation drop-down menu enables you to determine how the send task is implemented. ■ **Not Implemented**: No implementation is specified. ■ **Service Call**: Configures the message throw event to invoke a service contained in the business catalog. Name: Determines the service invoked by the message throw event. Operation: Defines which operation within the service is invoked. ■ **Process Call**: Configures the message throw event to invoke another BPMN process. Process: Determines the BPMN process called by the message throw event. Node: Determines the flow object called by the message throw event.

Table A–30 describes the implementation properties when **Continues** is selected.

Table A–30 Message Throw Event Properties (When Continues is Selected)

Property	Description
Initiator Node	Determines the message event that precedes this send task within the conversation.
Inputs	Defines the arguments required to invoke the operation the message start event exposes.
Type	Displays the process type as defined in the initiator. This property is read-only.
Operation name	Defines the name of the operation for this message catch event. Other processes and services that invoke this message catch use this operation name.

A.5.11 Signal Throw Event

You can use signal events to communicate a message to all the processes that are configured to wait for that message.

The signal throw event contains properties shared by multiple BPMN flow objects. See Section A.1, "Common Properties" for a list of these properties.

A.5.11.1 Implementation Properties

Table A–31 describes the properties that can be edited from the **Implementation** editor.

Table A–31 Signal Throw Event Properties

Property	Description
Event	Defines the event used to trigger the signal start event. Events are defined in the business catalog

A.5.12 None End Event

The none end event is used as a place holder in your process.

See Section 6.2.8, "Introduction to the None End Event" for more information on using the none end event.

A.5.13 Message End Event

The message end event is used to send a message to another process or service when the process is completed.

See Section 6.2.10, "Introduction to the Message End Event" for more information.

A.5.13.1 Implementation Properties

Table A–32 describes the properties that can be edited from the **Implementation** editor.

Table A–32 Message End Properties

Property	Description
Type	Defines how the conversation is implemented. Since message end events can only be configured to continue a conversation, this property is read-only.
Initiator Node	Determines the message event that precedes this send task within the conversation.
Inputs	For a synchronous process, this property defines the output arguments used to invoke the operation defined by the start or catch message event. For an asynchronous, this property defines the input and output arguments required by the callback operation defined by this end event.
Type	Displays the process type as defined in the initiator. This property is read-only.

Table A–32 (Cont.) Message End Properties

Property	Description
Operation Name	For an asynchronous process, this property defines the name of the operation for this message catch event. Other processes and services that invoke this message catch use this operation name. For a synchronous process, this property defines the operation of the event that precedes this end event. It can be a start or a catch event.

A.5.14 Signal End Event

You can use the signal end event to communicate a message to all the processes that are configured to wait for that message. This message communicates to these processes that the current process has finished.

A.5.14.1 Implementation Properties

Table A–33 describes the properties that can be edited from the **Implementation** editor.

Table A–33 SIgnal End Event Properties

Property	Description
Event	Defines the event used to trigger the signal start event. Events are defined in the business catalog.

A.5.15 Error End Event

The end error event is used when the end of a process is the result of an error condition.

See Section 6.2.9, "Introduction to the Error End Event" for more information.

A.5.15.1 Implementation Properties

Table A–34 describes the properties that can be edited from the **Implementation** editor.

Table A–34 Error End Event Properties

Property	Description
Exception	Defines the error exception implemented by the error catch event. This is stored in the business catalog.

A.5.16 Terminate End Event

The terminate end event is used to immediately stop a process. When a terminate end event is reached, the process stops immediately.

See Section 6.2.11, "Introduction to the Terminate End Event" for more information.

A.6 Measurement Mark Properties

Measurement marks enable you to measure a business indicator of type measure at a certain point in the process or in a section of the process. The following types of measurement marks are supported:

- Start Measurement Mark

- End Measurement Mark
- Snapshots

Table A–35 Measurement Mark Properties

Property	Description
Name	Defines the name for this measurement mark.
Description	.Provides an optional description for this measurement mark.
Type	Displays the type of measurement mark. This property is read-only.
Business Indicators	Defines the business indicators assigned to this measurement mark. See Section 10.3, "Working with Business Indicators and Counter Marks" for more information.

.

A.7 Sequence Flow Properties

Sequence flows define the order or sequence that work is performed within a process. The following sections describe the sequence flow properties you can edit using Oracle Business Process Composer.

A.7.1 Default Sequence Flow

Table A–36 describes the properties of default sequence flows.

Table A–36 Default Sequence Flow Properties

Property	Description
Name	Defines the name of this sequence flow. This name appears next to the sequence flow in your process diagram.
Description	Provides an optional description of this sequence flow. Adding a description can make your process more readable.

A.7.2 Normal Sequence Flow

Table A–37 describes the properties of default sequence flows.

Table A–37 Normal Sequence Flow Properties

Property	Description
Name	Defines the name of this sequence flow. This name appears next to the sequence flow in your process diagram.
Description	Provides an optional description of this sequence flow. Adding a description can make your process more readable.

A.7.3 Conditional Sequence Flow

Table A–38 describes the properties of default sequence flows.

Table A–38 Conditional Sequence Flow Properties

Property	Description
Name	Defines the name of this sequence flow. This name appears next to the sequence flow in your process diagram.
Description	Provides an optional description of this sequence flow. Adding a description can make your process more readable.
Condition	Specifies the expression used to evaluate this conditional sequence flow. You can define an expression by clicking **Edit** to launch the expression editor. See Section A.4.1, "Exclusive Gateway" for more information on configuring the order in which conditional sequence flows are evaluated.

B

Preparing Processes for Import into BPMN

This appendix provides information for each of the BPMN flow object properties. It contains these topics:

- Section B.1, "Preparing a Visio File to Import as a BPMN Process"
- Section B.2, "How to Customize XPDL Import Using XSL Doc"
- Section B.3, "Preparing an XPDL File for Import as a BPMN Process"

B.1 Preparing a Visio File to Import as a BPMN Process

You can import Visio files into Business Process Composer. Business Process Composer maps Visio elements to BPMN flow objects using a map file named *VisioMasterMap.xml*.

This file defines how each Visio element is mapped to a BPMN flow object. It is located in the following directory in the JDeveloper home directory:

`.../jdeveloper/jdev/extensions/tutor/xml.`

> **Note:** Do not edit this file. You can override the mappings in this file by creating and editing VisioUserMap.xml.

Business Process Composer checks for a VisioUserMap.xml file after it has parsed VisioMasterMap.xml and then modifies its rules accordingly. The user map serves to extend and/or override the master map.

The Visio import module is designed for extensibility. Users may add an XML file to support alternate Visio stencils and specify the mapping of additional Visio master shapes to BPMN objects. This file must be named "VisioUserMap.xml", and it must be placed in the same directory as the "VisioMasterMap.xml" file.

This file should have the same root element as the supplied "VisioMasterMap.xml" file, including the reference to the VisioMasterMap.xsd. "VisioUserMap.xml" must have the same format as the master map, which may be used as a reference. Entries added to the user map either add mappings or overwrite an existing entry in the main map. It should contain only the extended or overridden entries and should not repeat all the original entries.

Example B–1 Example Entry for VisioUserMap.xml

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<Masters xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://www.oracle.com/oracle/tutor/visio/masterMapElements
VisioMasterMap.xsd"
 xmlns="http://www.oracle.com/oracle/tutor/visio/masterMapElements">
<!— custom <Master/> elements go here -- >
</Masters>
```

B.1.1 How to Update VisioUserMap.xml

The following procedure describes a possible scenario for importing a Visio file to a BPMN process.

To update VisioUserMap.xml

In this scenario, assume that the source Visio file contains a shape named Report, that you what to map to a data object. By default, VisioMasterMap maps this shape to a send task. If you use it strictly as an input /output object, it is more accurate to map it to a Data Object.

1. In the VisioMasterMap.xml file, locate the master definition for DataObject which appears as:

   ```
   <Master Name="data object">
       <BPMNObject>DataObject</BPMNObject>
   </Master>
   ```

   This definition is followed by a series of additional master elements that use the *Like* attribute to clone the first definition and map additional Visio objects:

   ```
   <Master Name="sequential data" Like="data object"/>
   <Master Name="data" Like="data object"/>
   ```

2. To add or modify a mapping for a Visio *Report* element, add the following code to the VisioUserMap.xml file:

   ```
   <Master Name=”Report” Like=”data object”/>
   ```

Example B–2 shows how an object can be mapped to an exiting BPMN object with additional or different attributes by using the *Extends* attribute.

Example B–2 Mapping a Visio Object to a BPMN Object Using the Extends Attribute

```
<Master Name="flow">
    <BPMNObject>SequenceFlow</BPMNObject>
</Master>

<Master Name="conditional flow" Extends="flow">
    <Attributes>
      <Attribute Name="ConditionType" Value="Expression"/>
    </Attributes>
</Master>
```

In this example, the Visio object *conditional flow* is mapped to the SequenceFlow BPMN flow object, but has added the attribute “ConditionType” to the value “Expression.”

B.1.2 Valid BPMN Element Values

The following are the valid values for the <BPMNObject> element:

- Task
- Subprocess
- Event

- Gateway
- DataObject
- Group
- Annotation
- Lane
- Pool
- MessageFlow
- SequenceFlow
- Association
- null

Note: When you assign the null value to a Visio element, no BPMN flow object is created.

B.1.3 BPMN Element Attributes

The following tables show valid values for BPMN attributes based on the basic BPMN types of BPMN flow objects:

Task attributes and values

Table B–1 show the valid attributes and values for BPMN tasks.

Table B–1 Task Attributes and Values

Attribute	Values
TaskType	None, Script, Reference, Service, User, Manual, Send, Receive
LoopType	Standard, MultiInstance
isForCompensation	true, false

Subprocess attributes and values

Table B–2 show the valid attributes and values for BPMN subprocesses

Table B–2 Subprocess Attributes and Values

Attribute	Values
isExpanded	true, false
isATransaction	true, false
LoopType	Standard, MultIinstance
isForCompensation	true, false
AdHoc	true, false

Event attributes and values

Table B–3 show the valid attributes and values for BPMN events.

Table B–3 Event Attributes and Values

Attribute	Values
EventType	Start, Intermediate, End
Trigger	TriggerNone, Message, Timer, Conditional, Signal, Multiple, Error, Cancel, Compensation, Link, Terminate

Gateway attributes and values

Table B–4 show the valid attributes and values for BPMN gateways.

Table B–4 Gateway Attributes and Values

Attribute	Values
GatewayType	Parallel, Inclusive, Exclusive, Complex
ExclusiveType	Event, Data
MarkerVisible	true, false

Sequence flow attributes and values

Table B–5 show the valid attributes and values for BPMN sequence flows.

Table B–5 Sequence Flow Attributes and Values

Attribute	Values
ConditionType	None, Expression, Default

Association attributes and values

Table B–6 show the valid attributes and values for BPMN data objects.

Table B–6 Association Attributes and Values

Attribute	Value
Direction	None, One

Pool attributes and values

Table B–7 show the valid attributes and values for BPMN pools.

Table B–7 Pool Attributes and Values

Attribute	Values
BoundaryVisible	true, false

B.2 How to Customize XPDL Import Using XSL Doc

Often the conversion of models from source to target follows general transformation rules and has guidelines and constraints. If an XPDL document is imported and does not produce desirable results, there may be a mismatch between the format Business Process Composer is expecting and the format of the XPDL document. Under such circumstances, an XSLT transform may be used to create a better match between the original XPDL diagram and the Business Process Composer results.

The following procedure describes the recommended process for incorporating an XSL style sheet to modify the output of an Oracle BPM conversion. Oracle BPM ships with several XSLT files that may serve as examples to XSLT developers.

To customize XPDL Import Using XSL Doc:

Business Process Composer uses XSLT transformations to parse XPDL files and generate a consistent format before the data is passed to BPMN and displayed as a diagram. Business Process Composerwill search for a transformation file associated with an XPDL document.

Typically, an XPDL file contains a <*Vendor*/> element that provides the vendor name. By adding the Vendor's name and the path to an XSL file to Oracle BPM's "XSLFilePaths.xml", the vendor name will be searched to locate the associated transformation file's path.

The file path is retrieved from index file, and Business Process Composer applies the XSLT transformation on the XPDL file.

1. Create a custom XSLT file.

   Using an XML editor, create a new XSLT file. To view examples, see Appendix 3: XPDL Technical Details. It is best to put the XSLT file in the "XML" directory that contains "XSLFilePaths.xml".

2. Open the XPDL file to be imported and locate the <Vendor/> element under the <PackageHeader/> element, and note its value.

   For example, if the XPDL looks like this:

   ```
   <Package xmlns="http://www.wfmc.org/2004/XPDL2.0alpha" Id="6" Name="Untitled
   Document 6">
       <PackageHeader>
   <Vendor>Global 360</Vendor>
   …
   </PackageHeader>
   …
   </Package>
   ```

   The Vendor element's value is "Global 360".

3. Locate the "XSLFilePaths.xml" file.

   This file is located in a folder named "XML" which is nested beneath your JDeveloper install directory as follows: [JDeveloper install]\jdeveloper\jdev\extensions\tutor\xml.

4. Add a new <XSLFilePath/> element with a Vendor attribute containing the name of the vendor, and provide the relative path to the XSLT file as the value.

   For example:

   ```
   <XSLFilePaths xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="filesPaths PatchFilesPaths.xsd"
   xmlns="filesPaths">
   …
   <XSLFilePath Vendor=”Global 360”>Global360Patch.xsl</XSLFilePath>
   </XSLFilePaths>
   ```

   Note: In this example the file name "Global360Patch.xsl" has no additional path information. Business Process Composer will search for this file in the same folder that contains the "XSLFilePaths.xml" file.

   If additional path information is present, Business Process Composer will search for the file relative to the \xml folder.

5. Import the XPDL file in Oracle BPM. Your file will be transformed and processed according to the rules specified in the custom XSLT file.

B.3 Preparing an XPDL File for Import as a BPMN Process

For importing XDPL files that do not conform to XPDL 2.1 or might be missing elements that Oracle BPM expects, conversion is extensible through the use of XSLT. See Section B.1, "Preparing a Visio File to Import as a BPMN Process" for instructions on integrating custom XSLT to produce the best results. This appendix will provide several sample XSLT templates to handle common situations that an Oracle BPM user might encounter when importing XPDL files.

If you are using Tutor the files mentioned in this appendix are located in a directory named “XML” in the Tutor install directory. If you are working with BPM, they located under your JDeveloper install directory, then navigate to /jdeveloper/jdev/extensions/tutor/xml.

The following are requirements for importing business processes into Oracle BPM:

- Oracle BPM supports importing XPDL 2.1 files. The following namespace must be present in the root element [i.e. “<Package/>”] of the XPDL source document as the default namespace: http://www.wfmc.org/2008/XPDL2.1
- If the XPDLsource file targeted for import does not conform to XPDL 2.1, then an XSLT must be performed to generate the correct XPDL format for import into Oracle BPM. The following sections provide some guidance for creating those transforms.

B.3.1 Handling Namespaces

You must be familiar with namespaces handling in XSLT when importing XPDL. The default namespace used by Oracle BPM is:

```
http://www.wfmc.org/2008/XPDL2.1.
```

In some cases a source file may be XPDL 2.1 compliant, but still lack elements required by Oracle BPM. In such cases, XSLT can be used to add those elements. Refer to the scenarios below for a reference to such a transformation.

Any other default namespace in an XPDL document will result in elements and attributes being treated as conforming to XPLD 2.1 during import, which will likely produce poor results. XPDL source documents that do not conform to XPDL 2.1 must be transformed via XSL into XPDL 2.1 result documents.

The following possible scenarios exist:

- The source XPDL elements user the default namespace and conform to XPDL 2.1 See stylesheet “BizAgiPatch.xsl” as an example for handling this scenario.
- The source XPDL elements are in default namespace but they don’t conform to xpdl 2.1. See stylesheet “patch.xsl” as an example for handling this scenario.
- The source XPDL elements are not in default namespace and they don’t conform to XPDL 2.1. See stylesheet “ALBPMPatch.xsl” ” as an example of handling this scenario.
- 4.The source XPDL elements are not in default namespace but they conform to xpdl 2.1. See stylesheet “TibcoPatch.xsl” as an example of handling this scenario

Example B–3 shows an XSL that copies elements written using a local namespace in the source document to the default namespace in the result document.

Note: The local namespace is added in the stylesheet's root element. When matching elements in the source XPDL document, the local namespace prefix is required for a match to be successful. This example uses ANY_LOCAL_NAMESPACE.

Example B–3 Copying Elements From Local Namespace to the Default Namespace

```
<xsl:stylesheet version = "1.0"
                                xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
                                xmlns:ANY_LOCAL_NAMESPACE = "http://www.someURI" >
<xsl:output method = "xml" indent = "yes"/>
<xsl:template match="@*|node()">
<xsl:copy>
<xsl:apply-templates select="@*|node()"/>
</xsl:copy>
</xsl:template>
<xsl:template match = "ANY_LOCAL_NAMESPACE:Activity">
……
</xsl:template>
</xsl:stylesheet>
```

B.3.2 Handling Relative Coordinates

XPDL documents may have object coordinates defined as relative or absolute. An XPDL document using absolute coordinates will import all coordinates as absolute coordinates.

When importing XPDL documents into Oracle BPM using relative coordinates you should be aware of the following conditions:

- Pool coordinates are absolute
- Lane coordinates are relative to their parent pool
- Node coordinates not in a subprocess are absolute
- Node coordinates in an embedded subprocess (expanded or collapsed) are relative to the upper-left corner of the subprocess
- Sequence Flow path coordinates connecting nodes in a subprocess are relative to the upper-left corner of the subprocess
- All other flows are absolute

An XPDL document with relative coordinates must be converted to meet the above conditions. The following information may be helpful in converting coordinates from relative to absolute or vice versa when necessary:

- Pool coordinates

  Pool coordinates must be absolute. If coordinates provided for Pools are relative, then OBPC will render all Pools overlapping each other. OBPC assumes Pools contain absolute coordinates. To override this, convert Pool coordinates to absolute.

- Lane coordinates

  Lane coordinates must be relative to their parent pool. In some instances, lanes have absolute coordinates but the model has relative coordinates. In this case lane coordinates can be converted as follows.

Lane X-Coordinate can be taken as the difference between lane X-coordinate and its parent pool X-Coordinate. The same logic applies to the Y-coordinate.

The following XSL example shows how to perform this conversion:

```
<xsl:template match =
"xpdl2:Lane/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coordinates">

        <xsl:copy>
            <xsl:copy-of select = "@*[name() != 'XCoordinate' and name() !=
'YCoordinate']"/>
            <xsl:attribute name = "XCoordinate">
                <xsl:value-of select =
"ancestor::xpdl2:Pool/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coor
dinates/@XCoordinate - @XCoordinate"/>
            </xsl:attribute>
            <xsl:attribute name = "YCoordinate">
                <xsl:value-of select =
"ancestor::xpdl2:Pool/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coor
dinates/@YCoordinate - @YCoordinate"/>
            </xsl:attribute>
            <xsl:apply-templates/>
        </xsl:copy>
    </xsl:template>
```

- Node coordinates not in a subprocess

  Node coordinates not in a subprocess must be absolute. If node coordinates are relative to their parent lane or pool, convert them to absolute.

  Coordinates for activities relative to the parent pool can be converted to absolute by adding their pool coordinates to the activity coordinates.

  Coordinates for activities relative to parent Lanes, which are relative to the parent Pool, can be converted to absolute by assigning the sum of the activity, parent lane and parent pool coordinates to activity coordinate.

  The following example shows how to calculate activity coordinates if they are relative to the parent lane and the lane coordinates are relative to the parent pool where the pool coordinates are absolute.

```
<xsl:template
match="xpdl2:Activity/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coor
dinates">
        <xsl:copy>
            <xsl:copy-of select="@*[name()!='YCoordinate' and
name()!='XCoordinate']"/>
            <xsl:variable name="LaneId">
                <xsl:value-of
select="ancestor::xpdl2:NodeGraphicsInfo/@LaneId"/>
            </xsl:variable>
            <xsl:variable name="PoolId">
                <xsl:value-of
select="//xpdl2:Lane[@Id=$LaneId]/ancestor::xpdl2:Pool/@Id"/>
            </xsl:variable>
            <xsl:attribute name="YCoordinate">
                <xsl:value-of
select="//xpdl2:Pool[@Id=$PoolId]/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInf
o/xpdl2:Coordinates/@YCoordinate
                +
//xpdl2:Lane[@Id=$LaneId]//xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2
:Coordinates/@YCoordinate
```

```
  +
@YCoordinate"/>
            </xsl:attribute>
            <xsl:attribute name="XCoordinate">
                <xsl:value-of
select="//xpdl2:Pool[@Id=$PoolId]/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInf
o/xpdl2:Coordinates/@XCoordinate
                +
//xpdl2:Lane[@Id=$LaneId]//xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2
:Coordinates/@XCoordinate
  +
@XCoordinate"/>
            </xsl:attribute>
        </xsl:copy>
    </xsl:template>
```

- Node coordinates in an embedded subprocess

  Node coordinates in an embedded Subprocess (expanded or collapsed) must be relative to the upper-left corner of the Subprocess

  For child nodes of a Subprocess, coordinates are relative to the parent Subprocess.

  If coordinates of child nodes of a Subprocess are not relative to the parent Subprocess, it is necessary to determine whether the coordinates are relative to the parent Lane or parent Pool.

  To make Node coordinates relative to the Subprocess, subtract the appropriate Subprocess coordinates from the node coordinates and assign the resulting values to the node coordinates.

  A sample style sheet template is given below that calculates the coordinates of the child nodes of a Subprocess. This template will work if the child node coordinates are not relative to a Subprocess.

```
<xsl:template match =
"xpdl2:ActivitySet/xpdl2:Activities/xpdl2:Activity/xpdl2:NodeGraphicsInfos/xpdl
2:NodeGraphicsInfo/xpdl2:Coordinates">
        <xsl:variable name = "ActivitySetId">
            <xsl:value-of select = "ancestor::xpdl2:ActivitySet/@Id"/>
        </xsl:variable>
        <xsl:copy>
            <xsl:copy-of select = "@*[name() != 'XCoordinate' and name() !=
'YCoordinate']"/>
            <xsl:attribute name = "XCoordinate">
                <xsl:value-of select = "@XCoordinate  -
 //xpdl2:Activity[@Id =
$ActivitySetId]/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coordinate
s/@XCoordinate"/>
            </xsl:attribute>
            <xsl:attribute name = "YCoordinate">
                <xsl:value-of select = "@YCoordinate -
//xpdl2:Activity[@Id =
$ActivitySetId]/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coordinate
s/@YCoordinate"/>
            </xsl:attribute>
        </xsl:copy>
    </xsl:template>
```

- Sequence Flow path coordinates

Sequence Flow path coordinates connecting nodes in a subprocess must be relative to the upper-left corner of the subprocess.

In your XSL, you should determine whether Sequence Flow path coordinates are relative to Subprocess or not. If not, convert them to relative to upper-left corner of the Subprocess.

Simple program logic can be used to convert coordinates to relative. Such logic would subtract the Subprocess coordinates from the Sequence Flow path coordinates. All other Flows must be absolute.

Also, in your XSL code you should determine whether coordinates are relative or absolute. If coordinates are not absolute, convert them. To convert, include templates in a style sheet that contains logic to add the parent Pool coordinates to the Flow coordinates and then assigns these values to the Flow coordinates.

It is important to note the following if coordinates are relative to the parent pool or the parent lane:

- If sequence flows are relative to the parent pool, add the pool coordinates to the flow coordinates.
- If Sequence Flows are relative to the parent Lane, add the parent Pool coordinates and the parent Lane coordinates to the Flow coordinates.

B.3.3 Handling Extended Attributes

Two extended attributes are used by Business Process Composer to parse XPDL files. These extended attributes are optional but can be very useful when working with XPDL documents that employ relative coordinates for Flows. These attributes are:

- redrawConnections
- isRelativeObjectCoordinates

These extended attributes are not standard XPDL elements but can be used to mitigate the amount of work required to convert Flow coordinates from relative to absolute or from absolute to relative. Although the <ExtendedAttributes> element is found in many elements of the XPDL document, Oracle BPM will search for it under <Package/> element. Place the <ExtendedAttributes> element under <Package/> element to ensure it is located by Oracle BPM.

B.3.4 Handling redrawConnections

Setting this attribute to true will let Oracle BPM redraw flow connections instead of using the original coordinates. This is useful when coordinates of flows need to be converted to relative or absolute. If it is hard to convert coordinates of flows to relative or absolute, then setting this attribute to true will let Oracle BPM recreate the coordinates for flows.

When this attribute is set to true, there is no need to convert coordinates to absolute or relative. Oracle BPM will not consider the original coordinates of flows but will create the most suitable coordinates to flows.

Oracle BPM will use the original coordinates when the ExtendedAttribute is not included in XPDL file or is set to false.

Therefore, use of this attribute should be considered when retaining the layout of the original model is of lesser importance than providing a shorter development time.

Example B–4 shows how to set this attribute.

Example B–4 Setting the ExtendedAttribute when Handling redrawConnections

```
<xsl:template match="xpdl:Package/xpdl:ExtendedAttributes">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xpdl:ExtendedAttribute Name="redrawConnections" Value="true"/>
                <xsl:apply-templates/>
            </xsl:copy>
        </xsl:template>
        <xsl:template match="xpdl:Package">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xsl:if test="not(child::xpdl:ExtendedAttributes)">
                    <xpdl:ExtendedAttributes>
                        <xpdl:ExtendedAttribute Name="redrawConnections"
Value="true"/>
                    </xpdl:ExtendedAttributes>
                </xsl:if>
                <xsl:apply-templates/>
            </xsl:copy>
        </xsl:template>
```

The above templates add the 'redrawConnections' ExtendedAttribute to the <ExtendedAttributes/> element as a child of < Package/>.

B.3.5 Handling isRelativeObjectCoordinates

The isRelativeObjectCoordinates extended attribute is used to notify Oracle BPM that object coordinates in the XPDL file are relative or absolute.

If the source XPDL file presents ALL object coordinates in absolute form [every coordinate is measured from the 0,0 point at the top left corner of the diagram], then this attribute should be used and its value set to false.

Setting this extended attribute to true informs Oracle BPM that all coordinates in the source XPDL file are relative in conformance with the 'Relative Coordinates' rules of Oracle BPM as specified above.

Oracle BPM by default assumes all coordinates of input document comply with the 'Relative Coordinates' rules of Oracle BPM as outlined above. When this attribute is set to true or is omitted, make sure that all coordinates meet the Relative Coordinates rules of Oracle BPM. Otherwise, include templates in your style sheet to make them conform to the Relative Coordinates rules specified in this document.

Example B–5 shows how to include the <ExtendedAttribute/> element in the <ExtendedAttributes/> element, which is a child of the <Package/> element, is given below.

Example B–5 Including the ExtendedAttribute in the ExtendedAttributes Element

```
<xsl:template match="xpdl21:Package/xpdl21:ExtendedAttributes">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xpdl21:ExtendedAttribute Name="isRelativeObjectCoordinates"
Value="false"/>
                <xsl:apply-templates/>
            </xsl:copy>
        </xsl:template>
        <xsl:template match="xpdl21:Package">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xsl:if test="not(child::xpdl21:ExtendedAttributes)">
```

```
                    <xpdl21:ExtendedAttributes>
                        <xpdl21:ExtendedAttribute
Name="isRelativeObjectCoordinates" Value="false"/>
                    </xpdl21:ExtendedAttributes>
                </xsl:if>
                <xsl:apply-templates/>
            </xsl:copy>
        </xsl:template>
```

B.3.6 Removing Invisible Elements

Oracle BPM considers all graphical elements of the source XPDL file to be visible elements, even if their visibility is set to false. Thus you may find some differences as formerly invisible elements have now become visible.

The only way to resolve this issue is to remove these elements from the XPDL with the help of some XSL style sheet templates as shown in Example B–6.

Example B–6 Removing Invisible Elements

```
<xpdl:Activity Name="ProcessGroup" Id="ProcessGroup">
  <xpdl:NodeGraphicsInfos>
                        <xpdl:NodeGraphicsInfo ToolId="XYZ" LaneId="PMCoE"
IsVisible="false">
                            <xpdl:Coordinates XCoordinate="7740.0"
YCoordinate="80.0"/>
                        </xpdl:NodeGraphicsInfo>
                    </xpdl:NodeGraphicsInfos>
   ……..
</xpdl:Activity>
```

Here the Activity's visibility is set to false, but when this model is imported into Oracle BPM, you will still see this Activity. To eliminate invisible elements include a style sheet template to remove them.

Example B–7 shows how to remove invisible activities.

Example B–7 Removing Invisible Activities

```
<xsl:template match="xpdl:Activity">
        <xsl:variable name="isVisible">
            <xsl:choose>
                <xsl:when
test="xpdl:NodeGraphicsInfos/xpdl:NodeGraphicsInfo/@IsVisible = 'false'">
                    <xsl:text>false</xsl:text>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:text>true</xsl:text>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:if test="$isVisible = 'true'">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xsl:apply-templates/>
            </xsl:copy>
        </xsl:if>
    </xsl:template>
```

B.3.7 Handling the Orientation Attribute

Orientation is an attribute found in many XPDL documents to specify the orientation of the model. This attribute must have HORIZONTAL or VERTICAL as its value. XPDL documents generated by some tools may have coordinates that identify the model as horizontal when the model is actually in vertical orientation and vice versa.

Make sure the model has correct coordinates with respect to the model orientation.

If model has mismatching coordinates, include style sheet templates to swap the coordinates.

The orientation attribute of model is also important to Oracle BPM for calculating dimensions of pools and lanes (which will be discussed later in this document). The orientation attribute can be found in Pool elements. If this attribute is not found in the XPDL document then the model is assumed to be in Horizontal orientation by Oracle BPM.

If the orientation attribute is in a tool specific namespace it must be made available to Oracle BPM by creating an orientation attribute on the pool elements and setting their values to the one found in tool specific namespace.

Example B–8 shows how to set orientation of pools. This template tries to find the orientation of pools. If it is not found then this template will set the orientation attribute to HORIZONTAL.

Example B–8 Setting the Orientation of Pools

```
<xsl:template match="xpdl:Pool">
        <xsl:variable name="orientation">
            <xsl:choose>
                <xsl:when test="not(@Orientation)">
                    <xsl:text>HORIZONTAL</xsl:text>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="@Orientation"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:copy>
            <xsl:copy-of select="@*"/>
            <xsl:attribute name="Orientation">
                <xsl:value-of select="$orientation"/>
            </xsl:attribute>
            <xsl:apply-templates/>
        </xsl:copy>
    </xsl:template>
```

B.3.8 Specifying the View Type for Subprocesses

The view attribute of <BlockActivity/> and <Subflow/> elements specify whether the subprocess is an expanded or a collapsed subprocess. Oracle BPM assumes the default view type of subprocess elements is COLLAPSED. For this reason, if a model has an expanded subprocess element but its' view' type is not given in XPDL file, Oracle BPM will render this element as a collapsed subprocess element.

Note: Subflow elements of an XPDL file cannot be rendered with their child elements into Oracle BPM, only the subflow elements themselves will be found in the imported model. It is possible to identify the Subprocess according to its view type. If the subflow is expanded Oracle BPM will render this element as an expanded subflow element.

The following example shows <BlockActivity/> and <Subflow/> elements that do not have a view type specified. Their corresponding interpretation by Oracle BPM will result in a collapsed subprocess.

```
<xpdl2:BlockActivity ActivitySetId="_gJ5DQeE3Ed6tmt0cZVxmlA"/>
<xpdl2:SubFlow Id="_swKUEGyzEd6oxIP3ZfQL-g" PackageRef="ProcessPackage"/>
```

If the view attribute is not found in <BlockActivity> element, it may be present in a tool specific namespace. If so, include templates in XSL to create view attribute on <BlockActivity> or <Subflow> elements.

Example B–9 shows a style sheet used to add the view attribute to <Subflow> elements. This template will work for BizAgi generated XPDL files.

Example B–9 Adding the View Attribute to a Subflow Element

```
<xsl:template match="xpdl21:SubFlow">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xsl:attribute name="View">
                    <xsl:choose>
                        <xsl:when
test="ancestor::xpdl21:Activity/xpdl21:NodeGraphicsInfos/xpdl21:NodeGraphicsInfo/@
Expanded='false'">
                            <xsl:text>COLLAPSED</xsl:text>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:text>EXPANDED</xsl:text>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:attribute>
            </xsl:copy>
        </xsl:template>
```

Example B–10 shows a sample style sheet template that adds the View attribute to <BlockActivity> elements. This template will work for Oracle BPM Studio XPDL generated files. This template also shows how to create the View attribute on <BlockActivity> elements by accessing the View attribute value from a tool specific namespace.

Example B–10 Adding the View Attribute to BlockActivity Elements

```
<xsl:template match="xpdl:Activity/xpdl:BlockActivity">
  <xsl:copy>
   <xsl:copy-of select="@*"/>
   <xsl:choose>
    <xsl:when
test="ancestor::xpdl:Activity/xpdl:Extensions/albpm:ALBPMExtensions/albpm:FeatureS
et/albpm:BooleanFeature[@ name='collapsed']/@value='true'">
     <xsl:attribute name="View">
      <xsl:text>COLLAPSED</xsl:text>
```

```
     </xsl:attribute>
    </xsl:when>
    <xsl:otherwise>
        <xsl:attribute name="View">
  <xsl:text>EXPANDED</xsl:text>
</xsl:attribute>
    </xsl:otherwise>
   </xsl:choose>
   <xsl:apply-templates/>
  </xsl:copy>
 </xsl:template>
```

B.3.9 Handling the Object Pin

In some tools the coordinates provided for activities are given for the upper-left corner of the activity object, but for others the coordinates are based on the center of the activity object. These coordinates serve as the object pin.

When an XPDL document containing an object pin at center of activities is imported into Oracle BPM, the activities may occur at different positions than expected. This is because the object pin for activities is located in the center but Oracle BPM expects the object pin in the upper left corner. To resolve this discrepancy the activity coordinates must be recalculated.

You can use simple logic to calculate these coordinates, such as subtracting half of the activity's width from its XCoordinate and subtracting half of the activity's height from its YCoordinate.

Example B–11 shows a style sheet template that does this recalculation. Note that this template will work only if width, height, x, and y-coordinates are provided for activities.

Example B–11 Recalculating the Location of an Object Pin

```
<xsl:template match =
"xpdl:Activity/xpdl:NodeGraphicsInfos/xpdl:NodeGraphicsInfo/xpdl:Coordinates">
<xsl:copy>
            <xsl:copy-of select = "@*[name() != 'XCoordinate' and name() !=
'YCoordinate']"/>
            <xsl:attribute name = "XCoordinate">
                <xsl:value-of select = "@XCoordinate -
ancestor::xpdl:NodeGraphicsInfo/@Width div 2 "/>
            </xsl:attribute>
            <xsl:attribute name="YCoordinate">
                <xsl:value-of select = "@YCoordinate -
ancestor::xpdl:NodeGraphicsInfo/@Height div 2"/>
            </xsl:attribute>
            <xsl:apply-templates/>
        </xsl:copy>
    </xsl:template>
```

B.3.10 Modifying the Height and Width of Activities

Some tools do not provide height and width for activities in their XPDL files. But coordinates and dimensions of activities are needed to import an XPDL file into Oracle BPM. So include templates in an XSL style sheet that will set height and width for activities and lanes. If the XPDL file does not contain height and width for activities, set some default dimensions for them. For instance, set 80x40 width and height for tasks, collapsed subprocesses, and set 30x30 width and height to events and gateways.

It can be especially difficult to set the height and width for expanded <BlockActivity> and <Subflow> elements, as an expanded <BlockActivity> element may also contain another expanded <BlockActivity> elements. Here the innermost BlockActivity height and width should be calculated first and then its parent BlockActivity. This recursion must bubble up to the topmost <BlockActivity>. It can be difficult to code this recursion process in XSL. For this reason Oracle BPM provides a feature which calculates height and width of expanded Subprocesses. To use this feature set the height and width of <BlockActivities> to 0,0 using XSL templates.

Example B–12 is an event element which does not have width and height defined.

Example B–12 Event Element that Does Not Contain Height and Width Defined

```
<xpdl:Activity Name="Begin" Id="Begin">
                    <xpdl:Event>
<xpdl:StartEvent Trigger="None"/>
                    </xpdl:Event>
   ……..
      <xpdl:NodeGraphicsInfos>
                        <xpdl:NodeGraphicsInfo LaneId="Accounting"
IsVisible="true">
                            <xpdl:Coordinates XCoordinate="36.0"
YCoordinate="110.0"/>
                        </xpdl:NodeGraphicsInfo>
                    </xpdl:NodeGraphicsInfos>
 </xpdl:Activity>
```

Example B–13 shows a sample style sheet template used to set height and width of activities. This template can be used if height and width of activities are not given in the XPDL document. This example sets 80 x 40 dimensions (width, height) for task elements and collapsed BlockActivities, it will set 30 x 30 dimensions to route and gateways, and it will set 0x0 dimensions to expanded BlockActivities and thus lets Oracle BPM calculate the dimensions for these expanded BlockActivity elements. This template will work for BPM studio XPDL files.

Example B–13 Setting the Height and Width of Activities

```
<xsl:template match =
"xpdl:Activity/xpdl:NodeGraphicsInfos/xpdl:NodeGraphicsInfo">
        <xsl:variable name = "activityType">
            <xsl:choose>
<xsl:when test = "ancestor::xpdl:Activity/xpdl:Implementation/xpdl:SubFlow">
                    <xsl:text>SubFlow</xsl:text>
                </xsl:when>
                <xsl:when test = "ancestor::xpdl:Activity/xpdl:Implementation">
                    <xsl:text>Task</xsl:text>
                </xsl:when>
                <xsl:when test = "ancestor::xpdl:Activity/xpdl:Event">
                    <xsl:text>Event</xsl:text>
                </xsl:when>
                <xsl:when test = "ancestor::xpdl:Activity/xpdl:Route">
                    <xsl:text>Route</xsl:text>
                </xsl:when>
                <xsl:when test = "ancestor::xpdl:Activity/xpdl:BlockActivity">
                    <xsl:choose>
                        <xsl:when test =
"ancestor::xpdl:Activity/xpdl:Extensions/albpm:ALBPMExtensions/albpm:FeatureSet/al
bpm:BooleanFeature[@ name = 'collapsed']/@value != 'true'">
                            <xsl:text>ExpandedBlockActivity</xsl:text>
                        </xsl:when>
```

```
                        <xsl:otherwise>
                            <xsl:text>CollapsedBlockActivity</xsl:text>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:when>
            </xsl:choose>
        </xsl:variable>

        <xsl:copy>
          <xsl:copy-of select = "@*"/>
              <xsl:attribute name = "Width">
                  <xsl:choose>
                      <xsl:when test = "$activityType = 'Task' or $activityType =
'CollapsedBlockActivity' or $activityType = 'SubFlow'">
                          <xsl:text>80</xsl:text>
                      </xsl:when>
                      <xsl:when test = "$activityType = 'Route' or $activityType =
'Event'">
                          <xsl:text>30</xsl:text>
                      </xsl:when>
                      <xsl:when test = "$activityType = 'ExpandedBlockActivity'">
                          <xsl:text>0</xsl:text>
                      </xsl:when>
                  </xsl:choose>
              </xsl:attribute>
              <xsl:attribute name = "Height">
                  <xsl:choose>
                      <xsl:when test = "$activityType = 'Task' or $activityType =
'CollapsedBlockActivity' or $activityType = 'SubFlow'">
                          <xsl:text>40</xsl:text>
                      </xsl:when>
                      <xsl:when test = "$activityType = 'Route' or $activityType =
'Event'">
                          <xsl:text>30</xsl:text>
                      </xsl:when>
                      <xsl:when test = "$activityType = 'ExpandedBlockActivity'">
                          <xsl:text>0</xsl:text>
                      </xsl:when>
                  </xsl:choose>
              </xsl:attribute>
          <xsl:apply-templates/>
        </xsl:copy>
    </xsl:template>
```

B.3.11 Modifying the Height and Width of Lanes

In many XPDL documents, width or height dimensions for lanes is provided depending on orientation of the parent pool. For instance, if the orientation of the parent pool is horizontal then the width of the Lane might be found in the XPDL document but not its height. As mentioned before, the height and width of lanes and activities are needed for Oracle BPM to determine the sizes of graphical elements. If the lane height or width is not found in the XPDL document, set these attributes to a value which is sufficient to hold all its containing elements.

If the height of lanes is provided but not their width, simple logic can be used to set the width of lanes. Find the largest sum of x-coordinate plus the widths of Activities, and set this value to the widths of all lanes. If the lane width is merely sufficient to hold all elements, you can find the lane right border running from the right border of activity that has the largest sum of x-coordinate and width. A small padding value can

also be added to the largest sum to extend the lanes enough to allow them to appear as the containers of the other objects.

The above logic will work if all activities contain width and height values. But there are cases where activities might not contain height and width values. In such cases it is difficult to calculate lane widths using XSLT with the above logic as each activity height and width should be calculated before calculating lane height or width. To resolve this problem, Oracle BPM provides a feature that will set the lane widths or heights.

To use this feature set the missing dimension of lane to zero using XSLT. This feature assumes one dimension was provided for the lane.

B.3.12 Modifying the Height and Width of Pools

If height and widths are provided for pools, Oracle BPM will use those dimensions for Pools. If these values are absent, Oracle BPM will try to calculate them. Oracle BPM will calculate both dimensions even if one of dimensions is provided. The source XPDL document must contain both dimensions for Pools to circumvent this feature

B.3.13 Location of Activities

When coordinates and dimensions are provided in the XPDL file, Oracle BPM will use those values without manipulating them. But if some dimensions are missing, such as dimensions for Lanes dimensions or Subprocesses, Oracle BPM will try to calculate these dimensions. In the process of calculating dimensions, Oracle BPM will add some padding to the dimensions for the model to be friendlier but this model will look good only if the activities have some space around them. Otherwise the imported model may have borders of activities or lanes running on top of other activities or lanes. To avoid this problem position activities with some space around them.

Problems may occur when importing models containing duplicate ids for more than one lane or activity into Oracle BPM. Oracle BPM cannot create more than one lane or activity with the same name. Only one lane or activity from the source will be created, and the results will not accurately reflect the original model.

To avoid this problem, create model elements with unique ids.

B.3.14 Including Missing Elements

If the XPDL source document is missing elements or attributes required for Oracle BPM to properly perform its conversion, then add those elements and attributes using XSLT.

For instance, there are 8 types of tasks in XPDL. For Oracle BPM to recognize these task types, a <Task> element should contain another child element which specifies whether the Task is a service task, a receive task, and so on. If these child elements are not found under a source <Task> element, that <Task> element will be converted to a default <Task> element.

For example, the <Task> element shown below is a user task but does not have a child <TaskUser> element. Hence it is assumed to be a default Task element by Oracle BPM

```
<Implementation>
            <Task />
 </Implementation>
```

In order for the activity to be identified by Oracle BPM as user task, a <TaskUser> element must be added under Task element as follows:

```
<Implementation>
<Task>
        <TaskUser>
....
                    </TaskUser>
</Task>
</Implementation>
```

As mentioned before many attributes may be given under tool specific namespaces. If task type is not found under the XPDL namespace, try finding it in the tool specific namespace and include templates in the style sheet to include these elements under the Task element.

Example B–14 demonstrates how to include a <TaskService> element as a child of <Task> element whenever it finds an "Automatic" Task element. This template will work for XPDL source files generated by Oracle BPM Studio.

Example B–14 Including a TaskService Element as a Child of the Task Element

```
<xsl:template match="xpdl:Activity/xpdl:Implementation/xpdl:Task">
  <xsl:copy>
   <xsl:copy-of select="@*"/>
       <xsl:if
test="ancestor::xpdl:Activity/xpdl:Extensions/albpm:ALBPMExtensions/albpm:FeatureS
et/albpm:StringFeature[@name='type']/@value='AUTOMATIC'
and not(child::xpdl:TaskService)">
     <xpdl:TaskService/>
    </xsl:if>
     <xsl:apply-templates/>
  </xsl:copy>
 </xsl:template>
```

B.3.15 Checking the Correctness of Activities

Be sure that the source XPDL contains the correct elements to represent activities when they are exported as XPDL from the source tool. For example, if a model contains an event activity and while exporting that model into XPDL the tool creates <Route> or <Task> element for that Event Activity, this element must be replaced with an element that accurately represents an Event Activity.

Consider the XPDL element shown below. The activity is a start event, but instead of an event element under the <Activity> element, you find a <Route> element. When this model is imported into Oracle BPM, this activity is converted into a route activity rather than as an event activity.

Example B–15 Checking the Correctness of Activities

```
<xpdl:Activity Name="Group$Begin" Id="Group$Begin">
                            <xpdl:Route GatewayType="XOR" MarkerVisible="true"/>
……….
</xpdl:Activitiy>
The correct notation for the above element should be:
<xpdl:Activity Name="Group$Begin" Id="Group$Begin">
<xpdl:Event>
                        <xpdl:StartEvent Trigger="None"/>
        ……..
                </xpdl:Event>
    ……..
</xpdl:Activity>
```

If this problem is found in your source XPDL, include templates in a style sheet to replace the incorrect elements with correct elements.