Oracle® Fusion Middleware

Modeling and Implementation Guide for Oracle Business Process Management

11*g* Release 1 (11.1.1.5.0)

E15176-05

April 2011

Describes how to design and implement business processes using Oracle Business Process Studio.

ORACLE

Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management, 11*g* Release 1 (11.1.1.5.0)

E15176-05

Primary Author: Carolina Arce Terceros, Steven Leslie

Contents

2 Overview of Business Process Design

3 Introduction to Oracle BPM Studio

Part III Modeling a Process

6 Modeling Business Processes with Oracle BPM

7 Modeling Your Organization

8 Handling Information in Your Process Design

Part V Working with Business Components

12 Using the Business Catalog

13 Modeling Business Objects

14 Using Human Tasks

18 Adding Delays, Deadlines, and Time Based Cycles to Your Process

19 Handling Errors

20 Communicating With Other BPMN Processes and Services

21 Defining the Process Interface

22 Writing Expressions

23 Using SOA Composites with BPM Projects

24 Working with Guided Business Processes

25 Building a Guided Business Process Client Application

26 Using Approval Management

Preface

Welcome to *Modeling and Implementation Guide for Oracle Business Process Management*. This document describes how to use Oracle Business Process Studio.

Audience

This guide is intended for process developers who use the Business Process Studio application to create and implement business processes, and create and configure Oracle BPM projects used to created process-based applications using the Oracle Business Process Management Suite.

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

Documentation Accessibility

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

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html 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 Business Process Composer User's Guide for Oracle Business Process Management*
- *Oracle Fusion Middleware User's Guide for Oracle Business Process Management*

Oracle SOA Suite

- *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*
- *Oracle Fusion Middleware User's Guide for Technology Adapters*
- *Oracle Fusion Middleware User's Guide for Oracle Business Rules*

Oracle SOA and BPM Suite Installation and Administration

- *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*
- *Oracle Fusion Middleware Quick Installation Guide for Oracle SOA Suite and Oracle Business Process Management Suite*
- *Oracle Fusion Middleware High Availability Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Introduction to Oracle BPM Studio

This part provides a general introduction to the Oracle BPM Studio. It also provides an overview of the Oracle BPM Suite and shows how BPM Studio 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 Oracle 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.

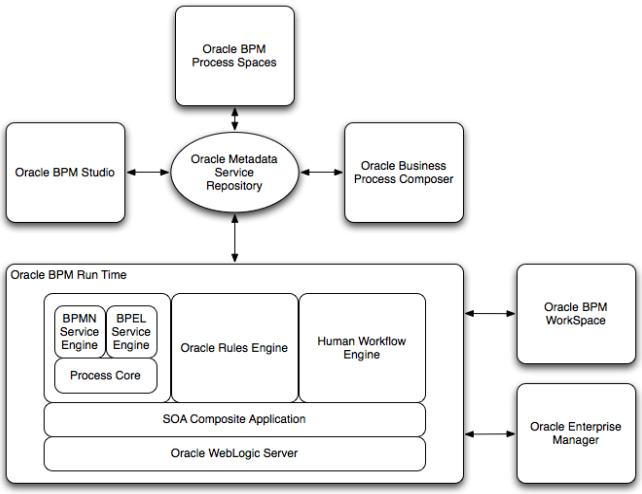
The Oracle BPM Suite provides a seamless integration of all stages of the application development life cycle from design-time and implementation to run-time and application management.

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 the 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 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 Oracle BPM 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 Oracle BPM 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 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 Process Blueprints. These are initial drafts of a process that can be used by process developers who use Oracle BPM Studio to add further implementation details and refinement to the project.

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 the *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* 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 enable dynamic decisions at runtime allowing you to automate policies, computations, and reasoning while separating decision logic from underlying process orchestration layer. This allows more agile rule maintenance and empowers business analysts with the ability to modify rule logic without programmer assistance and without interrupting business processes.

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 BPM WorkSpace

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

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

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

1.3.3.2 Oracle BPM Process Spaces

Oracle Process Spaces is a collaborative workspace built on top of Web Center Spaces and enables more productivity by increasing collaboration.

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

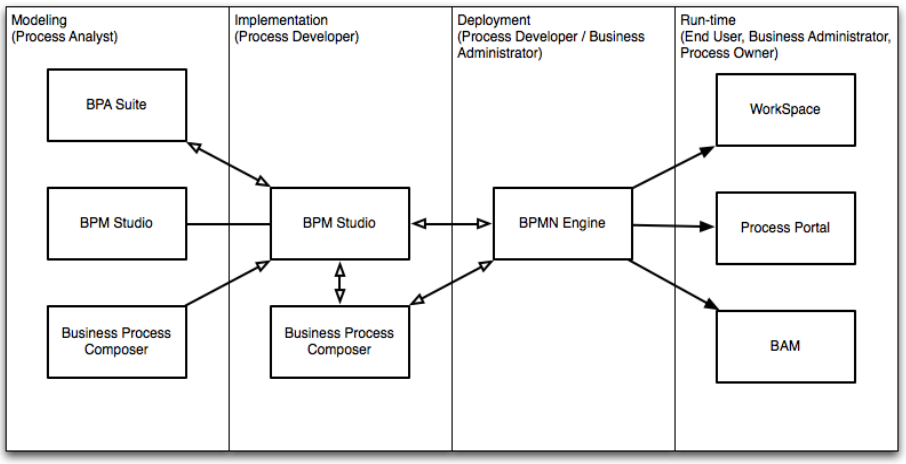
See the Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management for more information on using business processes created in Oracle BPA within Oracle BPM Studio.

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 brought inside 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 Oracle Business Process Composer

  Oracle BPM enables you to deploy projects created from project templates directly to the same run-time environment where Business Process Composer is installed. When creating a project, 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 Oracle BPM 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 Oracle BPM 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 Oracle BPM Workspace or Oracle 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 simulations 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 Process Blueprints

This use case involves creating process blueprints using Business Process Composer. These blueprints 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 Process Blueprints

1. Create process blueprints using Business Process Composer. (process analyst)
2. Provide implementation details for the business process 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 BPA 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 BPA 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, 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 of Business Process Management (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:

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

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 support BPMN 2.0.

For more information on BPMN see: `http://www.bpmn.org`.

2.1.2 Business Processes

A business process can be generally defined as a sequence of tasks that after it performed result in a well-defined outcome. As the term business 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 task 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 business process can generally define 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.

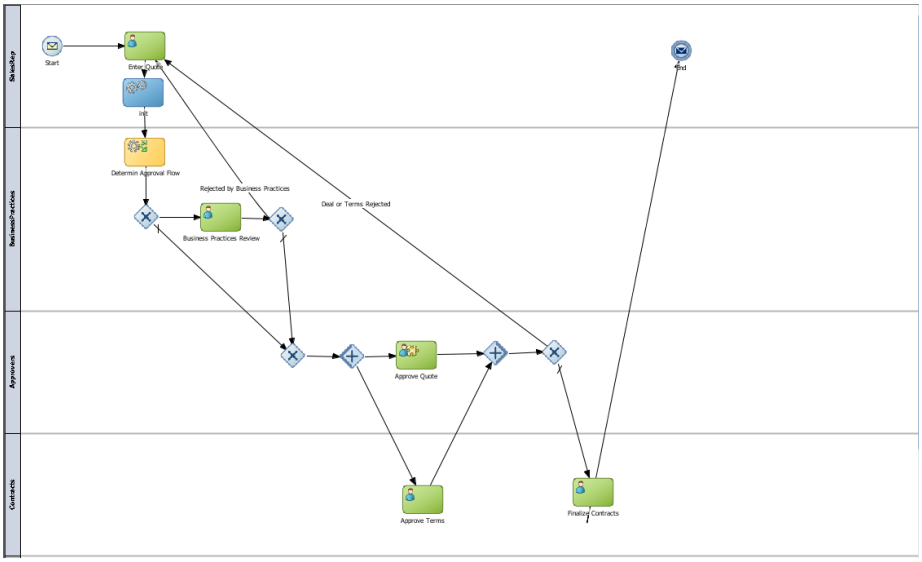
For example, the Sales Quote example defines several data objects used to store information about the sales quote. At run time, the process instance generate and stores specific values for these variables.

2.2 Introduction to the Sales Quote Example Project

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 demo project is shown in Figure 2–1.

Figure 2–1 The Sales Quote Example Project



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:

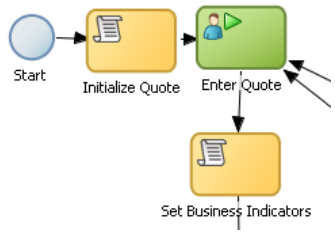
- Section 2.2.1.1, "Initiate Sales Quote"
- Section 2.2.1.2, "Determine Business Practice Review"
- Section 2.2.1.3, "Approve Quote"
- Section 2.2.1.4, "Approvals Outcome"

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:

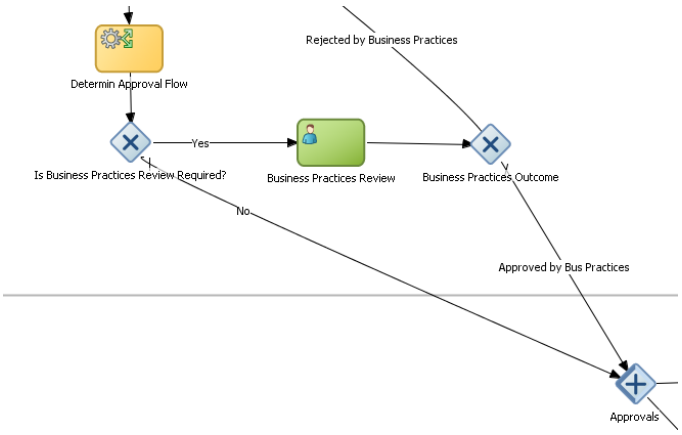
1. Define start point (none start event)
2. Initialize Data objects (script task)
3. Initiate the process instance (user task with initiator pattern)
4. Set values for the business indicator data objects (script task)

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), then
   - 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 define 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.

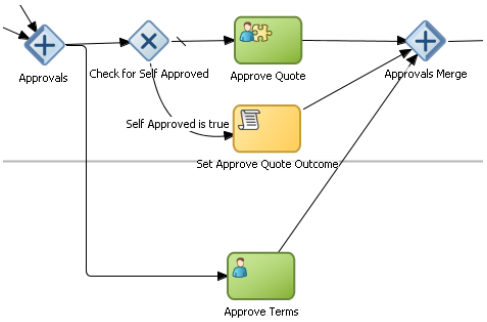
  This process path is also split into two paths. In this example, it is possible for the quote to be self approved, which means that the quote is approved based on certain criteria, or it may require explicit approval from a process participant.

- Approve the terms of the 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, wo 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)

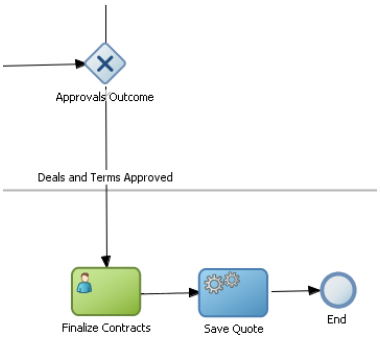
2.2.1.4 Approvals Outcome

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.

****

Approvals 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 BPM Studio

This chapter provides a general introduction to Oracle BPM Studio and describes how it is used within the Oracle BPM Suite.

This chapter includes the following sections:

3.1 Overview of Oracle BPM Studio

Oracle BPM Studio is a component of the Oracle BPM Suite that enables process developers to create process-based applications. It also enables process analysts and developers to model business processes.

Oracle BPM Studio is part of the Oracle JDeveloper IDE. and shares many of the JDeveloper user interface elements used by the Oracle SOA Suite.

3.1.1 Oracle BPM Studio Use Cases

There are three typical use cases for Oracle BPM Studio:

- Develop process-based business applications based on process models created using the Oracle BPA Suite or Oracle Business Process Composer.
- Create process models and implement them as process-based business applications.
- Create project templates that are used in Business Process Composer to create running business applications.

See Section 3.2 for information on how these use cases fit into the general workflow of the application development life cycle.

3.1.2 Introduction to JDeveloper Roles

The JDeveloper environment can be tailored based on the role selected by the user. The modified environment removes unneeded items from JDeveloper, including menus, preferences, New Gallery, and even individual fields on dialogs. The JDeveloper role you select determines which technologies and options are available to you as you work in JDeveloper.

Oracle BPM includes the Analyst Role which only includes process design elements that are useful to business analysts.

Process developers who need access to the complete functionality of the Oracle BPM and SOA Suites should use the default role.

3.2 Overview of the Application Development Life Cycle

This section describes the application development life cycle from the perspective of a process developer using Oracle BPM Studio.

3.2.1 Introduction to Modeling, Implementation, and Deployment

The different workflows of the application development cycle described in the following sections are divided into the following stages:

- Modeling
- Implementation
- Deployment

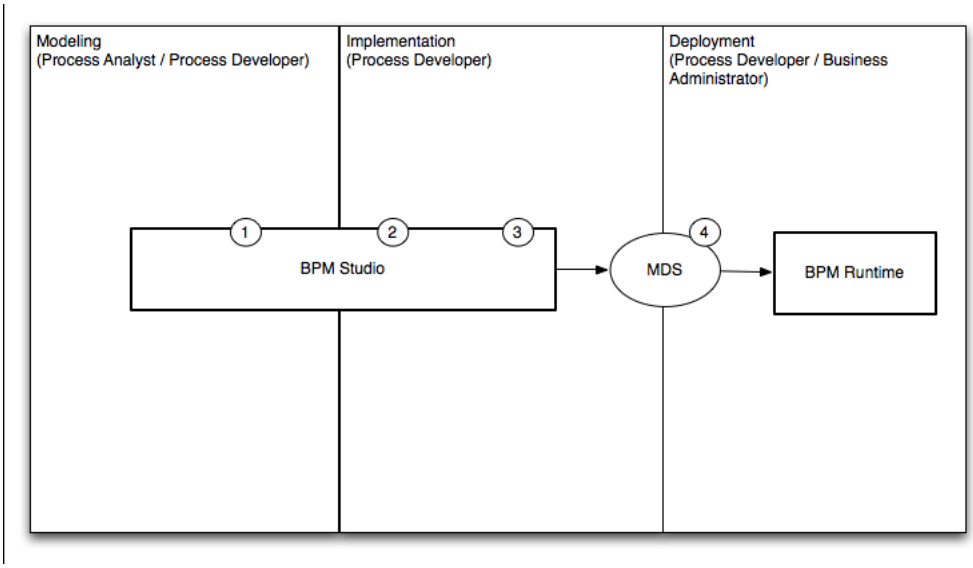
In a real-world application development environment, the distinctions between these stages may not be clearly defined. Process developers may need to add to or change a process when creating an application.

A final production application may go through several iterations of modeling and implementation before it is deployed as a working application. Additionally, applications may be deployed for testing then passed back to the modeling and implementation stages before being deployed to a production environment.

3.2.2 Workflow: Modeling, Implementing, and Deploying an Application

Figure 3–1 shows a typical workflow where all stages of application design are performed using Oracle BPM Studio.

Figure 3–1 Modeling, Implementation, and Deployment from Studio



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

The Modeling (Process Analyst) section contains a rectangle with the number 1 and labeled Oracle BPA Suite. From this rectangle, an arrow numbered 2 extends into the Implementation section to a rectangle numbered 3 and 4, and labeled BPM Studio.

From this BPM Studio 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 5a 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 5b and labeled SAR File.

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

****

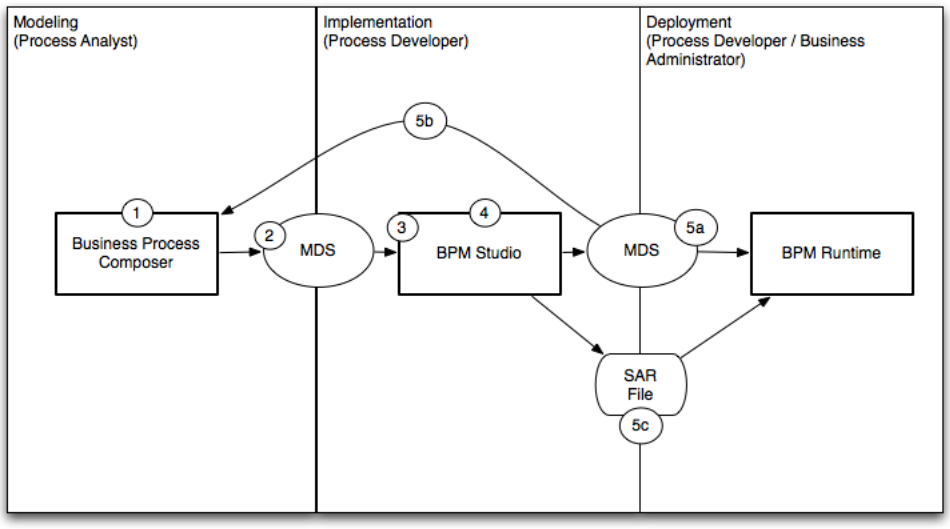
The following steps describe each stage of the workflow:

1. Create process models using Oracle BPM Studio (process analyst / process developer).
2. Implement the required services and application resources (process developer).
3. Compile the application (process developer).
4. Deploy to Oracle BPM runtime (process developer / process administrator).

3.2.3 Workflow: Creating Applications Based on Process Blueprints

Figure 3–1 shows a typical workflow for using Business Process Composer to perform the initial stages of the application development life-cycle.

Figure 3–2 Using Oracle Business Process Composer to Create Project Blueprints



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.

****

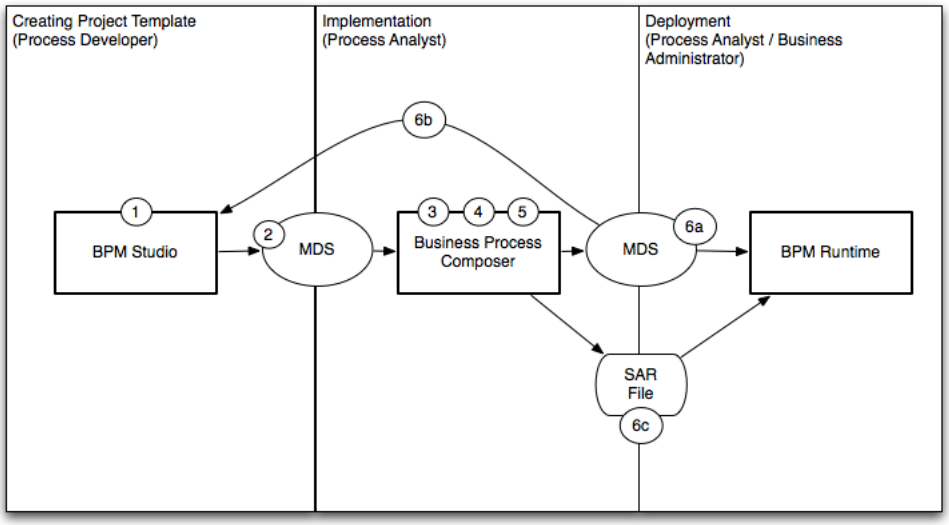
The following steps describe each stage of this workflow:

1. Create blueprints using Business Process Composer (process analyst)
2. Publish the project containing the process blueprint to MDS (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 runtime or save as a project template
   a. Deploy the process to runtime (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)

3.2.4 Workflow: Creating Project Templates

Figure 3–1 shows a typical workflow for using Oracle BPM Studio at the beginning of the workflow to create process templates which are then edited by process analysts using Business Process Composer.

Figure 3–3 Using BPM Studio to Create Project Templates



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

In the Modeling section, there is a rectangle numbered 1 and Business Process Composer.

From this rectangle, an arrow extends to an oval on the divider between the Modeling and Implementation sections. The oval is numbered 2 and labeled MDS.

From the oval labeled MDS, an arrow extends to a rectangle in the Implementation section numbered 3 and 4 and labeled BPM Studio.

From the BPM Studio rectangle, an arrow extends to an oval on the divider between Implementation and Deployment. This oval is numbered 5a and labeled MDS.

From this second MDS oval, two arrows extend. The first arrow, numbered 5b, extends back to the Business Process Composer rectangle in the Modeling section.

The second arrow extends to a rectangle in the Deployment section labeled BPM Runtime.

****

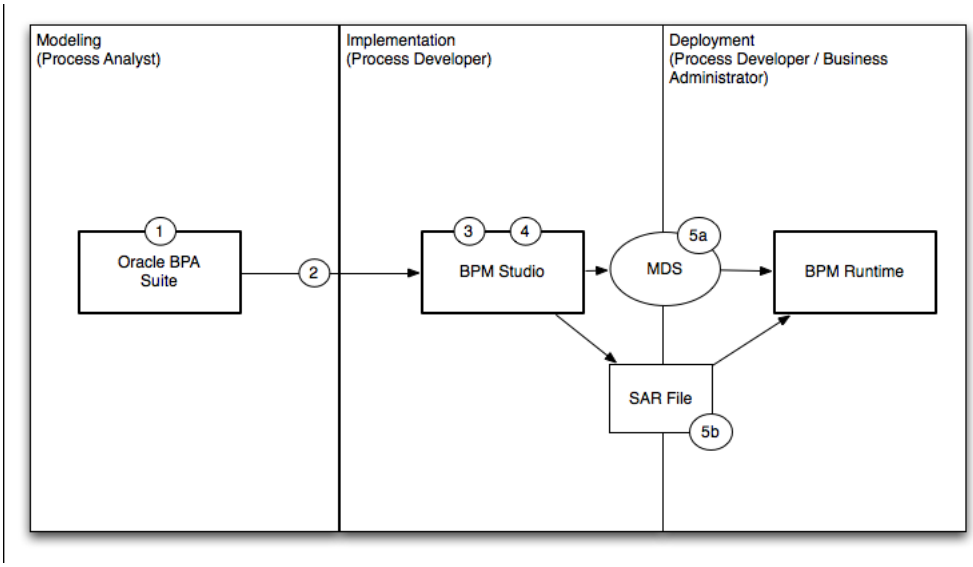
The following steps describe each stage of this workflow:

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

3.2.5 Workflow: Integration between the Oracle BPM Suite and Oracle BPA

Figure 3–4 shows a typical workflow for creating process models using the Oracle BPA Suite, then using Oracle BPM Studio to create and deploy process-based business applications.

Figure 3–4 Using Studio to Implement Processes from Oracle BPA



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

In the Modeling section, there is a rectangle numbered 1 and BPM Studio.

From this rectangle, an arrow extends to an oval on the divider between the Modeling and Implementation sections. The oval is numbered 2 and labeled MDS.

From the oval labeled MDS, an arrow extends to a rectangle in the Implementation section numbered 3 and 4 and labeled Business Process Composer.

From the Business Process Composer rectangle, an arrow extends to an oval on the divider between Implementation and Deployment. This oval is numbered 5a and labeled MDS.

From this second MDS oval, two arrows extend. The first arrow, numbered 5b, extends back to the Business Process Composer rectangle in the Modeling section.

The second arrow extends to a rectangle in the Deployment section labeled BPM Runtime.

****

The following steps describe each stage of the workflow:

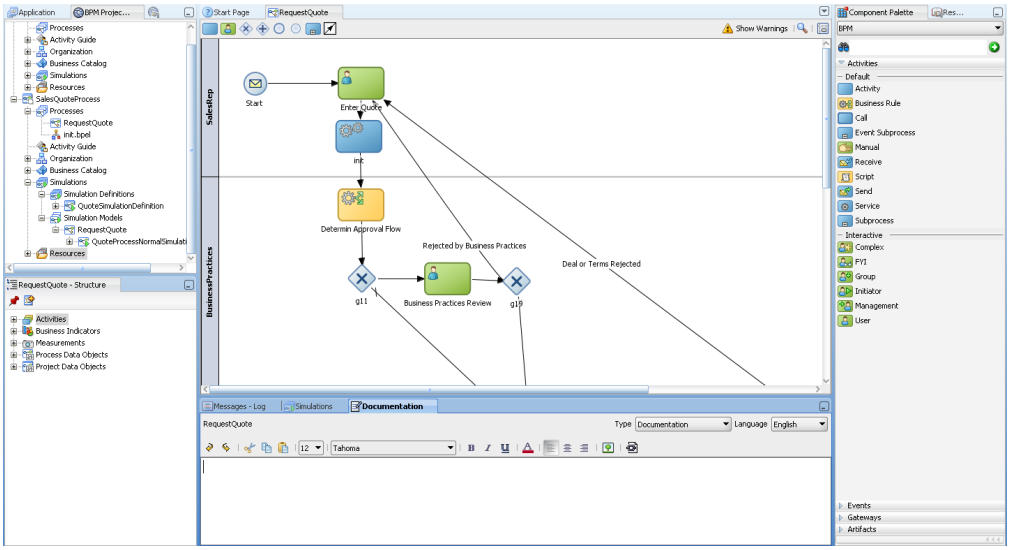
1. Create process models using the Oracle BPA suite (process analyst)
2. Import process models to Oracle BPM Studio (process developer)
3. Implement (process developer)
4. Compile (process developer)
5. Deploy to Oracle BPM runtime (process developer / process administrator)
   a. Deploy directly from Oracle BPM Studio using MDS.
   b. Export the project as a SAR file that can be imported to BPM runtime using Oracle Enterprise Manager

3.3 Introduction to the Oracle BPM Studio User Interface

Since Oracle BPM Studio is an integrated part of Oracle JDeveloper, the user interface uses many of the same components as other Oracle products. This section describes the various UI components used by Oracle BPM Studio.

Figure 3–5 shows the layout of Oracle BPM Studio displaying the Request Quote example process.

Figure 3–5 Oracle BPM Studio



This illustration is described in the text.

****

3.3.1 Oracle BPM Project Navigator

The Oracle BPM Project Navigator displays a hierarchical view of the components of an Oracle BPM project. The components displayed in the navigator are related to the modeling and implementation of business processes.

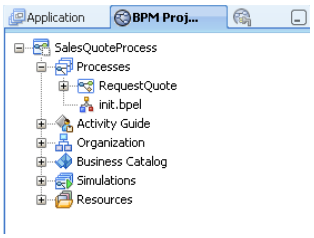
See Chapter 4, "Working with Projects and Project Templates" for more information.

Table 3–1 Oracle BPM Project Components

Component	Description
Processes	Contains the business processes of this project. These can include both BPMN and BPEL processes.
Activity Guide	Contains the milestones defined for this project.
Organization	Contains the organizational elements defined for this project.
Business Catalog	Contains the business catalog elements defined for this project.
Simulations	Contains the process and project simulation models defined for this project.
Resources	Contains the XSLT transformations of the project.

Figure 3–6 shows the Project Navigator displaying the contents of the Request Quote demo process.

Figure 3–6 Project Navigator



This graphic shows an expansion of the SalesQuoteProcess in the BPM Project Navigator.

The elements in this process are processes, Activity Guide, Organization, Business Catalog, Simulations, and resources.

The Processes element is expanded to display the items RequestQuote and init.bpel.

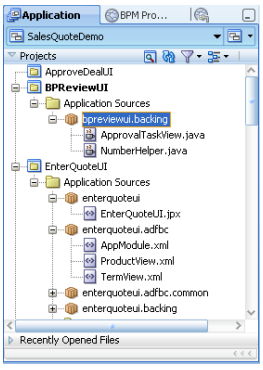
****

3.3.2 Application Navigator

Like the Project Navigator, the Application Navigator displays a hierarchical view of the components of a project. However, these are lower-level components that include the underlying configuration files, XML files, java classes, and other resources used by a SOA composite application.

Figure 3–7 shows some of the files of the Sales Quote example that appear in the application navigator.

Figure 3–7 Application Navigator



This illustration is described in the text.

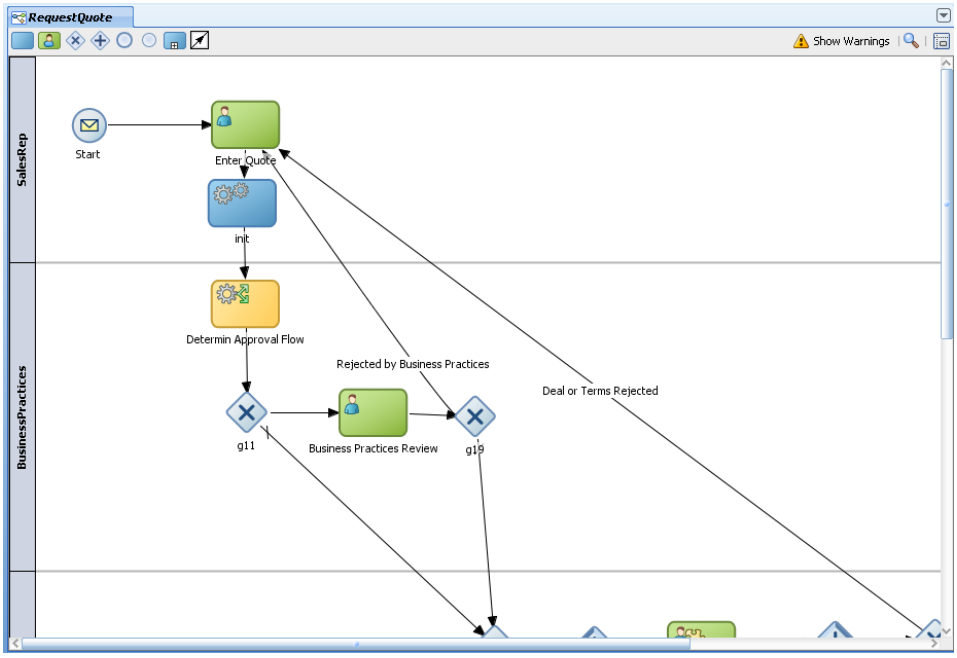
****

See the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite* for more information about the contents of a SOA composite application.

3.3.3 BPMN Process Editor

The process editor enables you to model business processes by dragging and dropping BPMN components, called flow objects, from the component palette.

Figure 3–8 Process Editor



This graphic shows the BPMN for as it appears in Oracle BPM Studio for the sample Request Quote project.

****

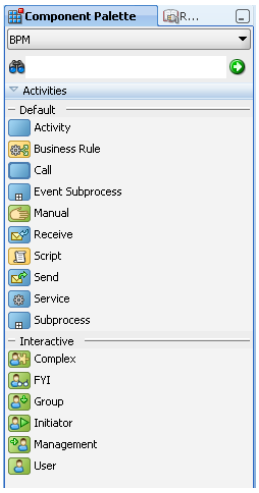
3.3.4 BPMN Component Palette

The BPM Component Palette contains a list of the BPMN flow objects supported by Oracle BPM. You can model business processes by dragging and dropping these flow objects from the BPM Component Palette to the Process Editor.

The BPMN flow objects are divided according to type. See Chapter 6, "Modeling Business Processes with Oracle BPM" for more information on using flow objects to model business processes.

Figure 3–9 shows the BPM Component Palette expanded to show activities.

Figure 3–9 Component Palette



This graphic of the BPM Component palette shows two types of activities: Default and Interactive.

The default activities are: Activity, business rule, call, event subprocess, manual, receive, script, send, service, and subprocess.

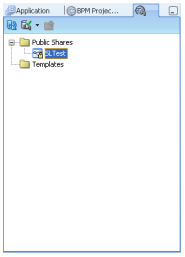
The interactive activities are: complex, fyi, group, initiator, management, and user.

****

3.3.5 Oracle BPM MDS Browser

The Oracle BPM MDS Browser allows you to view and use projects and project templates stored in the Oracle BPM MDS repository. See Section 4.5, "Using the Oracle BPM Metadata Service (MDS) Repository" for more information.

Figure 3–10 Oracle BPM MDS Browser



This graphic shows an example of the Oracle BPM MDS browser.

There are two folders: Public Shares and Templates.

The Public folder is expanded.

****

3.3.6 Structure View

The Structure window offers a structural view of the data in the component currently selected in the active window of those windows that participate in providing structure: the diagrams, the navigators, the editors and viewers, and the Property Inspector.

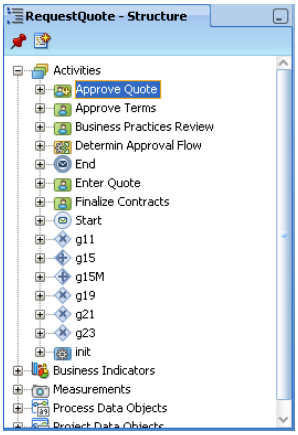
The components displayed in the Structure window are usually components in the Project or Application Navigators.

You can perform a variety of tasks from this window, including:

- Edit process properties
- Configure an activity guide and create new milestones
- Convert a BPMN Process to BPEL
- Create business objects, modules, and business exceptions
- Create new simulation models.

Figure 3–11 shows the Structure View when a BPMN process is selected in the Project Navigator.

Figure 3–11 Structure View



This graphic shows the structure view panel for the sample process RequestQuote.

The uppermost element is Activities, which is expanded to display its contents: Approve Quote, Approve Terms, Business practices Review, Determine Approval Flow, End, Enter Quote, Finalize Contracts, Start, g11, g15, g15M, g19, g21, g23, and init.

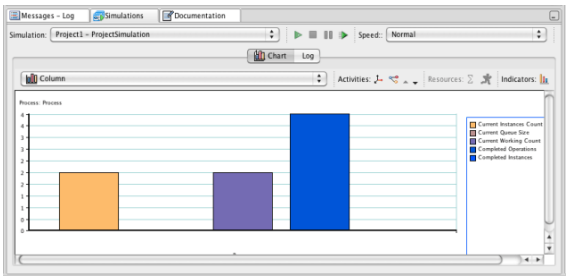
The other elements that are visible are Business Indicators, Measurements, Process Data Objects, and Project data Objects.

****

3.3.7 Simulation View

The Simulation View allows you to run and see the result of project simulation models. Figure 3–12, shows the results of a simulation displayed as a bar chart.

Figure 3–12 Simulation View



This illustration is described in the text.

****

.

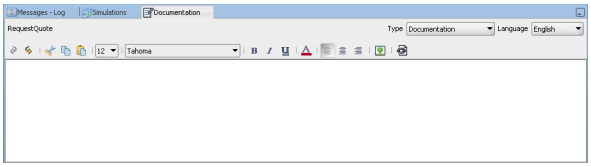
3.3.8 Log Window

The Log Window displays messages, errors, and warnings to the BPM project as well as compilation and deployment of SOA composite applications.

3.3.9 Documentation Window

The Documentation Window allows you to create end-user and use case documentation for your processes. You add documentation for the entire process or for each flow object within your process. Figure 3–13 shows the Documentation Window.

Figure 3–13 Documentation WIndow



This graphic shows the Documentation tab page in Oracle BPM Studio for the sample process RequestQuote.

The window shows the standard word processing controls for formatting text. below these controls there is a text box for entering text.

****

Part II

Using Oracle BPM Studio

This part describes how to create and use Oracle BPM projects and project templates. It also describes how to create processes and use the process editor.

This part contains the following chapters:

4

Working with Projects and Project Templates

This chapter describes how to create and use projects using Oracle BPM Studio.

This chapter includes the following sections:

- Section 4.1, "Introduction to Oracle BPM Projects"
- Section 4.2, "Creating and Working with Projects"
- Section 4.3, "Introduction to Project Templates"
- Section 4.4, "Working with Project Templates"
- Section 4.5, "Using the Oracle BPM Metadata Service (MDS) Repository"

4.1 Introduction to Oracle BPM Projects

A BPM project is a container for the resources used to create and support business applications created using Oracle BPM. Oracle BPM projects are based on SOA projects but they include additional functionality of the Oracle BPM Suite, including BPMN processes.

You can create new projects directly in Business Process Composer or you can create and edit projects based on project templates.

Projects can be shared between Business Process Composer and Oracle BPM Studio or deployed to BPM runtime. See Section 3.2, "Overview of the Application Development Life Cycle" for information on how projects are used within the development life-cycle.

4.1.1 Introduction to Project Resources

Each BPM project contains one or more business process and may include other resources used by the business processes or overall application. This can include other reusable resources that allow you to connect your application to other applications and systems.

The following are the key resources of an Oracle BPM project:

- Processes: can include both BPM and BPEL processes.
- Activity Guide: includes project milestones defined for each BPMN process.
- Organization: includes the organization elements used to mimic the organizational structure of your organization within BPMN process models.
- Business Catalog: includes reusable services including services, adapters, and human tasks.

- Simulations: includes the simulations models defined for a project and individual BPMN processes.
- Resources: contain the XML transformations define for your project.

Each of these resources are accessible from the BPM Project Navigator. Additional application resources are accessible from the Application Navigator.

4.1.2 Sharing Projects Between Oracle BPM Users

Oracle BPM uses the Oracle MDS repository to share projects and project templates between other Oracle BPM Studio and Business Process Composer users.

See Section 3.2, "Overview of the Application Development Life Cycle" for more information on how projects and project templates are shared between BPM Studio and Business Process Composer.

See Section 4.5, "Using the Oracle BPM Metadata Service (MDS) Repository" for more information on the Oracle BPM MDS repository.

4.2 Creating and Working with Projects

The following section describes how to create new Oracle BPM projects and perform other project-related tasks. See Section 4.5, "Using the Oracle BPM Metadata Service (MDS) Repository" for information on working with projects in Oracle BPM MDS.

4.2.1 How to Create a New Project

Oracle BPM projects are created in the same way as other types of SOA composite application components.

To create a new Oracle BPM project:

1. From the **File** menu, select **New**.
2. Under Categories, select **BPM Tier**, then select BPM Project.
3. Click **OK**.
4. Enter a name for your project.
5. Ensure that BPM and SOA appear in the **Selected** column.
6. Click **Finish**.

The new project is created and appears in the project navigator. After creating the project, the New Process Wizard starts automatically. You can choose to create a new process or cancel the wizard.

See Section 5.1.2, "How to Create a New Business Process" for more information on creating a new BPMN process.

4.2.2 How to Open a Project from the File System

You can open an Oracle BPM project directly from the file system. This is generally used to open local projects that you have previously closed.

Projects that are shared with other users are imported from an exported Oracle BPM project or using Oracle BPM MDS.

To open a project:

1. Select **File**, then **Open**.
2. Browse to the location of your project folder.
3. Select the Java Project (.jpr) file for your project
4. Click **Open**.

The project appears in the project navigator.

> **Note:** When you open a project from the file system, the project remains in its original location. It is not copied to the Oracle Jdeveloper working directory.

4.2.3 How to Export a Project

Exported projects enable you to share projects with other Oracle BPM Studio users. This is useful when it is not feasible to share projects by publishing them to Oracle BPM MDS.

To export a project:

1. Select **Export** from the **File** menu.
2. Select **Export as BPM Project**, then click **OK**.
3. Provide a name for your project, the browse to the location where you want to export the project.
4. Click **Next**
5. Click **Next**, then Click **Finish**.

4.2.4 How to Import a Previously Exported Project

After you export an Oracle BPM project from Oracle BPM Studio or Oracle Business Business Process Composer, you can import it back to Oracle BPM Studio. This enables you to share projects directly from a file system instead of using Oracle BPM MDS.

To import a project

1. From the **File** menu, select **Import**.
2. Select Import BPM Project, then click **OK**.
3. Browse to the location of the .exp file of the exported project, the click Open.
4. Select a project root folder, then click Next.
5. Provide a project name, then click Next.
6. Click Next, then Finish.

4.2.5 How to Edit Project Preferences

You can edit project preferences to configure the behavior of an Oracle BPM project, including the following:

- Configure sampling points and Process Analytics.
- Configure general process properties.

- Add languages to a project

To Edit Project Preferences:

1. From the **View** menu, select **BPM Project Navigator**.
2. Right-click the project whose preferences you want to edit, then select **Project Preferences**.
3. Edit the project preferences as necessary, the click **OK**.

For more information on specific project preferences, see the online Help for Project Preferences.

4.3 Introduction to Project Templates

A project template is an Oracle BPM project that is used as a base for creating new Oracle BPM applications.

Project templates are created using Oracle BPM Studio. In a project template, process developers can create BPM project that contains all of the required services and other components of the business catalog.

Project templates can then be published to Oracle BPM MDS where process analysts can use them in Oracle Business Process Composer to create new deployable projects based on the project template.

The exact changes process analysts can make to an project created from a project template is defined using the edit policies of the project template.

4.3.1 Introduction to Edit Policies

Project templates allow you to define edit policies for BPMN processes and flow objects within them. 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 at the process level. However, you can also define edit policies for individual flow objects.

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: Edit policies are defined using Oracle BPM Studio. You cannot change the edit policy settings of processes and elements using Business Process Composer.

4.3.1.1 Process Level Edit Policies

In a project templates, each process contains an edit policy which determines the changes you can make to the process from Oracle Business Process Composer.

Table 4–1 describes the process level edit policies.

Table 4–1 Process Level Edit Policies

Edit Policy	Description
Flow Sealed	The overall flow of the process cannot be changed. A user can edit specific implementation details, but cannot change the process flow

Table 4–1 (Cont.) Process Level Edit Policies

Edit Policy	Description
Activity Sealed	Individual activities (flow objects) within the process cannot be changed. This includes editing flow object properties including assigning components from the business catalog.

Note: If you do not define an edit policy template users can change the process flow, including adding and deleting BPMN flow objects. They may also be able to edit flow objects properties depending on the edit policies you define at the activity (flow object) level.

4.3.1.2 Activity Edit Policies

Within a process, you can also define edit policies that apply to individual flow objects.

Table 4–2 Component Level Edit Policies

Edit Policy	Description
Sealed	The component cannot be modified
Must implement	Template users are required to implement this component in order to create a deployable project.
Can modify implementation	Template users may redefine this component if necessary.
Use process permission	The component inherits the edit policy of the process.

4.3.2 Using Data Objects and Variables in Project Templates

A project template defines 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 process 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 already defined in the project template. You cannot create new types of complex data objects.

4.3.3 Using the Business Catalog in Project Templates

Project templates allow you to incorporate elements of the business catalog. This allows you to create reusable services that can be used in each project created based on a project template.

The following business catalog components can be included in a project template:

- Human Tasks
- Business Rules
- Services
- Business Objects

Using Business Process Composer, process analysts can reuse these components within a project by assigning a business catalog component to its corresponding activity within a process.

When creating a project template that is shared with Business Process Composer, you must create the necessary business catalog components before publishing the template to MDS. Business Process Composer only enables you to create and edit some business catalog components. See *Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management* for more information.

4.4 Working with Project Templates

The following sections describe how to create project templates and set the edit policies for processes and activities within a process.

After creating a project template, see Section 4.5, "Using the Oracle BPM Metadata Service (MDS) Repository" for information on publishing it to Oracle BPM MDS.

After publishing project templates to the Oracle BPM MDS repository, business users can use them to create new deployable BPM projects. See Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management for more information.

4.4.1 How to Create a New Project Template

You can create a project template from an existing project by right-clicking a BPMN process in the Project Navigator, then selecting Create Process Template. You can also create a project template using

To create a project template:

1. From the **File** menu, select **New**
2. Select **BPM Tier**, then select **BPM Project Template**.
3. Click **OK**.
4. Provide a name for your project, then click **Finish**.

After clicking Finish, the new project template appears in the Project Navigator. You can create new processes and define any required edit policies for processes and activities.

Also, if you plan to share your template with process analysts using Business Process Composer, you should also define any necessary business catalog components before publishing the template to Oracle BPM MDS

4.4.2 How to Create a Project Template from an Existing BPM Project

You can create a project template from and existing BPM project. This enables you to continue developing a BPM project in Oracle BPM Studio while making it available as a project template.

To create a project template from an existing process:

1. Open your project.
2. In the BPM Project navigator, right-click your project then select **Convert to Template**.

Note: After converting a project to a project template, you cannot convert it back to a regular project.

---

3. Click **Yes**.

4.4.3 How to Set the Edit Policies for a Process in a Project Template

Setting the edit policies for a process determines the types of process-level changes that can be made to processes within projects created based on project templates.

To set the edit policies for a process:

1. In the Project Navigator, right-click on the process, then select **Properties**
2. Select the **Advanced** tab.
3. Under Seal Type, select from the following:
   - Flow Sealed: prohibits changes to the overall flow of the process.
   - Activity Sealed: prohibits changes to individual BPMN flow objects within the process.
4. Click **OK**.

4.4.4 How to Set the Edit Policies for an Activity in a Project Template

You can set the edit policies at the activity level within a process template.

Note: Edit policies defined at the activity level override process-level edit policies.

To set the edit policies for an activity:

1. Right-click on the activity where you want to change the editor policy.
2. Select **Properties**.
3. Select the edit policy from the Permissions drop-down menu
4. Click **OK**.

4.5 Using the Oracle BPM Metadata Service (MDS) Repository

The following sections provide an introduction to the Oracle BPM Metadata Service (MDS) repository. They also provide tasks on how to configure and use it.

For general information on configuring Oracle MDS see "Managing the Metadata Repository" in *Oracle Fusion Middleware Administrator's Guide*.

4.5.1 Introduction to the Oracle Metadata Service (MDS) Repository

Oracle Metadata Service (MDS) repository is an Oracle Fusion Middleware component that stores metadata for certain types of deployed applications. Oracle BPM uses this repository when deploying applications to run time.

For more information on Oracle MDS see "Managing the Metadata Repository" in *Oracle Fusion Middleware Administrator's Guide*.

4.5.2 Introduction to the Oracle BPM Metadata Service (MDS) Repository

In addition to using Oracle MDS to store information about deployed applications, Oracle BPM also creates a partition in the MDS repository to store projects and project templates.

This partition is used by both Oracle BPM Studio and Business Process Composer to share project and project templates.

The Oracle BPM MDS repository contains the following default folders:

- **Public**: Contains all shared Oracle BPM projects.
- **Templates**: Contains all project templates.

You can create additional subfolders within these folders to organize your projects and project templates.

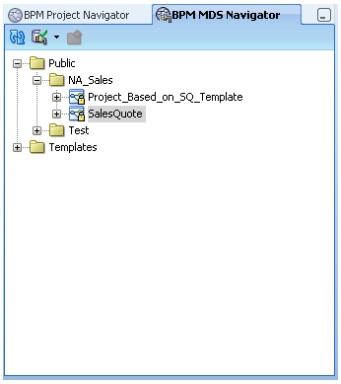
The Oracle BPM Metadata Services repository is installed as part of the Oracle BPM run time installation. After this installation is complete, you must configure your Oracle BPM Studio installation to connect to the repository.

4.5.3 Introduction to the Oracle BPM Metadata Service Browser

The Oracle BPM Metadata Service browser enables you to view the contents of the Oracle BPM MDS repository and perform related tasks.

Figure 4–1 shows an example of the Oracle BPM MDS browser.

Figure 4–1 The Oracle BPM MDS Browser



This graphic shows an example of the Oracle BPM MDS browser.

At the highest level, there are two folders: Public and Templates.

The Public folder is expanded to display two folders: NA_Sales and Test. The NA_Sales folder is expanded to display two items.

****

You can perform the following tasks using this browser:

- Publish projects and project templates.
- Create and configure Oracle BPM MDS connections.
- Checkout and lock projects and project templates

4.5.4 How to Configure a Connection to the Oracle BPM Metadata Service Repository

You can configure a connection to the Oracle BPM MDS repository.

For general information on configuring Oracle MDS see "Managing the Metadata Repository" in *Oracle Fusion Middleware Administrator's Guide*.

To Configure an Oracle BPM MDS Connection

1. From the View menu, select BPM MDS Navigator.
2. Click Configure Connection.
3. Perform the following if you have not created an Oracle MDS connection:
   a. Click Add.
   b. Select IDE Connections.
   c. Provide a connection name.
   d. Select DB-Based MDS from the drop-down list.
   e. Select the database connection that corresponds to you Oracle MDS installation.
   f. Select **obpm** from the list of MDS partitions.
4. Select an MDS connection from the drop-down list
5. Click **OK**.

4.5.5 How to Refresh the Oracle BPM MDS Repository

Before performing any administrative tasks related to Oracle BPM MDS, you should refresh the repository. This helps ensure that you do not conflict with the work of other Oracle BPM users.

To Refresh the Oracle BPM MDS Repository:

1. From the **View** menu, select **BPM MDS Navigator**.
2. Click **Refresh** from the toolbar.

4.5.6 How to Publish a Project or Project Template to Oracle BPM MDS

Publishing a project to the Oracle BPM MDS repository enables you to share projects and project templates with other process analysts and process developers. After a project or project template is published to the repository, it can be accessed by other process analysts and developers using either Oracle BPM Studio or Business Process Composer.

To publish a project to the Oracle BPM MDS Repository:

1. From the **View** menu, select **BPM Project Navigator**.
2. In the BPM Project Navigator, right-click the project you want to publish, then select **Publish to BPM MDS**.
3. Enter the name for your project. This is the name used in the Oracle BPM MDS repository.
4. Click **Override** if you want to overwrite an existing project or project template in the repository.
5. Select the folder where you want to publish, then click **OK**.

4.5.7 How to Checkout a Project in Oracle BPM MDS

Checking out a project copies a project from Oracle BPM MDS to your local file system. After checking out a project, you can make changes and edit the project locally. You can then republish the project to MDS.

To checkout a project from Oracle BPM MDS:

1. From the View menu, select BPM MDS Navigator.
2. Expand the folder containing the project you want to checkout.
3. Right-click the project, then select Checkout.
4. Provide a project name. This is the name of the project on your local file system.
5. Click **OK**.
6. Click **OK** to lock the project if necessary.

After checking out a project, you can edit it locally on your file system. If you need to ensure that other users do not make changes to the project in the repository, you can lock the project.

4.5.8 How to Lock or Unlock a Project in Oracle BPM MDS

Oracle BPM Studio enables you to lock and unlock projects stored in Oracle BPM MDS. This is useful when you need to make changes to a checked-out project and want to ensure that other users do not edit the project.

To lock or unlock a project in Oracle BPM MDS:

1. From the View menu, select BPM MDS Navigator.
2. Expand the folder containing the project you want to lock or unlock.
3. Right-click the project, then select Lock or Unlock.

> **WARNING: Unlocking a project that is locked by another may cause the owner of the original lock to lose all changes.**

4. Click **OK**.

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.

This chapter includes the following sections:

5.1 Working with Processes

This section provides information about creating and using business processes in Oracle BPM Studio.

5.1.1 Introduction to Business Processes

A business process can be generally defined as a sequence of tasks that after it performed 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.

5.1.1.1 Types of Processes

Oracle BPM enables you to create different types of BPMN processes depending on what work the process must perform. Table 5–1 describes the different types of processes supported by Oracle BPM.

Table 5–1 Process Types

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	A process that can be invoked from a call activity. reusable processes can only be invoked using the call activity.

5.1.2 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. Open your project.
2. Expand the node for your project in the BPM Project Navigator.
3. Right-click Processes, then select **New** then **Process**.
4. Select the type of process you want to create, then click **Next**.

   See Section 5.1.1.1, "Types of Processes" for more information on process types.
5. Enter a name and optional description.
6. Click **Ok**.

The new process is opened in the process editor.

New business processes are created with a start and end event connected by a default sequence flow. The type of start and end events depend on the type of process you created.

5.1.3 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. Open your project.
2. Expand the project node in the Project Navigator.
3. Expand **Processes**.
4. Double-click the process you want to open.

The process opens in the process editor window. See Section 5.2, "Introduction to the Process Editor" for more information on working with processes in the process editor.

5.1.4 How to Delete a Business Process

You can delete processes from your project. However, you should ensure that there are no remaining references to the deleted process elsewhere in your project.

To delete a business process from a project:

1. Open your project.
2. Expand **Processes** in the Project Navigator.
3. Right-click the process you want to delete, then select **Delete**.

5.1.4.1 What You Need to Know About Deleting a Business Process

When you delete a business process from a project, you must ensure that you remove any references to it from other parts of your process.

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.

5.1.5 How to Edit Process Preferences

You can edit the preferences for each process within a project using the Project Navigator.

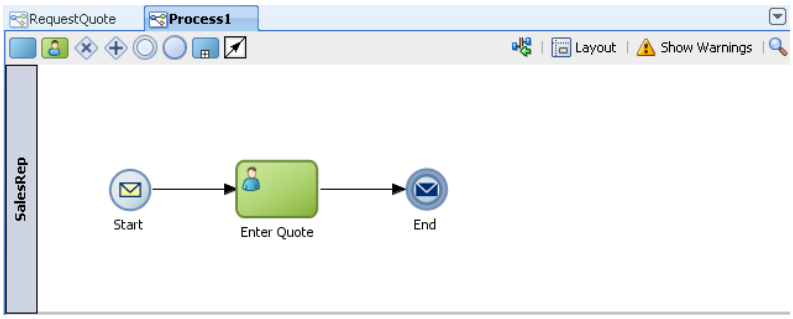
To edit process preferences:

1. Open your project.
2. Expand **Processes** in the Project Navigator.
3. Right-click the process whose properties you want to edit, then select **Properties**.
4. Edit the properties as necessary, then click **OK**.

5.2 Introduction to the Process Editor

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

Figure 5–1 The Process Editor



This graphic shows an example of the Process Editor. A tab page for a sample Process1 is displayed. It shows a swimlane labeled SalesRep. This swimlane contains a message start event with a sequence flow to a user task labeled Enter Quote. This task, in turn, has a sequence flow to a message end event.

There are icons that enable you to insert the various BPM notations, and controls for configuring the layout, showing warnings, and searching.

****

Flow Object Toolbar

The flow object toolbar provides easy access to common BPM flow objects. The following flow objects are available:

- Generic Activity
- User Task
- Exclusive Gateway
- Inclusive Gateway
- None Catch Event
- None Throw Event
- Subprocess
- Sequence Flow

See Chapter 6, "Modeling Business Processes with Oracle BPM" for more information on BPMN flow objects.

Go To Composite Editor

This toolbar item opens the SOA Composite Editor.

Layout

This toolbar item enables you to use and configure the auto layout utility.

Show Warning

This toolbar item enables you to determine the severity of messages displayed in the Log window. The following severity levels are provided:

This can also be configured in the project preferences. Se e Chapter 4.2.5, "How to Edit Project Preferences" for more information.

- **None**: No errors or warnings are displayed to the log window.
- **Show Errors**: Only error messages are displayed to the log window.
- **Show Warnings**: Both warning and error messages are displayed to the log window.

Zoom

This toolbar item enables you to zoom in and out of the processes.

5.3 Working with Flow Objects in Your Process

The following sections provide tasks for adding flow objects to your process and how to work with flow objects within the process editor.

See Chapter 6, "Modeling Business Processes with Oracle BPM" for more information on BPMN flow objects.

5.3.1 How to Add Flow Objects from the Process Editor Toolbar

The process editor toolbar contains shortcuts for common BPMN flow objects.

To Add Flow Objects from the Process Editor Toolbar:

1. Open the process where you want to add a flow object.
2. Click the icon in the toolbar, then position the cursor at the point in the process you want to add the flow object.
3. Click again to add the flow object.
4. Edit the flow object properties as necessary, then click **OK**.

For more information on specific flow object properties, see the online Help for each flow object.

5.3.2 How to Add Flow Objects from the Component Palette

You can add BPMN flow objects from the Component Palette.

To Add Flow Objects from the Component Palette:

1. Open the process where you want to add a flow object.
2. From the **View** menu, select **Component Palette**.
3. In the Component Palette, click the flow object you want to add.
4. In the Process editor, position the cursor at the point in the process where you want to add the flow object.
5. Click to add the flow object.
6. Edit the flow object properties as necessary, then click **OK**.

5.3.3 How to Edit Flow Object Properties

You can edit the properties for each flow object within your process.

To Edit the Properties of a Flow Object:

1. Open the process containing the flow object you want to edit.
2. Right-click the flow object, then select **Properties**.
3. Edit the properties as necessary, then click **OK**.

5.4 Documenting Your Process

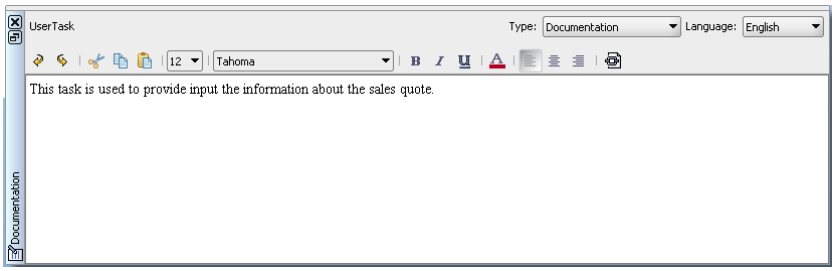
The documentation editor contains a toolbar and editor pane that enables you to enter the documentation for your process and for the flow objects within your process.

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

5.4.1 Introduction to the Documentation Editor

Figure 5–2 shows the documentation editor.

Figure 5–2 The Documentation Editor



This illustration is described in the text.

****

The documentation editor contains a toolbar and a text editor window. The tool bar allows you to select the type of documentation you want to create and allows you to select the language if you have defined additional languages for the product.

For more information on the documentation editor toolbar see the online Help.

5.4.2 How to Add Documentation to Your Process

You can add documentation to your process using the Documentation editor.

- **Documentation**: This is the documentation the process participants see using the Oracle BPM WorkSpace application.
- **Use case documentation**: This is the documentation that process analysts and process developers see when updating a business process.

To Add Documentation to a Flow Element in a Process

1. Open the process where you want to add documentation.
2. From the **View** menu select **Documentation**.
3. Select the flow object within your process that you want to document.
4. From the drop-down list, select the type of documentation you want to add.
5. Enter your documentation.
6. From the File menu select Save to save your changes.

Part III

Modeling a Process

This part describes how to use Oracle BPM Studio 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:

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 around 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/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 allow you to define functional categories that represent job functions or responsibilities within your organization.

The roles defined in you 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.

6.1.1.1 Roles in Context

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

The Sales Quote example project 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: Represent users who are responsible for approving the combination of products and pricing structure defined by the sales quote.
- Business Practices: Represent 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: 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 Chapter 2.2, "Introduction to the Sales Quote Example Project" 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.

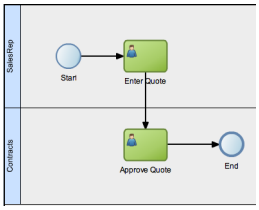
When you create a new manual process, you must specify a role as part of the wizard for creating the process.

Note: You cannot remove the role for a swimlane that contains a start or end event.

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 task 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 Adding Roles and Swimlanes to Your Process

To add a new swimlane and role to your process:

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. If you added a user or manual task, the swimlane contains a default role name.

6.1.4 Sharing Roles Between Business Process Composer and BPM Studio

Oracle BPM Studio enables you to integrate roles within a 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 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.

Since 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 the none end and start events, start and end events can have input and output to processes.

6.2.1.1 Default Start Events for Process Patterns and Subprocesses

When you create a new process Oracle BPM Studio and Business Process Composer create a default start and end event. The type of start and end event created depends on the type of process you are creating.

Table 6–1 shows the default start and end events for each process pattern.

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

Process Type	Default Start and End Event Types
Default process (Oracle BPM Studio only)	Message start and end event
Asynchronous service	Message start and end event
Synchronous service	Message start and end event
Manual process	None start and end event

See Chapter 5, "Working with Processes and the Process Editor" 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 Defining How a Process Instance is Triggered

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

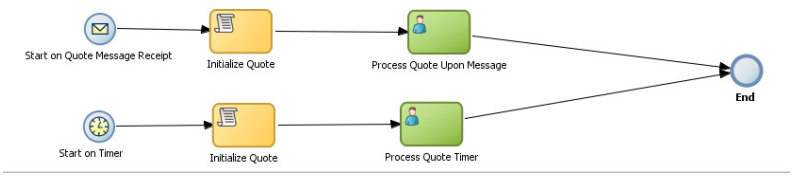
- Using a message, signal, or timer start event.
- Using a none start event followed by a receive task. The receive task must be configured to create a process instance.
- Using a none start event followed by a user task defined with the initiator pattern.

6.2.1.3 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.

6.2.1.4 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 terminated when the end event is reached.

When you are using multiple ends it is possible for different tokens to take different paths within a process. In normal 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 terminated 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 6.2.7, "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 6.2.9, "Introduction to the Terminate End Event" for more information.

6.2.2 Introduction to the None Start Event

The none start event is used when no instance trigger is 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. Although the none start event does not trigger the creation of a process instance, it is required when triggering a process instance using the following flow objects:

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

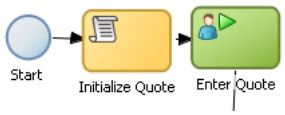
Like other start events, the none start event cannot have incoming sequence flows. It can only have default out-going sequence flows.

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

6.2.2.1 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.

****

In this example, the process instance is created by the Enter Quote user task. This user task is implemented using the Initiator pattern.

6.2.2.2 Data Associations

The none start event does not accept process input arguments.

6.2.3 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 for of exchanging 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.

****

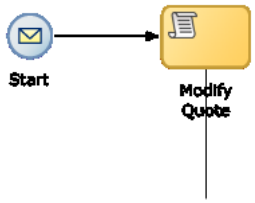
Like other start events, the message start event cannot have incoming sequence flows. Message start events require a default outgoing sequence flow.

You can exposed a BPMN process as service which 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. For more information see "Communicating with Other BPMN Processes and Services" in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

6.2.3.1 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 which 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.

****

6.2.3.2 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.

6.2.4 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 and signal start events, you can invoke multiple processes simultaneously.

The signal start and throw events are generally 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.

****

6.2.4.1 The Signal Start Event in Context

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

6.2.5 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 PM.
- 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.

****

6.2.6 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 place-holder by process analysts, to be changed later during implementation by a process developer.

Figure 6–9 shows the default notation for the none end event.

Figure 6–9 The None End Event

The none end event is represented by single circle.

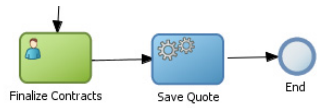
****

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

6.2.6.1 The None End Event in Context

Figure 6–10 shows an example of the none end event within the Sales Quote example process. In this example, the Sales Quote service task is used to perform the task of saving information about the sales quote to a database.

Figure 6–10 The None End Event within the Sales Quote Example Process



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.

****

Since no other work must 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.

6.2.7 Introduction to the Error End Event

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

Errors end events are normally used with the error boundary event. The error boundary event is used to alter 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 6–11 shows the default notation for the error end event.

Figure 6–11 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*.

6.2.8 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. For example, you cannot use a message end event to end a process instance initiated by a timer start.

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

Figure 6–12 The Message End Event

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

****

For information on how 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*.

6.2.9 Introduction to the Terminate End Event

The terminate end event is used to immediately terminate a process. When a terminate end event is reached, the process ends immediately. There is no error handling or additional clean up performed.

6.3 Adding User Interaction to Your Process

Most 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.

6.3.1 Introduction to 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.

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

- Human 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 (human activity)
- 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 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 human 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 the Oracle BPM and SOA suites.

Using Human Tasks, process developers can define the interface and workflow for end-user interaction by creating the following:

Human Tasks are reusable services that can be used within other processes that require the same UI.

- Roles and assignments
- Deadlines and escalations
- Presentations

Human tasks are created using Oracle BPM Studio.

> **Note:** In Business Process Composer, human tasks can be used in projects and project templates created in BPM Studio. You cannot create or edit Human Tasks in Business Process Composer.

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 6–13 shows the default notation for the user task.

Figure 6–13 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 Oracle BPM WorkSpace. The specific user interface elements, including the screens and panels that process participants see, are created using Oracle Human Tasks.

When designing a process, process analysts often only 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.

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*.

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

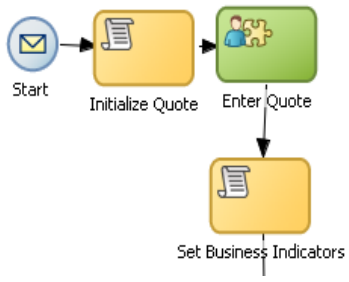
User tasks may also contain incoming and default outgoing sequence flows.

Note: Using Business Process Composer, you can only assign Human Tasks to User Task that were created as part of a project template in Oracle BPM Studio. You cannot create new human tasks or edit existing ones.

6.3.2.1 The User Task in Context

In the Sales Quote demo process, the Enter Quote task, shown in Figure 6–14, represents the work of entering information about the quote.

Figure 6–14 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 end-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 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.

> **Note:** Interactive activities are not available from the Business Process Composer component palette. To model user interaction with Business Process Composer, you can add a generic user task to your process. Interactive activities can later be implemented later by process developers using Oracle BPM Studio.

Table 6–2 shows the interactive activities are available from the Oracle BPM Studio 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 swim lane.
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 is automatically set to the role/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, add Human Tasks to the business catalog. Process analysts can use these in Business Process Composer when working with projects created from project templates.

> **Note:** You cannot create or edit Human Tasks using Business Process Composer. Human Tasks are created using Oracle BPM Studio

When adding the user task to a project template to be used within Business Process Composer, you should 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 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 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 of 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 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 6–15 shows the default notation for the manual task.

Figure 6–15 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 allow 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 process, a manual task could be added for printing and signing a copy of a formal contract as shown in Figure 6–16.

Figure 6–16 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, since it is not managed by the BPMN Service Engine, a manual task is used.

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

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 service 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 6–17 shows the default notation for the service task.

Figure 6–17 The Service Task

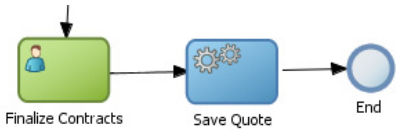
The service task is represented by blue rectangle with two gears in the middle.

****

6.4.1.1 The Service Task in Context

Figure 6–18 shows an example of the service task used to save the finalized sales quote to a database.

Figure 6–18 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 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 6–19 shows the default notation for the call activity.

Figure 6–19 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.

6.4.2.1 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.

6.4.2.2 Behavior of the Call Activity When Calling a Reusable Process

The following figure shows how control flow is passed to a child process and back to the parent process.

1. A token reaches the call activity in the parent process
2. A new instance of the child process is created.
3. The child process continues running until an end event is reached or an error occurs.
4. The process flow returns back to the call activity in the parent process.
5. Flow of the main process continues.

6.4.3 Introduction to the Send Task

The send task sends a message to a system or process outside the current process. Once 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 similar functionality 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 6–20 shows the default notation for the send task.

Figure 6–20 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*.

6.4.3.1 The Send Task in Context

See Chapter 6.4.5, "Using the Send and Receive Tasks to Communicate Between Processes" for information on using the send and received tasks to communicate between processes.

6.4.4 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. Once this message is received, the task is complete and running of the process continues to the next task in the process flow.

Figure 6–21 shows the default notation for the receive task.

Figure 6–21 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*.

6.4.4.1 The Receive Task in Context

See Section 6.4.5, "Using the Send and Receive Tasks to Communicate Between Processes" for information on using the send and received tasks to communicate between processes.

6.4.4.2 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.

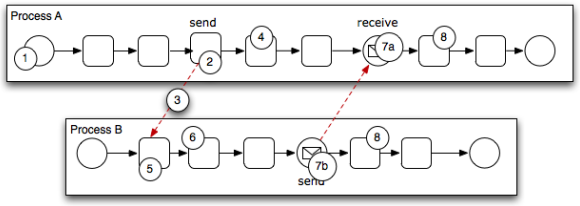
The following section describes how to use the send and receive tasks to communicate between processes.

6.4.5 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 6–22 outlines the basic behavior when using send and receive tasks to invoke a process and receive a response.

Figure 6–22 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 6.4.4.2.
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 communicate between the two processes.

6.4.6 Introduction to the Message Throw Event

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

Figure 6–23 shows the default notation for the message throw event.

Figure 6–23 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 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 6.2.3, "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 toke immediately moves to the next flow object of the process.

If your process receives a response synchronously, you should 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 similar functionality 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.

6.4.7 Introduction to the Message Catch Event

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

Figure 6–24 shows the default notation for the message catch event.

Figure 6–24 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 6.4.8 for information on how message throw events with message catch event.

For information on how 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*.

6.4.8 Using Message Throw and Catch 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 to invoke a synchronous process, there is no mechanism for catching messages synchronously from the process.

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

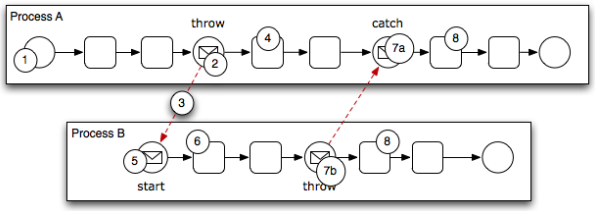
- The first time you use a throw event it must be paired to the message start event of the other process. This is required to trigger the process instance. After the instance has been trigger, you can use subsequent message throw events that are caught by the second process.

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 terminate processes invoked with a message throw event

Figure 6–25 shows the basic behavior when using throw and catch event to invoke a process and receive a response.

Figure 6–25 Using Message Throw and Catch Events Between Processes



This figure shows two processes: Process A and Process B.

****

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 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 rules task to incorporate Oracle Business Rules within your business processes. See Chapter 16, "Using 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 Rules Task

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

Figure 6–26 shows the default notation for the business rules task.

Figure 6–26 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 outwards.

****

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

- Using Structural Rules

  Structural rules allow you to perform calculations used within your business process. For example, you could 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 check perform a check of rule conditions within the rules catalog. Then, as part of the output data association, assign a value to a data object using an expression.

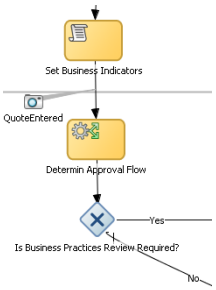
In this scenario, the business rules task is immediately followed by a gateway which is used to branch the process path according to the value of the data object.

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 project.

6.5.2.1 The Business Rule Task in Context

Figure 6–27 shows an example of the business rules task within the Sales Quote example process.

Figure 6–27 The Business Rules 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 flow 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 normal path between two flow objects. Default sequence flows are displayed as an arrowed line as shown in Figure 6–28.

Figure 6–28 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.

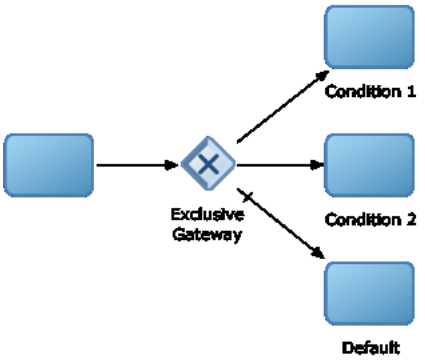
Exclusive, inclusive, and conditional gateways cannot have unconditional outgoing sequence flows. These gateways use conditional and default sequence flows to determine the flow of your process.

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 6–29 shows two outgoing conditional sequence flows and a default sequence flow.

Figure 6–29 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 gateways
- Conditional gateways
- Event-based gateways

The conditions used within a conditional sequence flow are defined using expressions. See Section 22.1, "Introduction to Expressions in Oracle BPM" 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 6–29.

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 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. 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 6–30 shows the default notation for the exclusive gateway.

Figure 6–30 The Exclusive Gateway

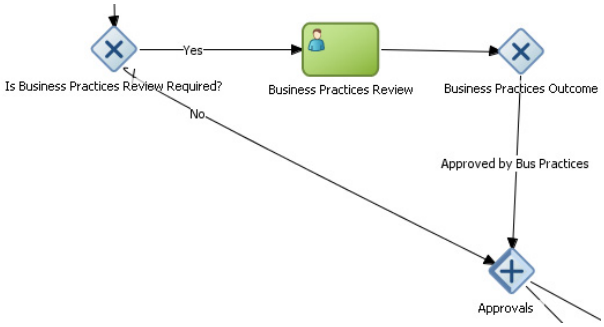
The exclusive gateway is represented by a diamond-shaped icon with an X in the middle.

****

6.7.2.1 The Exclusive Gateway in Context

Figure 6–31 shows an example of the exclusive gateway used within the Sale Quote example process. Here, the exclusive gateway is used to evaluate whether a review of business practices is required.

Figure 6–31 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 flow 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 6–32 shows the default notation for the inclusive gateway split.

Figure 6–32 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 6–33 shows the default notation for the inclusive gateway merge.

Figure 6–33 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 evaluate to true. Additionally, a token is generated for the default sequence flow.

These tokens are merged 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. Once all of these 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 where your process must perform multiple tasks in parallel.

Figure 6–34 shows the default notation for the parallel gateway split.

Figure 6–34 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 6–35 shows the default notation for the parallel gateway merge.

Figure 6–35 The Parallel Gateway (Merge)

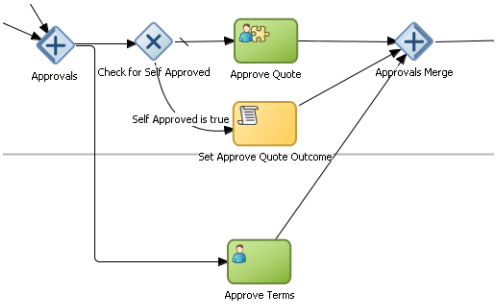
The parallel gateway merge is represented by a diamond-shaped icon with a plus sign in the middle. It is shaded on the right-hand side to represent the merge of the parallel gateway.

****

6.7.4.1 The Parallel Gateway in Context

The Sales Quote example process uses a parallel gateway during the approval stage of the process. Figure 6–36 shows how the parallel gateway is used to perform two process paths simultaneously.s

Figure 6–36 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: You should 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 a 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 since the inclusive gateway immediately evaluates all the conditional sequence flows, all of the flow objects in these process paths are also run.

Figure 6–37 shows the default notation for the complex gateway split. This icon is identical to the inclusive gateway split.

Figure 6–37 The Complex Gateway (Split)

The complex gateway is represented by a diamond with a star in the middle.

****

Figure 6–38 shows the default notation for the complex gateway merge.

Figure 6–38 The Complex Gateway (Merge)

The complex gateway is represented by a diamond with a star 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 6–39 shows the default notation for the event based gateway.

Figure 6–39 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 than 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, then the path that follows 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 flow are also deleted. However, 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 normal 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 normal 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 6–40 shows the default notation for the timer catch event.

Figure 6–40 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 event 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 defines. The flow that an interrupting can return directly to the main process flow.

When an 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

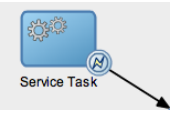
Error catch events are intermediate events used to handle an error that occurs within your process flow. Error catch events are always used as boundary events and can be attached to the following:

- Service Tasks
- User Tasks
- Send Tasks
- Receive Tasks
- Script Tasks
- Rules Tasks
- Event Subprocesses
- Subprocesses

Error catch events are always interrupting, meaning that they interrupt the normal flow of a process.

Figure 6–41 shows the default notation for the error catch event attached as a boundary event on a service task.

Figure 6–41 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 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 generally 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 back to the main flow.

  Note: If the boundary event is non-interrupting, the boundary flow cannot return to the main flow.

- The process flow from the error ends in an error event. The process is terminated immediately. Process control is passed to the service or process that initiated the process.

6.9 Using Subprocesses to Organize Your Process

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 6–42 shows how a collapsed subprocess appears within a process.

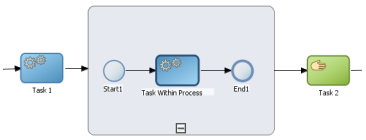
Figure 6–42 Example of a Collapsed Subprocess

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 6–43 shows how an expanded subprocess appears within a process. When a subprocess is expanded, you can edit the flow objects within. You can also click and drag the edge of the subprocess window to make the window larger or smaller.

Figure 6–43 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.

****

Like other types of processes, subprocesses have start and end events and contain their own flow. A subprocess must begin with a none start event and end with a none end event. Subprocesses do not contain swimlanes.

Subprocesses also behave like 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 boundary events.

If necessary, your process can contain nested subprocesses. However, you should use nested subprocesses only when necessary to make your process more readable.

6.9.1 Subprocesses and Sequence Flows

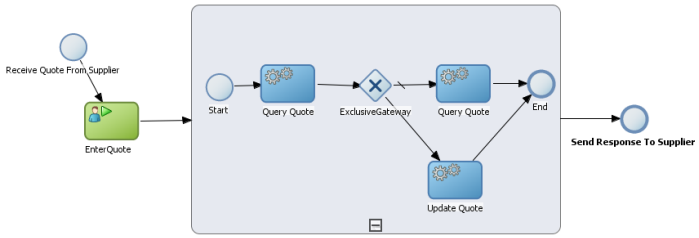
The flow objects within a subprocess cannot have sequence flows that connect to flow objects outside the subprocess.

Like other flow objects, subprocesses have incoming and outgoing sequence flows.

6.9.2 Subprocesses in Context

Figure 6–44 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 6–44 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.

****

6.9.3 Looping Subprocesses

You can configure a subprocess to repeat numerous within the context of a process flow. This is something that a process analyst should consider when designing a process, but the implementation is generally performed by process developers.

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.

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 model this explicitly this 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 6–45 shows the default notation for the script task.

Figure 6–45 The Script Task

The script task is represented by a yellow rectangle with a scroll icon in the middle.

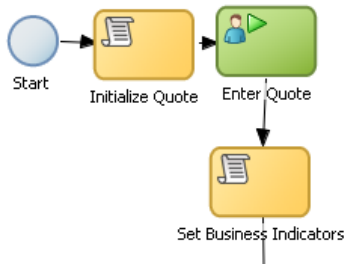
****

Script tasks are generally 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 6–46 shows two examples of the script task used at the beginning of the Sales Quote example process. The Sales Quote example process uses a script task to set initial values for data objects at the beginning of a the process and to set values for several business indicators.

Figure 6–46 The Script Task within the Sales Quote Example Project



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.

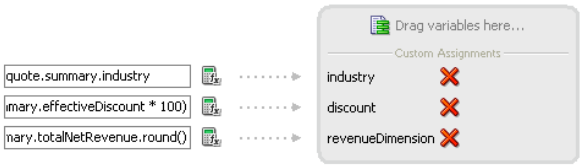
From the Enter Quote task, a sequence flow extends to a script task labeled Set Business Indicators.

****

Project data objects are data objects that you define in a project, all the processes in that project have those 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 6–47 shows the data associations used to set initial values for the business indicators.

Figure 6–47 Data Associations Used by the Set Business Indicators Script Task

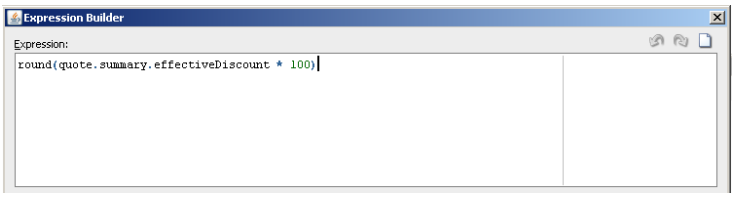


This figure shows how the data associations editor appears for the Set Business Indicators script task in the Sales Quote example project. It contains two columns. On the left is a list of data objects representing the input arguments to the script task. On the right are the three data objects that these are mapped to.

****

As with other flow objects that accept data associations, you can use expressions to change the values of data objects. Figure 6–48 shows how an expression is used to alter the value of the discount project variable.

Figure 6–48 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).

****

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 to 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 process you might want to view the total amount 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** text-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 you can click the **New** button under the **Selected** list.

7. Click **OK**.

6.12 Using Guided Business Processes to Set Project Milestones

This section describes how to you Guided Business Processes in Business Process Composer.

6.12.1 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.

6.12.1.1 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 has to 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 resume it at a later time.

6.12.2 Working with Guided Business Processes

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

To configure the activity guide for a project:

1. In the Project Navigator, expand the project where you want to configure the activity guide.
2. Right-click **Activity Guide**, then select **Configure**.
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 display the milestone and task links for all the milestones in this guided business process.
  - **When Instantiated**: Display 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 try to 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. In the Project Navigator, expand the project where you want to configure the activity guide.
2. Right-click **Activity Guide**, then select **Configure**.
3. Click New Milestone.
4. Select the milestone you just created from the list.
5. Configure the milestone as necessary.
6. 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.

7

Modeling Your Organization

This chapter describes how to use

This chapter includes the following sections:

7.1 Introduction to Organizations

Using Oracle BPM, you can create an organizational model that mimics your real world organization. During deployment of your project, the components of the modeled organization are mapped to your real-world organization.

In Oracle BPM, organizations are composed of the following components:

- Roles
- Organizational Chart
- Holidays
- Calendars

Organizations are defined at the project level. You can export organizational information to be used within other projects.

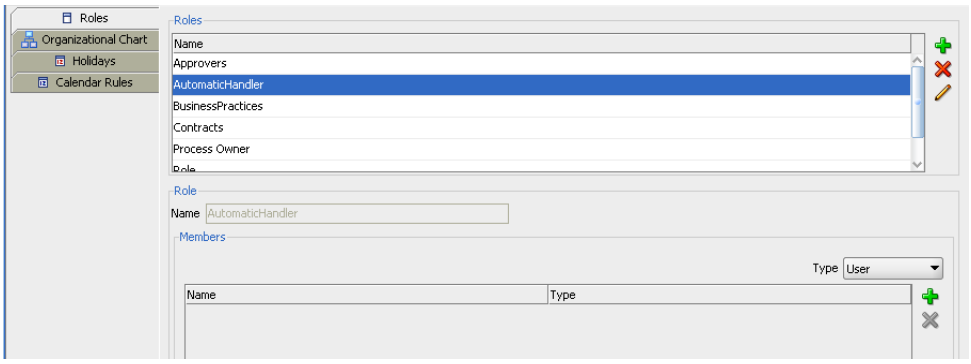
Note: You cannot create organizational charts, calendars, or holidays using Business Process Composer. You can define roles and assign them to swimlanes.

Note: Organization information is not carried over when a project is deployed to runtime.

7.1.1 Introduction to the Organization Editor

The organization enables you to create and edit the components within an organization. It contains tabbed pains for each of these components. Figure 7–1show an example of the Organization editor with the Roles tab selected.

Figure 7–1 The Organization Editor



This graphic of the Organization Editor shows a navigation panel on the left. The items in the navigation panel are Roles, Organizational Chart, Holidays, and Calendar Rules.

Selecting an item from the navigation panel opens the details for that panel in the right pane.

The Roles item is selected, and the corresponding details are displayed in the right pane.

****

7.2 Introduction to Roles

Allow you to define areas of responsibility that represent job functions or responsibilities within your organization. If your process-based application requires human interaction, you will have to define at least one role within your project.

Roles are abstract and help define and mimic responsibilities of an individual in the Enterprise. They need to be mapped to Participants.

The Order demo example process defines several roles including: Approvers and Sales Rep. These represent the types of people that perform the work within your process rather than specific people within your organization. Roles are assigned to the vertical swimlanes that show graphically the roles responsible for completing activities and tasks within your process. Roles also contain members which correspond to the end users responsible for using the actual process-based business application.

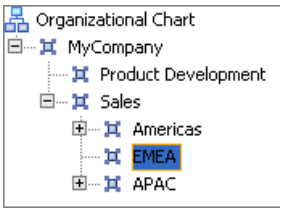
7.3 Introduction to Organizational Charts

An organizational chart defines the structure of your organization. Each project contains one organizational chart that can be divided into multiple organizational units that reflect the structure and hierarchy of an organization.

7.3.1 Introduction to Organizational Units

Organizational units define the structure of your organization An organizational chart contains one top level and may contain multiple levels of nested organizational units. Figure 7–2 shows how an organization can be structured using organizational units.

Figure 7–2 Example of Nested Organizational Units



This graphic shows a partially expanded organizational chart. The top level is MyCompany, which is expanded to display Product Developemnt and Sales. The Sales item is further expanded to display Americas, EMEA, and APAC.

****

In this example, MyCompany is the top-level organizational unit. Beneath MyCompany are various levels of nested organizational units.

For each organizational unit, you can assign members that represent the people within your organization. These are defined in Oracle WebLogic Server and are assigned using the Oracle Identity Service.

The following: members can be defined:

- Users: The individual participants or users
- Groups: Groups of participants. These are defined
- Application Roles

See <xref> for more information on mapping members to organizational units.

7.3.2 Introduction to Calendars

Calendars define when the resources in your organization are working. They allow you to define the following:

- The working days within a week.
- The start and finish times for each day.
- The time zone
- An optional holiday rule

You can specify a calendar rule for each organization unit. This allows you to model how your organization is structured across time zones and geographical regions.

7.3.3 Introduction to Holidays

You can define an optional holiday rule for each calendar rule in your organization. Holidays allow you to define the non-working days for a calendar rule. These can be viewed as exceptions to the normal working days you define in a calendar rule.

7.4 Working with Roles

The following sections describe how to create and edit roles.

7.4.1 How to Create a New Role

You can create roles to define who is responsible for performing the activities and tasks within your process. User tasks require you to define roles before you can add them to a process model.

To create a new role:

1. In the Project Navigator, expand the project where you want to create a new role.
2. Right-click **Organization**, then select **Open**.
3. In the Organization Editor window, select the Roles tab.
4. Click the **Add** icon, then supply a name for your role.
5. Click **OK**.

7.4.2 How to Add Members to a Role

Adding members to a role allows you to define what members of your real-world organization are responsible for performing the activities and tasks within your process. Before performing this task, you should ensure that you have configured a connection to your application server.

Note: Before performing this task, you should ensure that you have created an Identity Service connection.

To add members to a role:

1. In the Project Navigator, expand the project where you want to create a new role.
2. Right-click Organization, then select Open
3. In the Organization Editor window, select the Roles tab.
4. Click the **Add Role** icon
5. Select the type of application server and realm.
6. Enter a search pattern, then click the search icon.
7. Select the appropriate user from the search results, the click Select.
8. Click **OK**.

7.5 Working with Organizations

The following sections describe how to create and edit the components of an organization.

7.5.1 How to Create an Organizational Unit

You can create multiple organizational units within an organization.

To create an organizational unit:

1. In the Project Navigator, expand the project where you want to create a new role.
2. Right-click **Organization**, then select Open.
3. In the **Organization Editor** window, select the **Organizational Chart** tab.

4. Select Organizational Chart, then click the **Add** icon.
5. Provide a name for your organizational unit, then click **OK**.

   This defines the top-level organizational unit.
6. If you want to add a hierarchical structure to your organization, select the organizational unit you just created, then click the Add icon.
7. Provide a name for the organizational unit, then click **OK**.

   You can repeat steps 6 and 7 if you need to add additional levels to your organization.
8. If you want to add an optional calendar rule, select the appropriate rule from the drop-down list.
9. When you are finished, select **Save** from the **File** menu to save your organizational chart.

7.5.2 How to Create a Calendar

You can create calendars that can be assigned to an organizational unit.

To create a calendar:

1. In the Project Navigator, expand the project where you want to create a new role.
2. Right-click Organization, then select **Open**.
3. In the Organization Editor window, select the Calendar tab, then click the Add icon.
4. Provide a name, then click **OK**.
5. Select the calendar rule from the list.
6. Click the checkbox next to each day of the week you want to include.
7. Specify the start and end time for each day.
8. If you want to include an optional holiday rule, select the appropriate holiday rule from the drop-down list.
9. When you are finished, select Save from the **File** menu to save your organizational chart.

7.5.3 How to Create Holidays

You can create holiday rules that can be assigned to a calendar.

To create a holiday rule:

1. In the Project Navigator, expand the project where you want to create a new role.
2. Right-click **Organization**, then select **Open**.
3. In the Organization Editor window, select the **Holiday** tab, then click the **Add** icon.
4. Provide a name, then click **OK**.
5. Select the holiday rule from the list, then click the Add icon.
6. Provide the following for the holiday rule, then click OK.
   - Description: A description of the holiday rule.

- Type:
- Date: The date for this holiday rule. To specify a range, you must create a new entry for each day.

7. Click **OK**.

8

Handling Information in Your Process Design

This chapter describes how to handle the information in your process using data objects and project data objects. It also shows you how to pass that information along the process and how to transform it when necessary.

This chapter includes the following sections:

8.1 Introduction to Handling Information in Your Process Design

Generally processes access and store information. Often the process flow is based on the value of this information. In other cases this information is the result of running the tasks in the process.

Oracle BPM supports the following data structures to keep track of this information:

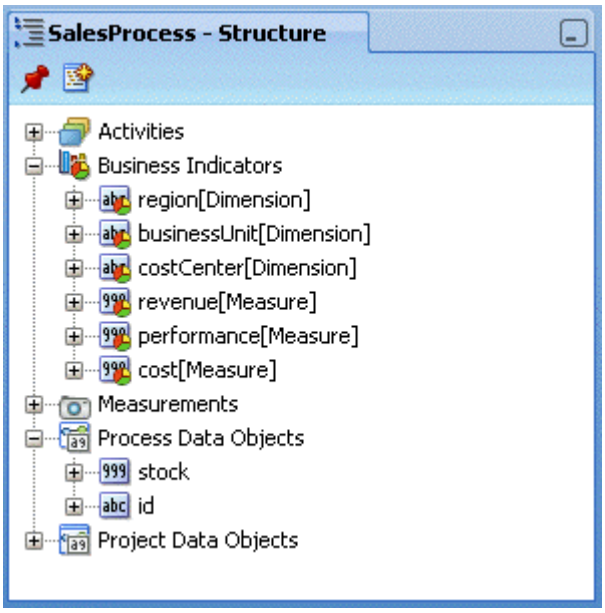
- Process Data Objects
- Subprocess Data Objects
- Project Data Objects
- Arguments

Additionally, you can pass information between the different elements of a process using data associations. Data associations enable you to pass information map the values of project and process data objects to the input and output arguments of the flow object implementations.

You can view the process data objects, project data objects, and business indicators in the Structure Window.

Figure 8–1 Show the structure window for a process that defines business indicators and process data objects.

Figure 8–1 Structure Window



This figure shows the Structure window for a process that defines business indicators and process data objects.

****

8.1.1 Basic Data Objects versus Complex Data Objects

Basic data objects are data objects defined using the basic types. For example, Int, Bool or String.

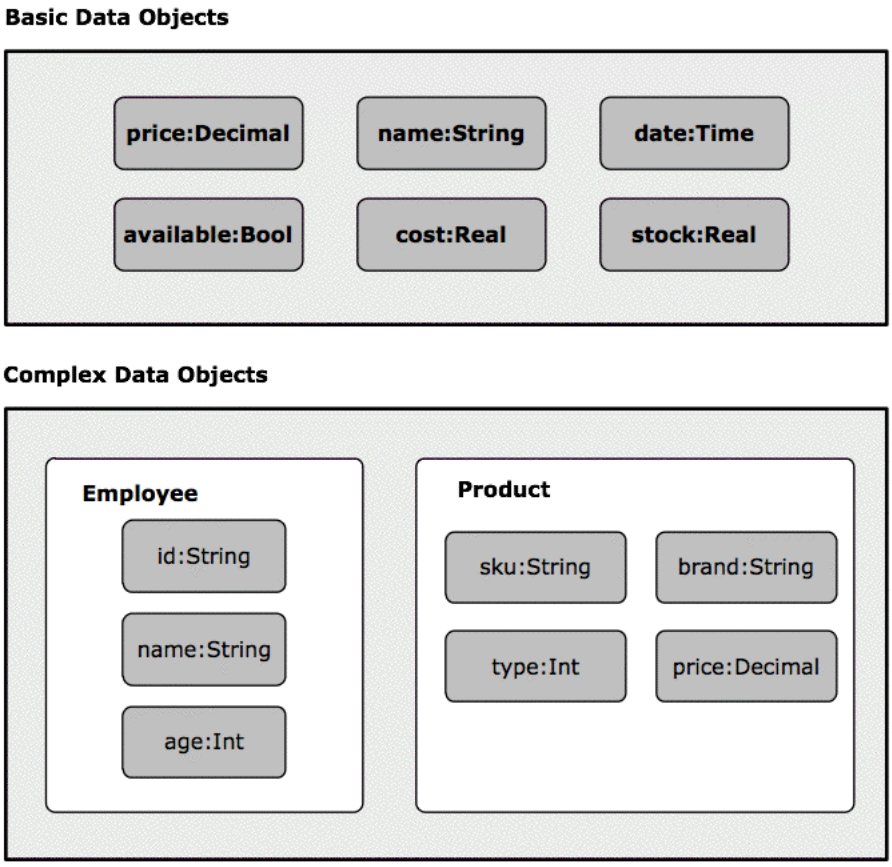
Complex data objects allow you to group data. Complex data objects are defined using business objects. See Chapter 13, "Modeling Business Objects" for more information on how to define business objects.

Business objects allow 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.

The structure of complex data objects is the same for all the process instances of a process. However the data they contain generally varies between the different instances of the process.

Figure 8–2 shows the relationship between basic data objects and complex data objects.

Figure 8–2 Basic Data Objects versus Complex Data Objects



This diagram shows examples of basic data objects defined using different basic types and examples of complex data objects defined using business objects.

****

8.2 Introduction to Data Objects

The main elements of a business process are tasks and information related to those tasks. The information of a process may change as you run the process. This information defines the state of a process at a given time.

This information determines how the process behaves for a particular instance. According to the value of this information the instance may take one path or another. You may monitor this information or store it to an external system.

The information of a process may change as the process runs. This information defines the state of a process at a given time.

The Sales Quote example process uses the following information:

- Approval flow
- Approval terms outcome
- Quote

Oracle BPM uses data objects to store the information related to the process. The value of these variables may or may not change as you run the process.

Oracle BPM data objects have the following characteristics:

- A name that identifies the data object
- A data type that determines the type of data that can be stored in the variable.

Data objects store information related to each process instance you create. The value of these data objects is different for every instance in the process. However the structure of the data object is the same for all process instances.

When you define a process you must define the data object to store information. You must also define in which part of the process you assign a value to these data objects. The value of data objects may come from the user input, from external systems or might be calculated based on other data objects.

When you create an instance, the Process Engine assigns Null as the default value for all the data objects defined for that process. Later on the activities in the process assign values to these variables.

In the purchase order process each order has its own total amount, payment type and customer ID. You can model this data by defining data objects that store this process information.

8.2.1 Supported Data Types for Data Objects

You can set the type of a data object to the following data types:

- String
- Int
- Bool
- Real
- Decimal
- Time
- Interval
- Binary
- Component

> **Note:** The binary data type is only used to map elements of an XML schema type. You cannot perform any operations with binary data types. You can only pass them between different components and flow objects.

8.2.2 Default Values

If you configure a data object to initialize automatically the BPMN Engine assigning it a default value. The default value varies according to the type of the data object.

Table 8–1 shows the default values for the supported data types.

Table 8–1 Default Values

Data Type	Default Value
String	""
Time, Date	'now'
Int, Real, Decimal	0

Table 8–1 (Cont.) Default Values

Data Type	Default Value
Bool	false
Interval	′0′

8.3 Working with Process Data Objects

You can add new process data objects to the process you are working on. You can also edit or delete them.

Typically the services in your process modify the value of the data objects in your process, but you might assign them an initial value, or change their value during the process.

8.3.1 How to Add a Process Data Object

To add a process data object:

1. In the BPM Project Navigator, select the process where you want to add the data object.
2. In the Structure window, right-click the **Process Data Objects** node.
3. Select **New**.
4. Enter a name to identify the data object.
5. Select a type from the Type list or click the **Browse More Types** button to select a type that is not included in the list.
6. Optionally, check **Auto Initialize** to initialize the data object with a default value.
7. Click **OK**.

Note: You can also add process data object from the Data Object tree in the Simple Expression Builder, XPath Expression Builder, and Data Association Dialog.

8.3.2 How to Edit a Process Data Object

You can modify the name and type of an existing process data object.

To edit a process data object:

1. In the BPM Project Navigator, select the process that contains the data object you want to edit.
2. In the Structure window, expand the **Process Data Objects** node.
3. Right-click the data object you wan to edit.
4. Select **Edit**.

   A dialog to edit the data object name and type appears.
5. Make the changes you want.
6. Click **OK**.

8.3.3 How to Delete a Data Object

You can delete a data object that you do not need or use.

To delete a data object:

1. In the BPM Project Navigator, select the process that contains the data object you want to edit.
2. In the Structure window, expand the **Process Data Objects** node.
3. Right-click the data object you want to edit.
4. Select **Delete**.

   A confirmation message appears.
5. Click **OK**.

8.3.4 How to Assign a Value to a Process Data Object

You can assign values to process data objects using a script task.

To assign a value to a process data object:

1. In the Process Editor, add a script task to the process.
2. Edit the implementation properties of the script task.
3. Define the data association or transformation to assign the value to the process data object.

   See Section 8.13, "Introduction to Data Associations" for information on how to define a data association.

   See Section 8.14, "Introduction to Transformations" for information on how to define a transformation.

8.4 Introduction to Activity Instance Attributes

Some data, like the status of the process, applies to all the process you define. You can use this data to trigger an event based on its value, or to provide it as input to a service. In both cases the process flow depends on the value of this data.

Oracle BPM tracks this data using a predefined set of activity instance attributes. You can access these activity instance attributes in the same way you access regular data objects, but you cannot assign them new values.

You can access activity instance attribute from the following components:

- Data associations
- Simple Expression Builder
- XPath Expression Builder

Table 8–2 provides detailed information about the activity instance attributes available for the different elements of a process.

Table 8–2 Activity Instance Attributes

Name	Type	Description	Availability
state	String	Specifies the state of the instance. Possible values are: ▪ none ▪ ready ▪ active ▪ canceled ▪ aborted ▪ completing ▪ completed	In complex gateways
loopCounter	Int	Specifies the number of times the engine ran this activity. The Process Engine updates this variable each time it runs a new loop.	In activities with loop marker.
loopCounter	Int	Specifies the sequence number that identifies each of the activations of this activity. The BPMN Engine assigns this number to each activation when it runs the activity.	In activities with multi-instance marker.
numberOfInstances	Int	.Specifies the number of activations created for a multi-instance activity. You can only access this value from the main instance.	In activities with multi-instance marker.
numberOfActiveInstances	Int	Specifies the number of active inner instances for a multi-instance activity. You can only access this value from the main instance. For sequential multi-instance activities this value is either 1 or 0. For parallel multi-instance activities this value is smaller or equal to the value specified by the predefined data object numberOfInstances.	In activities with multi-instance marker.
numberOfCompletedInstances	Int	Specifies the number of completed inner instances for a a multi-instance activity. You can only access this value from the main instance.	In activities with multi-instance marker.
numberOfTerminatedInstances	Int	Specifies the number of terminated inner instances for a a multi-instance activity. You can only access this value from the main instance.	In activities with multi-instance marker.
activationCount	Int	Specifies the number of tokens in the incoming sequence flow of the gateway.	In complex gateways.

8.5 Working with Activity Instance Attributes

Some process elements support activity instance attributes. You can use these activity instance attributes to control the flow of a process. Generally the Process Engine assigns the values of activity instance attributes, however some of them require you to assign them a value.

8.6 Introduction to Subprocess Data Objects

You can define data objects for a certain subprocess. These data objects are available only when the subprocess is running. When the instance leaves the subprocess the value of subprocess data objects is lost.

Using subprocess data objects is a good practice because:

- It reduces the number of unnecessary data objects in the main process, making it simpler and easier to read.
- By reducing the number of process data objects, it reduces the amount of memory each process instance occupies.
- It makes the subprocess easier to understand.

From within a subprocess you can access process data objects and subprocess data objects. If the name of a subprocess data object matches the name of a process data object, then when you access the data object you obtain the value of the subprocess data object.

8.7 Working with Subprocess Data Objects

You can add new project data objects to subprocesses. If necessary you can edit or delete them.

8.7.1 Adding a Data Object to a Subprocess

You can add data object to a subprocess. You can only access this data objects from within the subprocess.

To add a data object to a subprocess:

1. In the BPM Project Navigator, select the process that contains the subprocess where you want to add a data object.
2. In the Structure window, expand the **Activities** node.
3. Expand the node that corresponds to the subprocess.
4. Right-click the **Data Objects** node located under the subprocess node.
5. Select **New**.
6. Provide a name to identify the new data object.
7. Select a type or click the **Browse More Types** button to select a type that is not included in the list.
8. Optionally, check **Auto Initialize** to initialize the data object with a default value.
9. Click **OK**.

8.7.2 Editing a Data Object in a Subprocess

You can modify the name and type of an existing subprocess data object.

To edit a data object in a subprocess:

1. In the BPM Project Navigator, select the process that contains the subprocess with the data object you want to edit.
2. In the Structure window, expand the Activities node.

3. In the Structure window, expand the **Activities** node.
4. Expand the node that corresponds to the subprocess.
5. Expand the **Data Objects** node located under the subprocess node.
6. Right-click the data object you want to edit.
7. Select **Edit**.

   A dialog to edit the data object name and type appears.
8. Make the changes you want.
9. Click **OK**.

8.7.3 Deleting a Data Object from a Subprocess

You delete a subprocess data object that you do not need or use. If there are flow objects in your subprocess that use the removed data object, then you must remove these references manually.

To delete a data object from a subprocess:

1. In the BPM Project Navigator, select the process that contains the subprocess with the data object you want to delete.
2. In the Structure window, expand the **Activities** node.
3. Expand the node that corresponds to the subprocess.
4. Expand the **Data Objects** node located under the subprocess node.
5. Right-click the data object you want to delete.
6. Select **Delete**.

   A confirmation dialog appears.
7. Click **OK**.

8.8 Introduction to Project Data Objects

The processes in a BPM project often have a set of data they share. 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. The value of this data is different for every instance in each of those processes, they only share the necessity to keep track of that data.

Project data objects allow you to ensure that all the processes in a certain project keep track of a set of data. Then each process has to 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 WorkSpace views to show the values of those variables. This is only possible if you use project data objects.

Another benefit is that if you change the definition of a data object, then you only have to do it one time, as opposed to having to make those changes in all the processes in the project that define the same data object.

Note: It is not advisable to change the data type of a project data object after deploying a BPM Project. This can cause problems when the Process WorkSpace tries to render the value of the instances created before changing the data type.

Note: Avoid naming a project data object with the same name used for a process data object. If you name a process data object and a project data object with the same name, then the data associations editor does not allow you to access the project data object.

8.8.1 Business Indicators

When you mark a project data object as a business indicator the Process Engine stores its value in the Process Analytics databases. You can use this information to monitor the performance of your business processes.

For more information about Process Analytics, see Chapter 11, "Using Process Analytics".

8.8.2 Supported Data Types for Project Data Objects

You can set the type of a project data object to the following data types:

- String
- Int
- Bool
- Real
- Decimal
- Time
- Interval
- Binary
- Component

8.9 Working with Project Data Objects

You can add new project data objects to the project you are working on. You can also edit or delete them.

8.9.1 How to Add a Project Data Object

To add a project data object:

1. In the BPM Project Navigator, select a project.
2. In the Structure window, right-click the **Project Data Objects** node.
3. Select **New**.
4. Provide a name to identify the new project data object.

Note: You cannot use the name of existing process data objects.

5. Select a type.

   Some types allow you to define their length or decimal places.

6. Optionally, check **Auto Initialize** to initialize the project data object with a default value.

7. Click **OK**.

Note: You can also add process data object from the Data Object tree in the Simple Expression Builder, XPath Expression Builder, and Data Association Dialog.

8.9.2 How to Edit a Project Data Object

You can modify the name and type of an existing project data object.

To edit a project data object:

1. In the BPM Project Navigator window, select a project.
2. In the Structure window, expand the **Project Data Objects** node.
3. Right-click the project data object you want to edit.
4. Select **Edit**.

   A dialog to edit the project data object properties appears.

5. Make the changes you want.
6. Click **OK**.

8.9.3 How to Delete a Project Data Object

You can delete a project data object that you do not use or need. If there are processes in your project that use the deleted project data object, then you must remove these references manually.

How to delete a project data object:

1. In the BPM Project Navigator window, select a project.
2. In the Structure windows, expand the **Project Data Objects** node.
3. Right-click the project data object you want to delete.

   A confirmation message appears.

4. Click **OK**.

8.9.4 How to Assign a Value to a Project Data Object

You can assign a value to a project data object using a script task.

To assign a value to a project data object:

1. In the Process Editor, add a script task to the process.
2. Edit the implementation properties of the script task.

3. Define the data association or transformation to assign the value to the project data object.

   See Section 8.13, "Introduction to Data Associations" for information on how to define a data association.

   See Section 8.14, "Introduction to Transformations" for information on how to define a transformation.

8.10 Introduction to Arguments

You can use arguments to pass data between the different components in a process.

A component may require you to provide certain data when you invoke it. To pass this data you use input arguments.

When you run a component, it provides it results through its output arguments.

The process components that may have arguments are:

- **Service Operations**: may require data to process and may provide data that contains the results of running them, the input and output arguments of the component represent this data.
- **Human Tasks**: may require data to run and may provide data that contains the results of running them, the input and output arguments of the Human Task represent this data.
- **Business Rules**: require an input that they use to evaluate the rules they contain, they return the result of this evaluation using output arguments. When you run a Business Rule using a business rule task it uses the input and output arguments to invoke the selected decision function.
- **Message Start Events**: enable you to define input arguments. You can add input arguments to a start event when a process is used as a subprocess and it receives data from the invoking process. These input arguments represent the data that a process requires when another process invokes it.
- **Message End Events**: enable you to define output arguments. You can add input arguments to an end event, when a process is used as a subprocess and passes information to the process that invokes it. These output arguments represent the data that result from running the process.
- **Catch Events**: allow you to define input and output arguments that define the process interface. If the operation they expose is asynchronous, then you can only define input arguments. If the operation they expose is synchronous, then you can define input and output arguments.
- **Throw Events**: enable you to define input and output arguments that define the process interface.If the operation they expose is asynchronous, then you can only define output arguments. If the operation they expose is synchronous, then you can define input and output arguments.

8.11 Naming Conventions

When you name a process data object, a project data object or an argument, you should respect the following rules:

- Use one or more nouns, or nouns modified by adjectives.
- Do not start the name with a number.

- Use capital letters only to distinguish internal words.
- Keep names simple and descriptive.
- Use whole words, avoid using acronyms, unless they are widely known.
- Avoid using the same name for a process data object and a project data object.

8.12 Scope and Access

The scope an access varies according to the structure used to store information:

- **Process Data Objects**: You can access them from any task within the process. The Process Engine creates them when it creates an instance in the process. Generally the process data objects have different values for each instance in the process. After the instance arrives to the end event, you cannot access process data objects anymore.
- **Subprocess Data Objects**: You can access them from any task within a subprocess. The Process Engine creates them when the subprocess is triggered. After the instance leaves the subprocess, these data objects are no longer available.
- **Project Data Objects**: You can define project data objects at a project level, however the scope of project data objects is a process. Project data objects are predefined for all the processes in a BPM project. The value of a project data object may vary between processes. Generally project data objects have different values for each instance in the process. You can access project data objects from any process in a project, however the value assigned to it during a process is lost when the process finishes running. Figure 7-12 shows the difference between the scope and the life span of project data objects.
- **Arguments**: You can only access arguments from within data associations. You use arguments to pass information between processes or process components. When the Process Engine runs a process or a process element that contains a data association, it maps the value of the arguments to the data objects defined in the data association.

Figure 8–3 Scope of the Data Structures in a Process

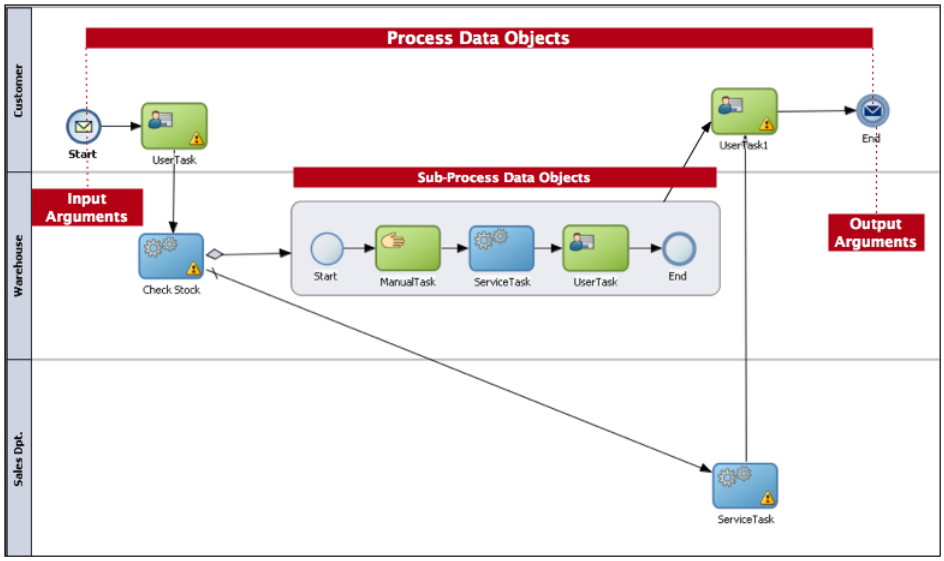
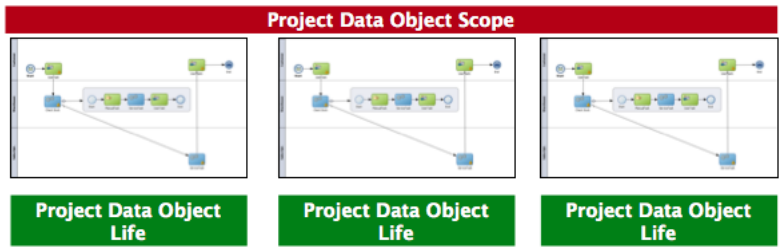


Figure 8–4 Scope and Life Span of Project Data Objects



8.13 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 8–4 lists the flow objects where you can define data associations. It also lists the objects implemented.

Table 8–3 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

You can use data associations to define the input and output from a flow object to an external service or process.

It is important to note that although the inputs and outputs are defined in the data associations for a flow object, the defined values are passed to the implemented systems and services.

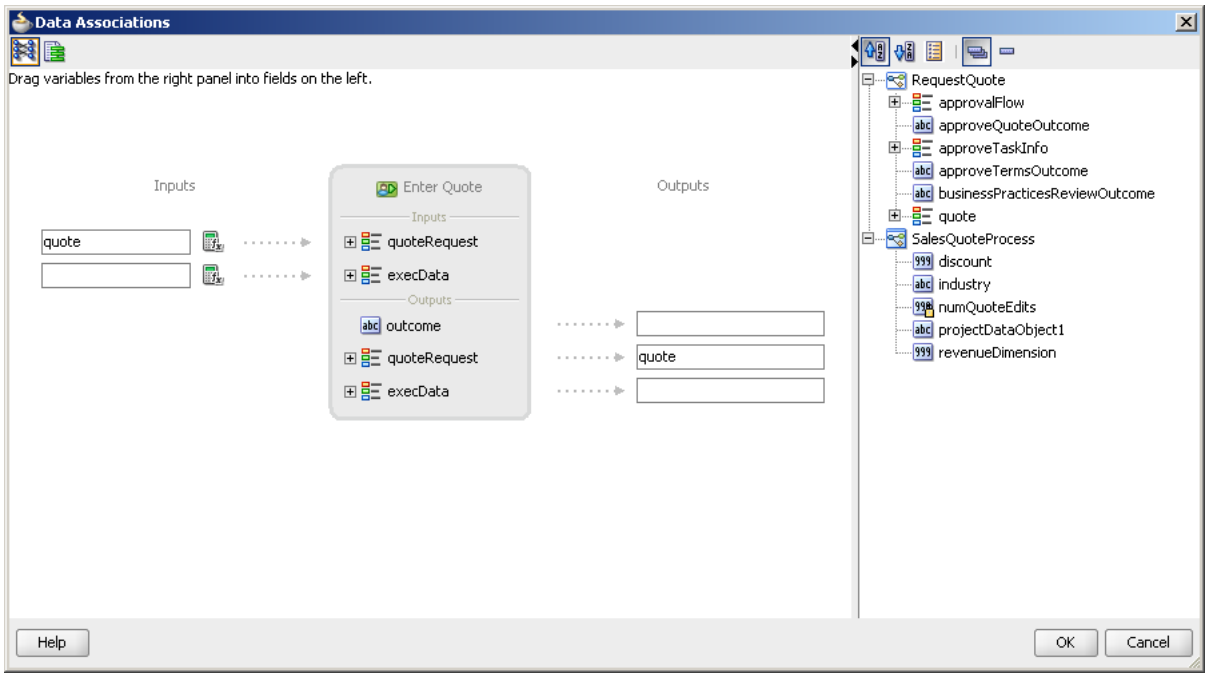
You can use expressions to evaluate and change the input and output values

8.13.1 Introduction to the Data Association Editor

The data associations editor enables you to configure the input and output values passed between a flow object and a its implementation.

Figure 8–5 shows the data association for the Enter Quote user task in the Sales Quote example.

Figure 8–5 The Data Association Editor



This figure shows the data association for the Enter Quote user task in the Sales Quote example. The Enter Quote task implementation requires a Quote object as an input argument and return a modified Quote object as a result of running the Human Task.

****

Table 8–4 describes the different areas of the data association editor.

Table 8–4 The Data Association 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.

8.14 Introduction to Transformations

You can use XSL transformations to transform:

- the values of the data objects in the process, before you pass them to the implementation of a flow object as input arguments.
- the values of the output arguments of a flow object implementation, before you assign them to the data objects in your process.

You can combine the use of transformations with the use of data associations only if you apply them over different arguments.

> **Note:** You must not use transformations and data associations to map the value of an argument simultaneously.

When you define the transformation you can only use as sources data objects that are based on an business object created using an XML schema or type.

You can edit the transformations you create using the SOA XLS Editor. See *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite* for more information on how to use the SOA XLS Editor.

8.15 Defining Transformations

You can modify the values you use assign to input arguments and output arguments in the implementation of a flow object using XSL transformations.

8.15.1 How to Define a Transformation

You can define an XSLT transformation to transform the data you pass to and from the implementation of a flow object.

To Define a transformation:

1. Edit the flow object implementation properties.
2. Select **Use Transformation**.
3. Click the **Edit XSL Transformation** button.

   The Transformation dialog appears.
4. Click the input or output according to the argument value to transform.
5. Click **Add**.

   The Transformation dialog appears.
6. From the Sources List, select a source.

   The sources list only contains data objects that are based on a business object created using an XML schema or type.
7. Click **Add**.

   The source appears in the Selected Elements list.
8. From the Target list, select a target to assign the result of the transformation.
9. In the Transformation section select a way to define the transformation:
   - Create: creates a new transformation and opens the SOA transformation editor for you to define the transformation.
   - Use Existing: enables you to select an existing transformation that you copied to the project XSL directory.

8.15.2 What Happens When You Define a Transformation

The BPMN Service Engine uses the specified XSL transformation to assign the values of the input an output arguments of a flow object. The XSL transformation modifies the values before assigning them.

9

Importing BPMN Processes from a BPA Repository

This chapter describes how to create a BPM project using a BPMN process stored in a BPA repository. Importing BPA projects to Oracle BPM enables Process analysts to develop a project using Oracle BPA and then hand it over to process developers for them to complete the implementation details using Oracle BPM Studio.

This chapter includes the following sections:

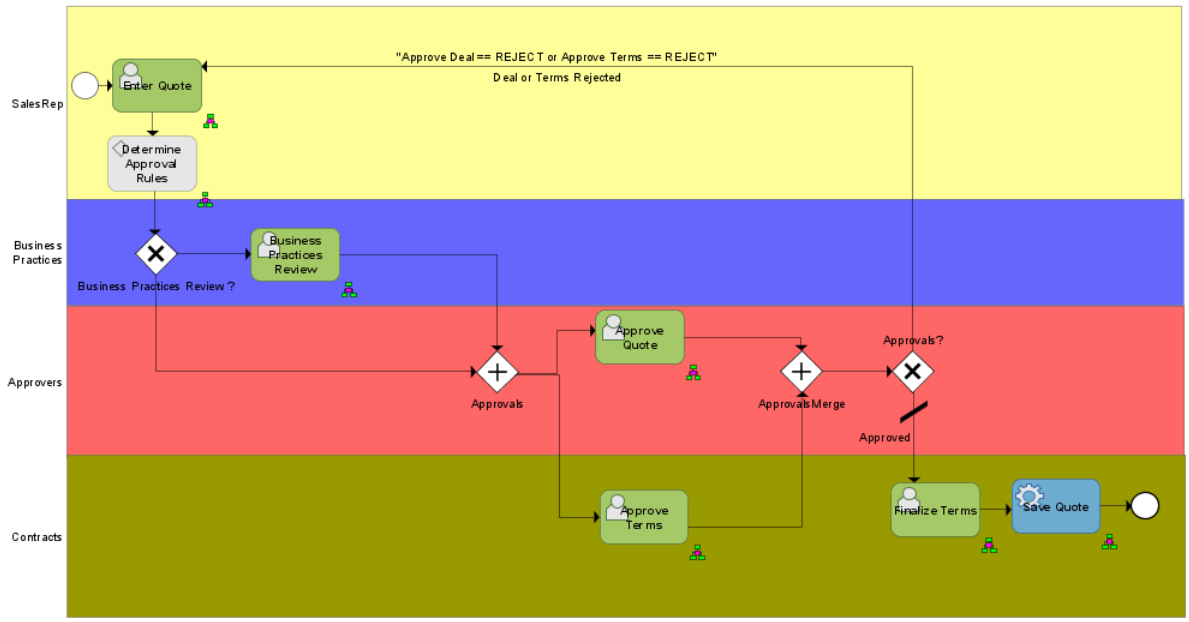
9.1 Introduction to Importing Processes from the BPA Repository

Oracle BPM Studio enables you to create a BPM project based on a BPMN process stored in a BPA repository.

You can create and implement a BPMN process in BPA and then import it to Oracle BPM. This creates a BPM Project that contains the BPMN process and all of the components used to implement it, such as Human Tasks, Service Adapters and Mediators.

Figure 9–1 shows a BPMN model of the Sales Quote example implemented in BPA.

Figure 9–1 BPMN Model of the Sales Quote example process implemented using BPA Architect



This figure shows a BPM Model of the Sales Quote example created using the BPA Architect application.

****

9.2 Creating a BPM Project from a BPA Project

You can create a BPMN model using the BPA Architect application and import it to Oracle BPM. Oracle BPM Studio creates a BPM project that contains the BPMN process you selected and the components used to implement the activities in the BPMN model.

Before importing a BPA project to BPM, ensure your BPA project respects the following rules:

- If the Catalog and Repository information are missing in the BPA project, then the process created by importing the BPA project to Oracle BPM contains unimplemented business rule tasks.
- If the Human Tasks in the BPA project are not associated with a swimlane, then importing the BPA project to Oracle BPM causes an error because the Human Task is not associated with a valid role.
- It does not contain nested subprocesses.

Table 9–1 shows how the different BPA constructs are translated to BPMN constructs when importing a project. Gateways and events remain the same. Note that since Oracle BPM Suite does not support pools and message flows, you can only translate one BPA tool at a time.

Table 9–1 Correspondence Between BPA and BPMN constructs

BPA Construct	BPMN Construct
Automated Activity	Service Task
Human Task	User Task
Rules Task	Rules Task

Table 9–1 (Cont.) Correspondence Between BPA and BPMN constructs

BPA Construct	BPMN Construct
Notification Task	FYI Task

9.2.1 How to Configure a BPA Project to Use It from Oracle BPM

If you want to use a BPA project to create a BPM project in Oracle BPM, then you must configure your BPA project to access it from Oracle BPM.

To configure a BPA project to use it from Oracle BPM:

1. In the BPA Application, select **SOA**.
2. Select **Share Blueprint**.

   A validation dialog appears.
3. Click **Yes** to validate the project, or **No** to continue without validating.

   A dialog that displays the progress of the conversion appears.
4. Click **OK**.

9.2.2 How to Create a BPM Project from a BPA Project

Before following this procedure ensure that your BPA project is configured to use it from Oracle BPM. See Section 9.2.1, "How to Configure a BPA Project to Use It from Oracle BPM" for information on how to configure a BPA project to use from Oracle BPM.

To create a BPM project from a BPA project:

1. In Oracle BPM Studio, select **File** and then select **New**.

   The New Gallery dialog appears.
2. In the Categories tree, select **BPM Tier**.
3. In the Items list, select **Create Project from BPA**.
4. Click **OK**.

   The Create Project from a BPA Project wizard appears.
5. Enter a name to identify the BPM project.
6. Enter a directory to store the project, or click the browse button to browse and select one.
7. Click the **Browse** button next to the BPA Server field.

   The Create BPA Blueprint Composite dialog appears.
8. If the BPA Servers list is empty, then you must add a new BPA server.

   See Section 9.2.3, "How to Add a BPA Server" for information on how to add a BPA server.
9. In the BPA Servers list, expand a server node and its subfolders until the BPMN processes appear.
10. Select a BPMN process.
11. Click **OK**.

The Create BPA Blueprint Composite dialog closes and the selected server appears in the Server field.

12. Click **OK**.

    The Summary page appears.

13. Click **OK**.

    The BPM project you created from the BPMN model you selected from the BPA repository appears in Oracle BPM Studio.

> **Note:** You must configure the implementation of certain BPMN flow objects, such as gateways, business rules, and human tasks, in a BPM project created from a BPA project. Not doing so causes errors when building the BPM Project.

9.2.3 How to Add a BPA Server

You must define a BPA server from which Oracle BPM obtains the BPA project. After you define a BPA server you can reuse it for all the BPA projects that reside in that BPA server to import them to Oracle BPM.

To add a BPA server:

1. Click the **Create BPA Server Connection** button.

   The BPA Server Connection dialog appears.

2. In the Name field, enter a name to identify the connection.
3. From the Location list, select the type of location that corresponds to your BPA Architect configuration.

   The available location types are:

   - **Local Server**: the BPA Architect application directly connects to the BPA repository.
   - **Remote Server**: the BPA Architect application connects to BPA repository server, which in turn connects to the BPA repository.

4. From the Database list, select the database that contains the BPMN process.
5. From the Locale list, select the locale of your BPA project.
6. In the Username field, enter the username for the BPA Architect user.
7. In the Password field, enter the password for the BPA Architect user.
8. If you want to test the connection, then you must click the **Test** tab and then click **Test Connection**.

   The Connection Diagnosis field shows you a message with the result of the test.

9. Click **OK**.

   The BPA Server Connection dialog closes and the new connection appears in the BPA Servers list in the Create BPA Blueprint Composite dialog.

9.2.4 What Happens When You Create a BPM Project from a BPA Project

Oracle BPM Studio creates a BPM project that contains the BPMN model you chose to import. The BPMN project contains an SOA Composite with the components used to

implement the BPMN model such as Human Tasks, Service Adapters, and Mediators. These components also appear in the business catalog as any component in the SOA Composite. See Chapter 12, "Using the Business Catalog" for more information on how components are organized in the business catalog.

Note that you must configure the implementation of certain BPMN flow objects, such as gateways, business rules, and human tasks, in a BPM project created from a BPA project. Not doing so, causes errors when building the BPM Project.

Part IV

Analyzing Process Performance

This part describes how to use simulations and Process Analytics to analyze the performance of your business process.

This part contains the following chapters:

10

Running Simulations in Oracle BPM

By running simulations, business analysts and developers can predict the behavior of business processes under specified conditions. They can run simulations to verify that the output meets the metric objectives and identify any bottlenecks. They can also run simulations to test the effects of changes on an existing process design.

This chapter includes the following sections:

10.1 Introduction to Running Simulations in Oracle BPM

After you define a simulation, you run it in Oracle BPM to determine the efficiency of that definition. Your simulation can reflect either real or anticipated data.

Using the simulation capabilities of Oracle BPM, you can:

- Define multiple models for a given process so that different conditions can be analyzed
- Run multi-process simulations to learn how working in different business processes can affect resources

10.1.1 Simulation Models and Simulation Definitions

Before you run a simulation, you must specify the behavior of each element of your process. To define a simulation you must create and configure the following elements in your BPM Project:

- **Simulation model**

  Enables you to define the behavior for an individual process model. Note that, for any given process model, you can have multiple simulation models, so that you can mimic a variety of scenarios.

- **Simulation definition**

  Enables you to define the processes and resources that define a simulation scenario.In a simulation definition you specify the processes that participate in the

simulation by selecting the simulation models associated to those processes. A process may have multiple simulation models defined for it. If a process has multiple simulations models defined, then you must select one of those models to use in the simulation definition.

Simulations do not call each individual task within a process. For example, they do not run the service associated to a service task, variables are not assigned values, and external resources are not updated.

However, simulations mimic the behavior of an activity using the following simulation variables that you can define using the Simulation Editor:

- duration
- resources
- cost
- queue information
- probability of the instance passing through a sequence flow

10.2 Creating Simulation Models

Simulation models enable you to simulate the behavior of an individual process. They enable you to define how a process behaves as part of a simulation definition.

You can define multiple simulation models for each process, creating different simulations based on different combinations of resource allocation and activity behavior.

10.2.1 How to Create and Configure a Simulation Model

To create and configure a simulation model:

1. In the **BPM Project Navigator**, expand the **Simulations** node.:
2. Right-click the **Simulation Models** node.
3. Select **New Simulation Model**.

   The Create Simulation Model dialog box appears.
4. Select a process.
5. Enter a name for the simulation model.
6. Click **OK**.

   The simulation model appears under the Simulation Models node in the BPM Project Navigator and the Simulation Model editor opens.
7. Specify the number of instances to create during the simulation:
   a. Select **Specify number of process instances to be created**.
   b. Specify the number of instances to create during the simulation.

   Note: The process simulation runs until the completes the specified duration or reaches the maximum number of instances.
8. In the **Flow Nodes** tree, select an activity.

Depending on which type of activity you select, the following tab pages appear on the right of the Flow Nodes tree:

- Duration: defines the distribution that determines the time an activity takes to complete.
- Resources: specifies the number of participants assigned to a particular role. You can define this parameter in the simulation model or at a global project level in the project simulation definition.
- Cost: specifies the cost of processing the activity. For user tasks it also specifies the cost of the resources assigned to the user task.
- Queue Info: defines the simulated behavior of how process instances are queued for a given activity
- Sequence Flows: determines the probability percentage of instances routed through the different outgoing sequence flows

9. Configure each activity as follows:

   a. In the Duration page, in the Instance Execution Duration section, specify the Distribution Type. Options are listed and described in Table 10–1.

Table 10–1 Options in the Simulation Model Flow Nodes Duration Page

Option	Description
Constant	Specifies that the simulation model uses the value specifies in the Period property to calculate the completion time for all the activities in the process.
Uniform	Determines the period required to complete an activity consistently, taking into account the variation specified in the delta property. When you select this option, you are prompted to specify each of the following: ■ Mean: Determines the mean time it takes to complete an activity ■ Delta: Defines the upper and lower limit variation of the mean parameter when determining how long it takes to complete a simulated activity
Exponential	Determines how long it takes to complete a simulated activity by specifying how many instances are completed within a specific period. When you select this option, you are prompted to specify: ■ Average Frequency: Determines the number of average instances processed within the interval defined by the Every property ■ Every: Defines the interval used for exponential distribution
Normal	Uses the Gauss Bell distribution to determine how long a simulated activity takes to complete. You must specify the mean and standard distribution. When you select this option, you are prompted to specify: ■ Mean: The mean period required to perform an activity ■ Standard Deviation: The standard deviation of the mean period required to perform an activity

Table 10–1 (Cont.) Options in the Simulation Model Flow Nodes Duration Page

Option	Description
Real	Enables you to specify the amount of time required to complete a simulated activity for a specific time interval. When you select this option, you must specify: ▪ Distribution Criteria: determines the time interval for determining how long a simulated activity takes to complete ▪ Interval: specifies the period during which the simulation runs. ▪ Mean: defines the mean time to complete an activity ▪ Standard Deviation: defines the standard deviation of the mean parameter

b. In the Cost page, specify the following:

Table 10–2 Properties in the Simulation Model Flow Nodes Cost Page

Property	Description
Fixed Base Cost	Defines the cost required to perform the simulated activity
Fixed Base Cost Plus Resource Cost	Calculated based on the defined cost per hour and the time it takes the resource to execute the instance. This property is only available for user tasks.

c. In the Queue Info page, specify the following:

Table 10–3 Properties in the Simulation Model Flow Nodes Queue Info Page

Queue Warning Size	Determines the number of incoming instances that can be waiting for an activity simultaneously.
Activity Queue Policy	Determines how incoming instances are handled by the activity. The following values are available: F.I.F.O (First In, First Out), L.I.F.O (Last In, First Out), Random, and By Priority

d. In the Sequence Flows page, move the slider to specify the probability of each outgoing sequence flow occurring.

If the activity contains boundary events, then the flows of the boundary events appear in this page. For more information about configuring boundary events, see Section 10.3, "Configuring Boundary Events".

10.3 Configuring Boundary Events

If the BPMN process you want to simulate contains boundary events, then you must specify the probability of these events happening.

You can specify the probabilities for the different boundary events in the Outgoing Flows page.

The way you specify the probability of a boundary event varies according to the type of the event:

- **Interrupting Boundary Message and Error Events**

The probabilities of all the interrupting boundary message and error events for an activity are related. If you add these probabilities the result must always be 1.

The simulation model editor displays a set of sliders to configure these activities. If you move a slider, the values in the other sliders automatically adjust. You can lock the values by clicking the lock icon next to the slider. When you lock a value the simulations model editor does not modify it when you move the other sliders. The simulation model editor forces you to leave at least two values unlocked.

- **Non-Interrupting Boundary Message and Error Events**

  The probability of a non-interrupting boundary message or error event is independent from the probability of other events happening. This value of this probability can vary between 0 and 1.

  The simulation model editor displays a slider for each non-interrupting boundary message or error event. You can move this slider to specify any value between 0 and 1.

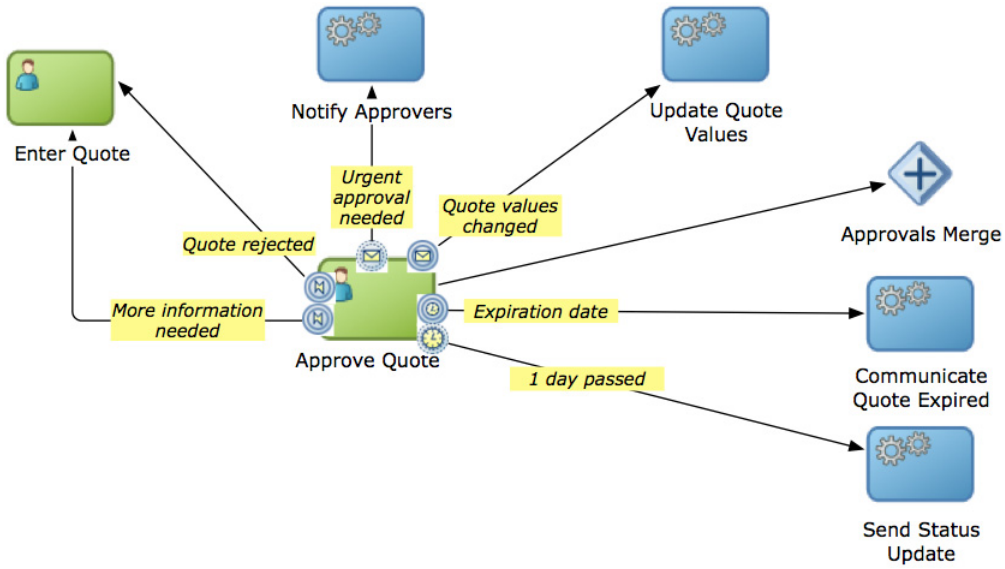
- **Timer Events**

  To specify the probability of a boundary timer event, you must define the time interval between occurrences of the event. If the implementation of the timer event in the BPMN process uses an expression, then you must define a fixed time interval to use during the simulation. If the implementation of the timer event in the BPMN processes uses a fixed time interval, then redefining the time interval is optional because you can use the interval defined in the BPMN process for the simulation.

  The simulation model editor displays a table for interrupting timer events and another one for non-interrupting timer events. You can redefine the time intervals for each of the events using these tables.

Figure 10–1 shows the Approve Quote user task with different types of boundary events. Figure 10–2 shows the simulation configuration page for the Approve Quote user task.

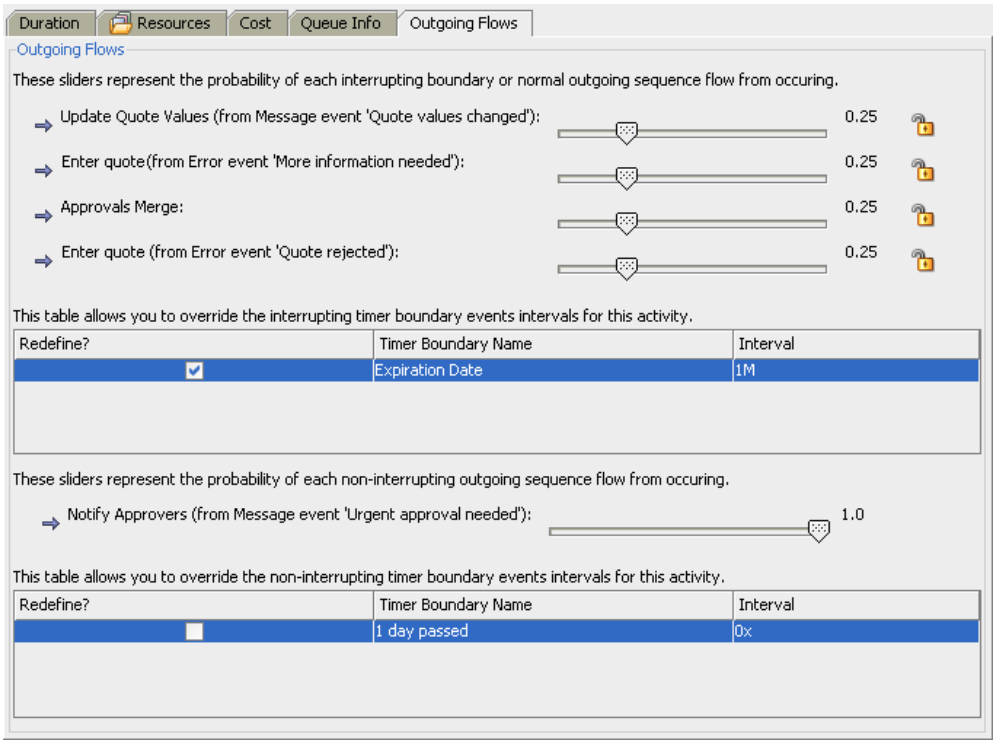
Figure 10–1 The Approve Quote user task with multiple boundary events



This figure shows the Approve Quote user task with different types of boundary events. The 'More information needed' and 'Quote rejected' error events and the 'Quote values changed' message events are interrupting boundary events. The 'Urgent approval needed' event is a non interrupting message event. The 'Expiration date' and '1 day passed' events are timer events.

****

Figure 10–2 Outgoing Flows tab page for the Approve Quote task



This figure shows the Outgoing Flows page for the Approve Quote user task. The first section enables you to configure probability for the non-interrupting message and error events. The table below enables you to specify the time interval for the interrupting timer events. The slider below enables you to specify the probability for the only non-interrupting message event for this activity. And the last table enables you to specify the time interval for the non-interrupting timer event.

****

10.4 Creating Simulation Definitions

You can create a simulation definition to represent a simulation scenario for a group of simulation models. You can select which simulations model to run from the group of simulation model contained in the simulation definition.

10.4.1 How to Create a Simulation Definition

In a simulation definition, you can customize the following parameters to see how they influence the performance of your project:

- Start time and duration of the simulation
- Which process simulation models you want to include in the project simulation

- The participant resources you want to include in the simulation

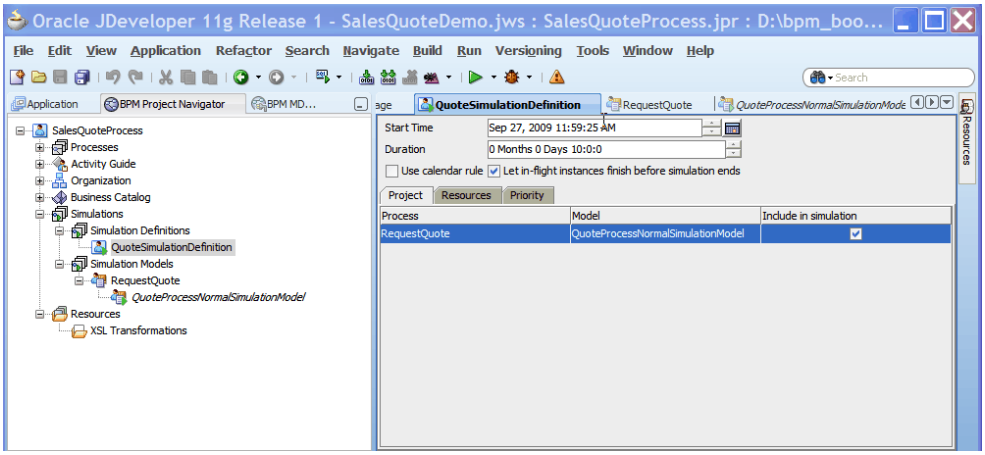
To create and configure a simulation definition:

1. In the BPM Project Navigator, expand the **Simulations** node.
2. Right-click the **Simulation Definitions** node.
3. Select **New Simulation**.

   The Create Simulation Definition dialog box appears.
4. Enter a name for the simulation definition.
5. Click OK.

   The simulation definition appears under the Simulation Definitions node and the Simulation Definition editor opens. Figure 10–3 shows the Simulation Definition editor.

Figure 10–3 Simulation Definitions Project Page



This graphic shows the BPM Project Navigator on the left. The simulation definition being configured is selected. On the right, the corresponding details page for that simulation definition is displayed. Within that details page, there are three tab pages: Project, Resources, and Priority. The Project tab page is selected. The particular elements of both the details page and the Project page are described in the surrounding text.

****

6. Specify the general parameters for this simulation as described in Table 10–4.

Table 10–4 General Parameters for Simulation Definitions

Parameter	Description
Start Time	Defines the start time for the simulation. This time is used only for logging. It is not used for scheduling purposes.
Duration	Defines the period the simulation runs. This interval is specified in months, days, hours, minutes, and seconds.
Let in-flight instances finish before simulation ends	If selected, simulation ends only when the specified number of instances completes. If unselected, simulation stops after the simulation duration is completed. At that point, all incomplete instances are shown in either “in-process” or “queue” status.

7. The Project page contains a table that lists all of the processes within the current project. For each process, you can select which simulation model you want to use. Also, you must specify which processes to include in the simulation.

   Specify the parameters in the Project page as described in Table 10–5.

Table 10–5 Project Parameters for Simulation Definitions

Parameter	Description
Process	Lists the processes that you can include in this simulation.
Model	For each process, lists the model specified in Section 10.2.1, "How to Create and Configure a Simulation Model"
Include in Simulation	Enables you to specify whether to include the process in the simulation

   After you specified the parameters in the Project page, select the **Resources** tab.

8. In the Resources page, you can define the resources to use within the simulation.

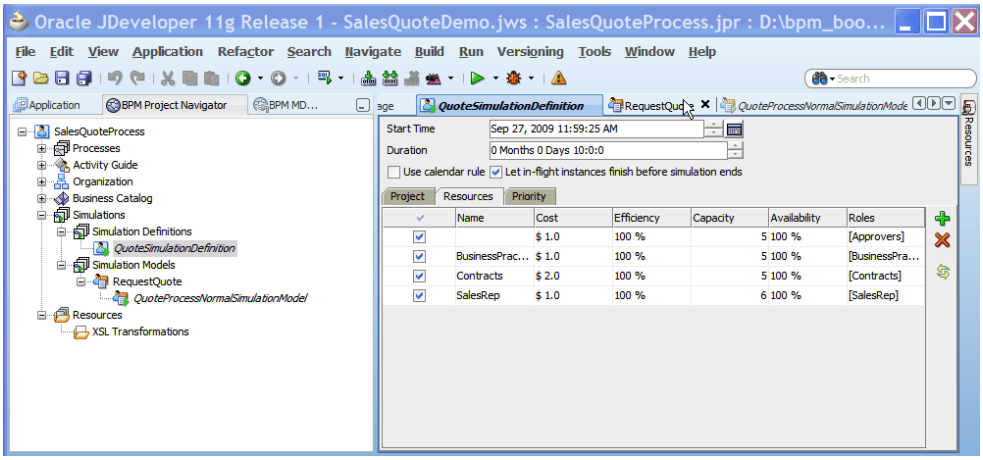
   All processes included in the simulation share these resources. The cost of each resource is defined per hour.

   To define the resources click the following buttons:

   - **Add Resource**: adds a resource to the simulation definition
   - **Delete Resource**: deletes the selected resource from the simulation definition

   Figure 10–4 shows an example of the Resources page.

Figure 10–4 Simulation Definitions Resources Page



This graphic shows the BPM Project Navigator on the left. The simulation definition being configured is selected. On the right, the corresponding details page for that simulation definition is displayed. Within that details page, there are three tab pages: Project, Resources, and Priority. The Resources tab page is selected. The particular elements of both the details page and the Resources page are described in the surrounding text.

****

9. Click **Save**.

Note: Ensure you saved the changes before running the simulation. If you do not save the changes, then the simulation engine does not use them when you run the simulation.

10.5 Running Simulations

You can pause, stop, or run a simulation to the end. If you stop the simulation, you must restart it from the beginning.

10.5.1 How to Run a Simulation

To run a simulation, you must have created simulation models and at least one simulation definition.

To run a simulation:

1. In the BPM Project Navigator:
   a. Open the **Simulation** view.
   b. From the **Simulation** list, select the simulation definition you want to run.
   c. Click **Run Simulation**.

      The simulation begins.

10.5.2 What Happens When You Run a Simulation

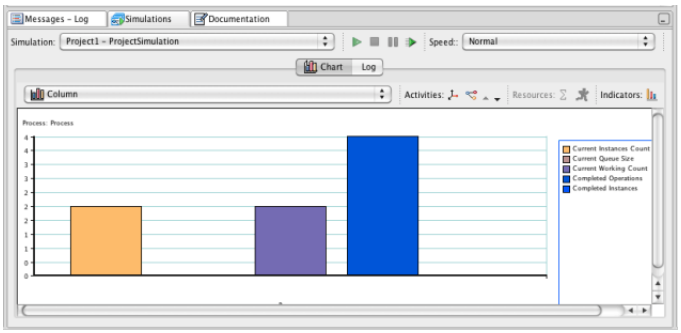
The animation of the simulation appears in the project editor, and the results appear according to your specifications in the Simulation page.

Note: If you place the mouse pointer over a column in the chart, a tooltip with the value of the activity or indicator appears.

10.5.3 Understanding the Simulation View

The simulation view enables you to configure and run a simulation. Figure 10–5 shows a simulation view.

Figure 10–5 Example of a Simulations Page



This illustration is described in the surrounding text.

****

The toolbar on the Simulations view enables you to:

- Select which simulation to run from the Simulation list.
- Start, stop, and pause the simulation, or run it to the end by clicking the appropriate button. If you choose to run the simulation to the end, the simulation runs in the background with no animation making the simulation faster.

> **Note:** If you stop a simulation, you must restart it from the beginning.

- Select the speed at which to run the simulation from the Speed list. In normal speed, instances are created at rate of one per second.

10.6 Analyzing the Results of a Simulation

You can display simulation results either as a chart or as a log file by clicking either the Chart tab or the Log tab in the Simulations window.

The Log tab displays a log that tracks the movements of all the instances in the simulated process. Each line in the log contains the following information:

- Date and Time
- Process
- Instance
- Instance path

10.6.1 How to Analyze the Results of a Simulation Using a Chart

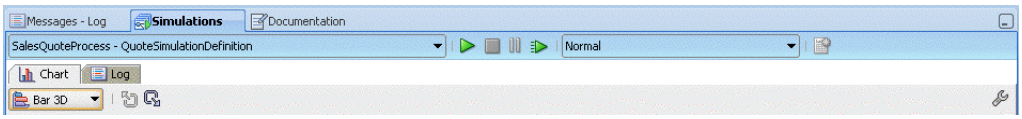
The Chart tab enables you to select a type of chart to display the result of the simulation. You can configure this chart to display the resources to monitor. You can also select the units the chart uses to measure the resources use.

In the Chart tab you can configure how to display the chart with the results of the simulation by configuring the following:

- Type of chart
- Activities or resources to monitor
- Indicators

Figure 10–6 shows the toolbar for a sample Chart page.

Figure 10–6 Simulations Chart Page



This illustration is described in the text.

****

To analyze the results of the simulation using a chart:

1. From the list below the Chart tab, select the type of chart to display.

They available types are:

- Column
- Bar
- Bar 3D
- Column 3D
- Table

2. Click the **Configure** icon located on the right hand side of the Charts tab.

   A Configuration dialog box appears.

3. Select a resource or an activity to monitor.
4. Select the axis where to display the activities or resources.
5. From the list below the Show list, select the activities or resources to monitor in the simulation.
6. From the **Indicators** list, select the type of indicators to monitor.

   The available types of indicators are:

   - Cost
   - Time
   - Units

7. From the list below the indicators list, select the indicators to monitor.

   The chart displays the variables and indicators you selected.

8. Click **Close**.
9. Optionally, click the drill up and drill down icons located next to the Types list to increase or reduce the level of detail in the chart.

10.6.2 How to Generate a Simulation Report

You can generate a simulation report that contains the result of the simulation.

To generate a simulation report:

1. Run the simulation.

   For more information on how to run a simulation, see Section 10.5.1, "How to Run a Simulation".

2. Click the **Generate Documentation** icon.

   The Simulation Report dialog box appears.

3. In the Directory Location field, enter a directory to store the report or click the button next to it to browse the file system and select a directory.
4. In the Activities tab, select an option:

   - Summary
   - Details

5. In the tree below, select the activities to include in the report.

   You can specify which activities to include using the following options:

- Select All to include all the processes in the simulation definition.
- Select a process to include all the activities in the selected process.
- Individually select the activities to include.

6. In the indicators tab, select an option:
   - Summary
   - Details
7. In the tree below, select the indicators to include in the report.

   You can individually select which indicators to display in the report, or select a type of indicator to display all the indicators of that type.
8. Select the type of graphic to use in the report.

   The preview are shows an example of the graphic you chose.
9. Click **OK**.

10.6.3 What Happens when You Generate a Simulation Report

Oracle BPM Studio creates a directory using the name and location you selected. This directory contains an HMTL file for each of the processes in the simulation.

The HTML file contains:

- a graphic that displays the result of the simulation
- a link to a CSV file with the simulation data
- a link to a CSV file with the simulation resources data

You can view the CSV files with the simulation data and resources data in a spreadsheet application.

11

Using Process Analytics

This chapter describes how to use and configure BPM Process Analytics to monitor the activity of the processes in your project. Process Analytics enable you to obtain performance and workload metrics of the processes in your project. You can use this metrics to make decisions about your process.

This chapter includes the following sections:

11.1 Introduction to Process Analytics

Business Process Analytics enables you to monitor the performance of your deployed processes. It measures the key performance indicators in your project and stores them in a database. Process analysts can view the metrics stored in the Process Analytics databases using WorkSpace dashboards or Oracle BAM, depending on the database you select to store the information.

Process analysts can monitor standard pre-defined metrics and process specific user-defined metrics. Process developers can define process specific metrics using Business Indicators. Business Indicators are a special type of project data object that the BPMN Service Engine stores to the Process Analytics databases when it runs the BPMN processes.

Process developers define the key performance indicators you want to monitor while developing your process. After publishing the application business analysts can use the default dashboards BPM WorkSpace provide or create custom dashboards to view the metrics the BPMN Service Engine gathered while running BPMN processes.

Process Analytics track:

- Process and Activity Performance Metrics
- Workload Metrics

- Human Resource Metrics

You can store the key performance indicators in your process using business indicators. By default the BPMN Service Engine stores the values of pre-defined measures and dimensions that are common to all BPMN processes.

The supported pre-defined measures are:

- Number of active instances by activity, process, and participant
- Average time to complete an activity
- Average time to complete a process

The supported pre-defined dimensions are:

- Process
- Activity
- Participant

You can also define custom measures according to your needs. To define custom measures you use business indicators. The different types of business indicators enable you to measure specific values, keep track of categories or count the times an instance completes one or more activities.

Oracle BPM provides you with a set of pre-defined cubes you can use to store the Process Analytics data. Cubes are a structure used to organize a database so that it enables you to analyze data in real time and view it from multiple perspectives.

You can also choose to store these data to Oracle BAM or use both systems simultaneously.

11.1.1 Process and Activity Performance Metrics

Process analytics track the time a process takes to complete and the average time each of the flow objects in that process take to complete.

Process performance metrics track the time an instance takes to run that process from the start to the end event.

Activity performance metrics track the time that passes from the moment the process instance arrives at a flow object until it moves to the next flow object in the process.

Note that when the flow object invokes a synchronous service operation, activity performance metrics include the time it takes to run the synchronous service operation because the process instance does not leave the flow object until it receives an answer from the service. However when the invoked service operation is asynchronous, activity performance metrics do not include the time it takes to run the service operation because the process instance leaves the process after invoking the service without waiting for the service to complete.

11.1.2 Workload Metrics

Process analytics track the number of instances sitting in each activity at a certain time. You can view the workload for a certain process, activity or instance.

Oracle BPM takes snapshots at fixed intervals and stores the number of instances and the value of the business indicators at that moment. To obtain the current workload in the process you must select the information from the most recent snapshot.

11.1.3 Human Resource Metrics

You can view performance and workload metrics filtered by participant. This enables you to monitor the workload and performance of the different participants in the BPMN process.

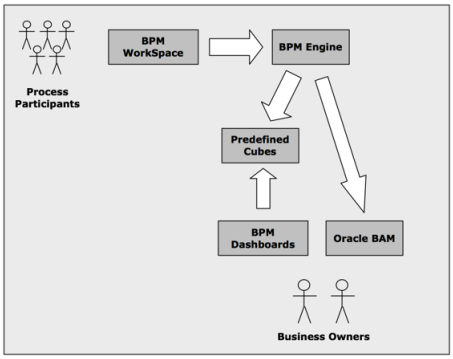
11.2 Typical Process Analytics Workflow

The following list describes the typical tasks you perform when you use Process Analytics in a BPM Project:

- Create a BPM project and one or more BPMN processes.
- Configure the sampling points generation for the project or process.
- Configure the project to use BPM Cubes or Oracle BAM, or both.
- Add business indicators to the processes in your project.
- Assign values to the business indicators.
- Add measurement marks or counter marks in the processes where you want to track the value of the business indicators.
- Deploy your project.
- Use BPM Workspace or BAM Architect to configure custom dashboards.

Figure 11–1 shows the cycle the process analytics data goes through after deploying and running a BPMN process.

Figure 11–1 Process Analytics Data Cycle



This diagram describes the cycle the process analytics data goes through. Process participants interact with the BPMN process using BPM WorkSpace. The BPMN Service Engine runs the activities in the BPMN process and stores the values of the business indicators in the process to BPM pre-defined cubes or Oracle BAM, or both. Business owners can monitor the process using Oracle BAM or BPM dashboards based on the pre-defined BPM cubes.

****

11.3 Configuring Projects, Processes and Activities to Generate Sampling Points

When the BPMN Service Engine runs the activity in the process it stores data about the process to the BPM Cubes and Oracle BAM Data Objects. This data comes from the

sampling points defined in the project. You can configure which processes in your project, or which activities in your project generate sampling points in these databases.

You can configure the sampling point generation at the following levels:

- Project
- Process
- Activity

You can configure your project to generate sampling points for all the activities in the processes it contains or only for interactive activities. You can also choose not to generate sampling points for the processes in this project. BPMN processes use this value when they are configured to use the project default settings.

By default, the project is configured to generate sampling points only for interactive activities.

You can configure your processes to use a setting for sampling point generation different from the one defined by the project. Generally you do this to improve the performance of the project. For example, if your project contains a process that contains multiple activities and you are not interesting in obtaining process metrics for this process, then you might choose to configure the process to not generate sampling points. Another example is if you are interested in measuring only one process within your project, then you might choose to configure the project to not generate sampling points and configure that particular process to generate sampling points.

By default, the process is configured to use the project sampling point configuration.

You can also configure one or more of the activities in your process to use a sampling point setting different from the one used in your process. For example, you might choose to configure all the gateway activities in your process to not generate sampling points because you consider these metrics do not provide relevant information.

By default activities are configured to use the process sampling point configuration. If in turn, the process is configured to use the project configuration, then the activities use the configuration the project specifies.

11.3.1 How to Configure the Sampling Point Generation of a Project

You can configure the sampling point generation at the project level.

To configure the sampling point generation of a project:

1. In the BPM Project Navigator, right-click the project you want to configure.
2. Select **Project Preferences**.
3. In the **Process Analytics Summary** tab, in the Project Sample Points section, select an option. Available options are:

Option	Description
Generate Only for Interactive Activities	Generate sampling points only for the user tasks in the processes contained in the project.
Generate for All Activities	Generate sampling points for all the activities in the processes contained in the project.
Do Not Generate	Do not generate sampling points for any of the activities in the processes contained in the project.

4. Click **OK**.

11.3.2 What Happens When You Configure a Project To Generate Sampling Points

All the processes you create within a project use the sampling point configuration defined for that project, unless you edit the process properties to use a different configuration for that specific process.

11.3.3 How to Configure the Sampling Point Generation for a Process

You can configure the sampling point generation at a process level.

To configure sampling point for a Process:

1. In the BPM Project Navigator, right-click the process you want to configure.
2. Select **Properties**.
3. Click the **Advanced** tab.
4. In the Process Sampling Points section, select an option. Available options are:

Option	Description
Inherit Project Default	The process uses the project sampling point configuration to decide if it generates sampling points.
Generate Only for Interactive Activities	Generate sampling points only for the user tasks in the process.
Generate for All Activities	Generate sampling points for all the activities in the process.
Do Not Generate	Do not generate sampling points for any of the activities in the process.

5. Click **OK**.

11.3.4 What Happens When You Configure the Sampling Point Generation for a Process

The BPMN Service Engine uses the defined sampling point configuration to decide whether to store the Process Analytics information, regardless of what the project sampling point configuration indicates. The activities in the process use the process sampling point configuration, unless you edit them to use a different configuration.d

11.3.5 How to Configure the Sampling Point Generation for an Activity

You can configure the sampling point generation at the activity level.

To configure the sampling point generation for an activity:

1. Right-click the activity you want to configure.
2. Select **Properties**.
3. Click the **Basic** tab.
4. Expand the **Sampling Points** section.
5. Select an option. Available options are:

Option	Description
Inherit Process Default	The BPMN Service Engine uses the process sampling point configuration to decide if it generates sampling points for this activity.
Generate	Generate sampling points for this activity.
Do Not Generate	Do not generate sampling points for this activity.

6. Click **OK**.

11.3.6 What Happens When You Configure the Sampling Points for an Activity

The BPMN Service Engine uses the activity sampling point configuration to decide wether to store the Process Analytics information, regardless of what the project and process sampling point configurations indicate.

11.4 Adding Business Indicators to Projects

Business Indicators are project data objects you use to store the value of the key performance indicators of your process. For your convenience business indicators have their own entry in the structure window.

Key performance indicators represent relevant information in your process that can help you determine if your process is running as expected.

The following are examples of common business indicators:

- Order Amount
- Product Stock
- Elapsed Time
- Shipping Status

You can use business indicators to store the value of an indicator you want to measure in your process, or to store a category you want to use to group the values you measured in your process.

According to the type of information you want to store, you can define your business indicator as a:

- Measure
- Dimension
- Counter

The type of business indicator determines the available data types you can use. Table 11–1 shows the available data types for each business indicator type.

Table 11–1 Available Data Types for Business Indicator Types

Business Indicator Type	Allowed Data Types
Dimension	■ String ■ Bool ■ Time ■ Int (with ranges) ■ Real (with ranges) ■ Decimal (with ranges)
Measure	■ Int ■ Real ■ Decimal
Counter	■ Int

Measures

Measures store the value of a key performance indicator that you can measure. Measures only allow data types that are continuous. You must use them with measurement marks. The deal amount and the discount percentage are examples of measures in the Sales Quote process.

Dimensions

Dimension store the value of a key performance indicator that you can use to group the values of the measure business indicators in your process. If you use a continuous data value to define a dimension, then you must add it at least one range. The Process Analytics database only stores the range value if the data value is a continuous one. The deal range and the industry type are examples of dimensions in the Sales Quote process.

Counters

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. For more information on how to configure counters, see Section 11.6, "Adding Counters to the Activities in a Process".

11.4.1 How to Add a Business Indicator to a Project

Business indicators enable you to define the key performance indicators to measure in your project.

To add a business indicator to a project:

1. In the BPM Project Navigator, select the project.
2. In the Structure Window, right-click **Business Indicators** and select **New**.

   The Create Business Indicator dialog appears.
3. Enter a name to identify the business indicator.
4. In the Business Indicator list, select a type of business indicator.

5. From the Type list, select a data type.

   Note: The available data types vary according to the type of business indicator you selected. Table 11–1 shows the available data types for each business indicator type.

6. If you selected a continuous data type and selected Dimension as its business indicator type, then add at least one range.
7. Optionally, check **Auto Initialize** to initialize the business indicator with a default value.

   For more information about default values, see Section 8.2.2, "Default Values".
8. Click **OK**.

   The Create Business Indicator dialog closes and saves the business indicator you created.

11.4.2 What Happens When You Add a Business Indicator to a Process

You can use the business indicators that you added to your project to store data about the processes you want to monitor.

Some business indicators require you to add different artifacts to your process to indicate the BPMN Service Engine must store their values in the Process Analytics databases.

- **Dimensions**

  The BPMN Service Engine automatically stores the data in the dimension business indicators in the pre-defined and custom sampling points defined for your process.

- **Measures**

  Pre-defined measures are always measured in every flow element that is configured to produce sampling points.

  You can add a measurement mark to specify the point, or process sections where you want the BPMN Service Engine to measure and store a custom business indicator of type measurement. For information on how to add a measurement mark, see Section 11.5, "Adding Measurement Marks to Processes".

- **Counters**

  You must add a counter mark to those activities where you want the BPMN Service Engine to store the value of the counter business dimension. For information on how to add a counter mark, see Section 11.6, "Adding Counters to the Activities in a Process".

11.5 Adding Measurement Marks to Processes

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.

You can use one measurement mark to measure multiple business indicators.

Measurement marks store the following data into the Process Analytics databases:

- The value of the process default measures

- The value of the measure business indicators associated to that measurement mark
- The value of the dimensions defined in the process

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 process you might want to view the total amount of quotes approved by region.

The types of measurement marks you can define are:

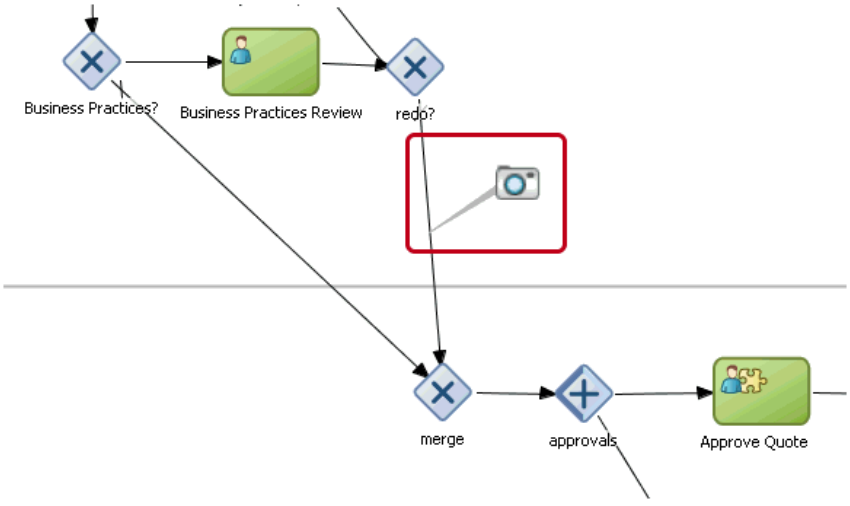
- Single Measurement
- Interval Start
- Interval Stop

Measurement marks are associated to a flow element. Measurement marks of type interval start track the value of business indicators before running the flow elements that proceeds them. Counter marks, measurement marks of type interval stop and single measurement marks track the value of business indicators after running the flow element that precedes them.

Single Measurements

If you defined measure business indicators in your process, then you must add single measurement marks in those points in the process where you want to measure those business indicators. Single measurement marks indicate the BPMN Service Engine that at that point in the process it has to store the value of the measure business indicators associated to that measurement mark. The BPMN Service Engine also stores the values of the default process measures and the dimension business indicators at this point in the process.

Figure 11–2 Single Measurement Mark



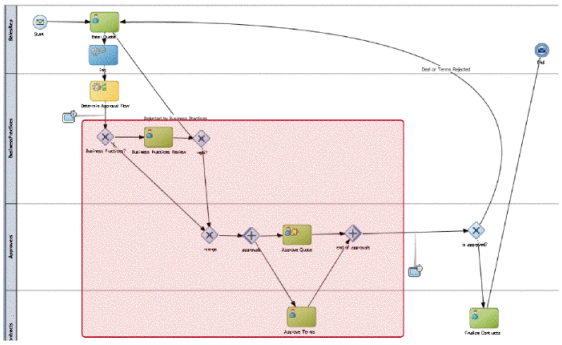
This image shows a single measurement mark placed on a transition between two exclusive gateways. This measurement mark stores the values of its associated business indicators have after the process instance completes the first exclusive gateway.

****

Interval Start and Interval Stop

If you want to measure a business indicator in a section of your process, then you must use an interval start measurement mark to indicate the start of the section and an interval stop measurement mark to indicate the end of the section. These measurement marks enable you to measure the default business indicators or business indicators you defined in a a section of the process. Generally you use these measurement marks to monitor critical sections of your process. For example, you might want to monitor the amount of instances in a part of the process you identified as a bottle-neck.

Figure 11–3 Interval Start and Interval Stop Measurement Marks



This image shows the portion of a process measured by a pair of interval measurement marks. Interval measurement marks measure the process business indicators in all the activities enclosed between the interval start and interval stop measurement marks.

****

11.5.1 How to Add Single Measurement Marks to a Process

You can use single measurement marks to measure business indicators in a specific point in your process.

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 **Measurement**.
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. Select **Single Measurement**.
6. In the **Name** text-field, enter a name to identify the measurement mark.
7. In the Business Indicators section, select a business indicator from the Available list and move it to the Selected list using the arrows between the two lists.

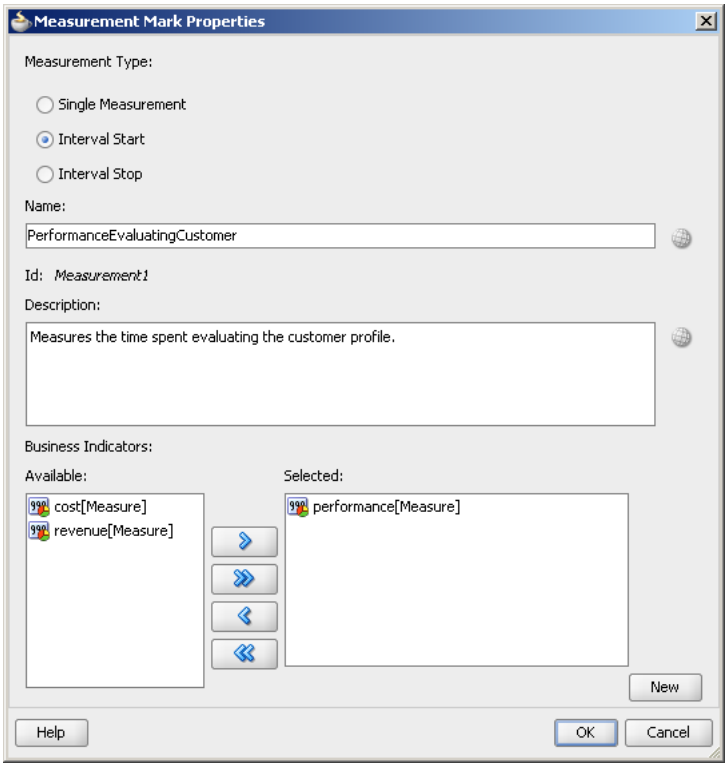
   Figure 11–4 show the Measurement Mark Properties dialog.

> **Note:** You can measure multiple business indicators in the same measurement mark.

Note: If you do not select a business indicator, then Oracle BPM Studio displays a warning message. If you want to add a business indicator without leaving the Measurement Mark Properties dialog, then you can click the **New** button under the **Selected** list.

8. Click **OK**.

Figure 11–4 Measurement Mark Properties Dialog



This figure shows the Measurement Mark Properties Dialog. This dialog enables you to configure the measurement mark type, provide a name and a description for the measurement mark, and select the business indicators to measure.

****

11.5.2 What Happens When You Add a Single Measurement to a Process

When the BPMN Service Engine runs a single measurement mark, it stores the current value of the following business indicators in the Process Analytics databases:

- Business indicators associated to the measurement mark
- Default measurements
- Dimensions defined for the process

11.5.3 How to Measure a Business Indicator in a Process Section Using Measurement Marks

You can use measurement marks to define a section in your process in which to measure certain business indicators.

Note: You must only define one interval stop measurement mark for each interval start measurement mark. Defining multiple interval stop measurement marks is not supported and may cause unexpected behavior.

To measure a business indicator in a process section:

1. Open the BPMN process.
2. In the **Component Palette**, expand the **Artifacts** section and select **Measurement**.
3. Place the measurement mark near the sequence flow where the section of the process begins. When the sequence flow turns blue, drop the measurement mark.
4. Right-click the measurement mark and select **Properties**.
5. Select **Interval Start**.
6. In the **Name** text-field, enter a name to identify the measurement mark.
7. In the Business Indicators section, select a business indicator from the Available list 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 Oracle BPM Studio displays a warning message. If you want to add a business indicator without leaving the Measurement Mark Properties dialog, then you can click the **New** button under the **Selected** list.

8. Click **OK**.
9. In the Component Palette, expand the **Artifacts** section and select **Measurement**.
10. Place the measurement mark near the sequence flow where the section of the process ends. When the sequence flow turns blue, drop the measurement mark.
11. Right-click the measurement mark and select **Properties**.
12. Select **Interval Stop**.

    The **Start Measurements** list replaces the **Name** text-field.
13. From the **Start Measurements** list choose the measurement mark that indicates where the process section begins.
14. Click **OK**.

11.5.4 What Happens When You Measure a Business Indicator in a Process Section Using Measurement Marks

The BPMN Service Engine stores the values of the measure business indicators associated with the pair of measurement marks to the Process Analytics databases. It also stores the values for the dimensions and default measures for that process section. The default measures that contain average values provide information about the average value for that section of the process.

11.6 Adding Counters to the Activities in a Process

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.

Generally you 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.

11.6.1 How to Add a Counter Mark to an Activity in a Process

You can add a counter mark to an activity to track the number of times the instance runs this activity.

To add a counter to an activity in your process:

1. Add a business indicator of type counter to your process.

   For more information on how to add a business indicator, see Section 11.4, "Adding Business Indicators to Projects".

2. Open the BPMN process.
3. Right-click the activity to which you want to add the counter.
4. In the Available list, select the counter business indicator where you want to store the data. Move it to the Selected list using the arrows between the two lists.

   You can add multiple counter business indicators to the same counter mark.

5. Click **OK**.

11.6.2 What Happens When You Add a Counter Mark to an Activity in a Process

When the BPMN Service Engine runs an activity with a counter mark, it increases the value of the counter business indicator associated to that counter mark by one, in the Process Analytics databases.

11.6.3 How to Delete a Counter Mark

You can delete a counter mark that you do not use or need.

To delete a counter mark:

1. Open the BPMN process that contains the counter mark.
2. Right-click the activity that contains the counter mark.
3. Select **Delete Counter Mark**.

11.6.4 What Happens When You Delete a Counter Mark

After you remove the counter mark, when you right-click the activity it does not show the option Delete Counter anymore. Instead, it shows the option New Counter Mark.

Because you removed the counter mark, when an instance passes through that activity the BPMN Service Engine does not increase the value of the counter business indicator in the Process Analytics databases.

11.7 Configuring Cubes Generation in a Project

Oracle BPM provides a set of pre-defined cubes you can use to monitor the activity of your processes. These pre-defined process cubes are enabled by default.

If cubes are enabled then when the BPMN Service Engine runs the processes in your project, then it populates the cubes. You can then view the data stored in these cubes using the dashboards provided by the BPM WorkSpace application. For more information about how to build and view dashboards in BPM WorkSpace, see *Oracle Fusion Middleware User's Guide for Oracle Business Process Management*.

11.7.1 BPM Process Cubes

Process cubes enable process analysts to analyze the process metrics from different perspectives. The perspectives are defined when defining the process. Measures are numeric facts that you can categorize by dimensions. For example, in the Sales Quote example process, you must define a dimension for the product and a measure for the deal amount to analyze the deal amount by product.

BPM Process Cubes support the following dimensions:

- Activity
- Process
- Participant

They also support the following measures:

- Completion time for a specific process
- Completion time for a specific activity
- Number of tasks by participant

Process cubes store the data related to activities in the Task Performance (BPM_CUBE_TASKPERFORMANCE) and Workload(BPM_CUBE_WORKLOAD) tables. It stores the data related to processes in the Process performance (BPM_CUBE_PROCESSPERFORMANCE) and Workload(BPM_CUBE_WORKLOAD) tables.

The workload table contains data about the currently active activity and process instances. The Task and Process performance tables contain data about completed activities and processes respectively.

The data for task and process performance is computed and persisted when an activity or task is completed. The data for workload is calculated and persisted using a timer based on the configuration of the cubeUpdateFrequency property. You can specify the value for the cubeUpdateFrequency property using Enterprise Manager.

All the dimensions and measures are stored for the enabled out of the box sampling points. Selected dimensions, measures and counters are stored for user-defined sampling points.

11.7.2 How to Configure BPM Process Cubes Generation in a Project

If you use BPM process cubes to monitor the performance of your project, then you must enable the generation of the process cubes at developing time. When you deploy your application Oracle BPM uses this configuration to enable BPM Process Cubes.

To enable BPM process cubes in a Project:

1. In the BPM Project Navigator, right-click the project.
2. Select **Project Preferences**.
3. In the Category tree, select **Process Analytics Summary**.
4. In the Process Analytics Summary section, click the **Data Targets** tab.
5. Select **Enable Cubes** to enable cubes generation, or deselect it otherwise.
6. Click **OK**.

11.7.3 What Happens When You Enable BPM Process Cubes in a Project

The BPMN Service Engine populates the pre-defined cubes each time it runs an activity or completes a process. The engine uses the sampling points configuration you defined to populate the cubes. If you configure a process not to generate sampling points, then the BPMN Service Engine does not store this information in the pre-defined cubes.

11.8 Enabling Oracle BAM in a Project

You can Oracle BAM to monitor the activity of the process in your project, leveraging the capabilities of Oracle BAM while using Oracle BPM.

You can use Oracle BAM with the pre-defined cubes, or you can choose to disable the latter.

For more information about Oracle BAM, see *Part X Using Oracle Business Activity Monitoring* in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

11.8.1 How to Enable Oracle BAM in a Project

Before enabling Oracle BAM in your project, you must configure Oracle BAM correctly. See chapter "Configuring Oracle BPMN Process Service Components and

Engines" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*, for information on how to configure Oracle BAM to work with Oracle BPM.

When you deploy a BPM project, Oracle BAM automatically creates the custom and predefined BAM data objects for that BPM project.

To enable Oracle BAM in a project:

1. In the BPM Project Navigator, right-click the project.
2. Select **Project Preferences**.
3. In the Category tree, select **Process Analytics Summary**.
4. In the Process Analytics Summary section, click the **Data Targets** tab.
5. Select **Enable BAM**.
6. From the JNDI Name BAM Adapter list, select the name for the BAM adapter.

   The BAM Adapter is labeled as `eis/bam/soap`. The JNDI name specifies the connection pool the BAM Adapter uses.
7. If you want the BAM adapter to process the requests in batch, then you must select **In Batch**.
8. In the Data Objects Path text-field, enter the path to the pre-installed BAM data objects.

   The default path is `/Samples/Monitor Express`.
9. Click **OK**.

11.8.2 What Happens When You Enable Oracle BAM

When you run a process that has Oracle BAM enabled the BPMN Service Engine populates Oracle BAM database with information about the business indicators measured in that process. The BPMN Service Engine generates this information based on the Sampling Points preference you defined in your project.

For more information, see *What You Need To Know About Monitor Express Data Objects* in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Part V

Working with Business Components

This part provides a general introduction to the business catalog. It also provides detailed information on each of the components that you can store in the business catalog.

This part contains the following chapters:

12

Using the Business Catalog

This chapter describes how to use the business catalog to store and organize the components needed to implement the processes in your BPM Project.

This chapter includes the following sections:

12.1 Introduction to the Business Catalog

The business catalog is a repository that stores the components you use to implement some flow objects in BPMN processes.

The business catalog stores the following types of components:

- Errors
- Events
- Human Tasks
- Business Rules
- Service Adapters
- Synthesized Types
- BPEL Processes and Mediators
- Business Objects
- Business Exceptions

Depending on the component you can use them for the implementation of a specific activity or multiple flow objects or to define the data associations of a flow object.

Table 12–1 shows which flow objects use each of the components in the business catalog for their implementation.

Table 12–1 Flow Object Implementation

Component	Flow Objects
Error	■ Start error event ■ Throw error event ■ Catch error event ■ End error event
Business exception	■ Start error event ■ Throw error event ■ Catch error event ■ End Error Event
Event	■ Start signal event ■ Throw signal event ■ End signal event
Human Task	User task
Business Rule	Business rule task
Service Adapter	■ Service task ■ Message throw event ■ Message catch event ■ Message end event ■ Send and receive tasks
Mediator	■ Service task ■ Message throw event ■ Message catch event ■ Message end event ■ Send and receive tasks
BPEL Process	■ Service task ■ Message throw event ■ Message catch event ■ Message end event ■ Send and receive tasks
Business Object	You can use them as arguments in the data associations of the following: ■ Service task ■ User task ■ Business rule task ■ Message throw event ■ Message catch event ■ Message end event ■ Send and receive tasks

Depending on the type of component, the business catalog uses two different ways of storing them. You can divide the components by the way the business catalog stores them into the following categories:

- Non-Synthesized Components
- Synthesized Components

12.1.1 Non-Synthesized Components

The business catalog stores a file with information about these components. When you open a BPM project, the business catalog reads the file it created to load the component.

The following components are not synthesized:

- Business objects
- Exceptions
- Modules
- Customized services and references

12.1.2 Synthesized Components

The business catalog generates the component structure dynamically based on an SOA component included in the SOA composite or an XML type or element. You cannot modify these components. Depending on the type of component they appear on a different predefined module. You cannot move the component to another module. To modify or store the component in another module, you must customize the service or the type.

The business catalog does not store any type of file for synthesized components. It generates the structure of synthesized components dynamically based on the XML or SOA component they represent.

You cannot modify synthesized components or move them to another module. Because these components are dynamically generated, they automatically reflect any change you make to the XML schema or SOA component they are based on.

The following components are synthesized:

- Synthesized Types
- Business Rules
- Human Tasks
- BPEL Processes
- Mediators

12.1.3 Adding Components to the Business Catalog

The way you add a component to the business catalog varies according to the type of component.

- **Errors**: When you add a service or a reference to your BPMN Project, if the operations they contain can generate errors, then these errors appear in the *Errors* predefined module.
- **Events**: When you add events to the SOA Composite, they appear in the *Events* predefined module. See Section 20.10, "Introduction to Communication Between Processes Using Signal Events", for more information on how to use events in a BPM project.

- **Human Tasks**: The existing Human Tasks included in the SOA Composite and the new ones you add automatically appear as components in the *HumanTask* predefined module. This component is generated from the Human Task you added. See Chapter 14, "Using Human Tasks" for more information.
- **Business Rules**: The existing Business Rules included in the SOA Composite and the new ones you add automatically appear as components in the *BusinessRules* predefined module. This component is generated from the Business Rule you added. See Chapter 16, "Using Business Rules" for more information.
- **Service Adapters**: The existing Service Adapters in the SOA Composite and the new ones you add, appear in the *Services* and *References* predefined modules. See Chapter 15, "Working with Services and References" for more information.
- **BPEL Processes and Mediators**: The BPEL Processes and Mediators included in the SOA Composite and the new ones you add, automatically appear as components in the *Services* and *References* predefined modules. See Section 15.3, "Introduction to Oracle Mediator in Oracle BPM" and Section 15.4, "Introduction to BPEL Processes in Oracle BPM" for more information.
- **Synthesized Types**: When you add a service or a reference that requires one or more arguments, if the data type of those arguments does not exist in the *Types* predefined module, then Studio automatically adds them. See Section 15.1, "Introduction to Services and References" for more information. You can customize a synthesized type to change its name and move it to a user-defined module. See Section 12.4, "Customizing Synthesized Types" for more information on how to customize a synthesized type.
- **Business Objects**: There are different ways of adding Business Objects to the business catalog. See Chapter 13, "Modeling Business Objects" for more information.

12.1.4 Using Modules to Organize Business Components

You can organize the Business Objects in the business catalog into different groups using modules. Generally you group all the related components into a module.

In the Sales Quote example you can create a module named *Quotes* to store all the components used to manage the information about the quotes used in the process.

You can nest modules. Nesting modules enables you to create a hierarchical structure that reflects the organization of your components.

In the Sales Quote example you might want to group all the modules that handle the information in the project into a single module. To do this you can create a module named *Data* that contains modules like *Quotes* and *Contracts*.

Organizing components using modules has the following benefits:

- It improves the readability of your project. Ideally the name of the module provides information about the components it contains.
- It makes it easier to locate a specific component.
- If needed, you can use the same name to identify different components that belong to different modules.

You cannot add Business Objects in the root level of the business catalog. You must always create a module where you can store your Business Objects.

It is a good practice to name the modules using a descriptive identifier. This makes it easier to find a component and makes your project easier to understand for other developers.

12.1.4.1 Predefined Modules

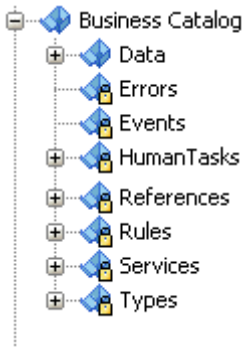
The business catalog contains the following predefined modules:

- **Errors**: Stores the errors that the operations in the services and references in your project define.
- **Events**: Stores the Events that you add to the SOA Composite.
- **HumanTasks**: Stores the Human Tasks that you add to your project.
- **References**: Stores the interfaces you can use to define the interface of your BPMN process.
- **Rules**: Stores the Business Rules that you add to your project.
- **Services**: Stores the components that you can use to implement the flow object in your BPMN process.
- **Types**: Stores the types that Studio generates when you add a service or a reference that require arguments of types that do not exist in the business catalog.

These modules are a permanent part of the business catalog and you cannot remove or rename them. Studio does not allow you to create new top level modules with these names.

You cannot create new modules within these predefined modules. Because the components stored in them are synthesized, you cannot rename them or move them to other modules.

Figure 12–1 Custom and Predefined Modules



This figure shows the business catalog node of the Sales Quote example. The example contains a custom module named Data to store the components used in the project. The figure also shows the predefined modules Errors, Events, Human Tasks, References, Rules, Services and Types. These predefined modules are marked with a lock icon.

****

12.2 Adding a New Module

The business catalog enables you to create new modules to store and organize the Business Objects in your project. You can add a module at the root level of the business catalog or within another module.

12.2.1 How to Add a New Module

You can create a new module to store the new components you create and the customized services and references.

To add a new module:

1. In the BPM Project Navigator, right-click the **Business Catalog** node or a user-defined existing module.
2. Select **New**.
3. Select **Module**.
4. Enter a name to identify the module.
5. Click **OK**.

12.2.2 What Happens When You Add a New Module

The new module appears in the business catalog. You can create new business objects, business exceptions and modules within the new module. You can also customize services and references

12.3 Deleting a Module

You can delete the custom modules in the business catalog. You cannot delete the predefined modules. When you delete a module, you also delete all the components within the module.

12.3.1 How to Delete a Module

You can delete those modules that you do not use or need.

To delete a module:

1. In the **BPM Project Navigator**, right-click the module you want to remove.
2. Select Delete.

   A confirmation message appears.
3. Click OK.

12.3.2 What Happens When You Delete a Module

When you delete a module, the module and the components within the module are removed from the business catalog.

If there are any flow objects or components in your project that use any of the deleted components, then this causes errors when you build your BPM project.

12.4 Customizing Synthesized Types

You can customize a synthesized type to change its name for a more descriptive name that helps process analysts and process developers understand the use of the component. When you customize a type you must also provide a user-defined module in which to store the customized type.

12.4.1 How to Customize a Synthesized Type

You can customize a synthesized type to change its name and move it to a user defined module.

To customize a synthesized type:

1. In the BPM Project Navigator, expand the **Types** predefined module.
2. Right-click the type you want to customize.
3. Select **Customize Type**.

   The Create Business Object dialog appears.
4. In the Name field, enter a name for the customized type.
5. In the Destination Module field, enter the name of the module in where you want to store the customized type, or click the browse button to browse the available modules and select one.
6. Click **OK**.

12.4.2 What Happens When You Customize a Synthesized Type

The synthesized type in the *Types* predefined module disappears and a new business object appears in the module you selected for the customized type.

Oracle BPM Studio automatically replaces all the references to the synthesized type with references to the customized type.

If you delete the customized type, then the synthesized type appears back in the *Types* predefined module.

13

Modeling Business Objects

This chapter describes how to use Business Objects in a BPM Project. Business Objects allow you to manage the data in your process efficiently and enable you to reuse existing components. They reduce the complexity of your process making it easier to maintain.

This chapter includes the following sections:

13.1 Introduction to Business Objects

Business Objects allow you to model and develop the business entities that are part of your process using the Object Oriented paradigm.

Using Business Objects simplifies the management of the data in your process by encapsulating the data and business behavior associated with the business entity it represents.

A business object is composed of a set of attributes and a set of methods. Attributes store the data related to the entity you are modeling. Methods manipulate the value of these attributes, or perform calculations based on their values.

Typically Business Objects represent entities in an actual business, but you can also use them to encapsulate business logic that is not associated to any particular entity.

Generally when your process contains a large number of data objects, you can group those that describe the same identity in a business object. For example, in the Sales Quote example you can group the following data in a Quote object:

- Quote Summary
- Quote Request Status
- Recommended Discount

Using Business Objects to manage a group of related data reduces significantly the complexity of your process by replacing multiple process data objects by a single data object of the type of the business object you defined. Additionally it provides you

other benefits described in the Section 13.1.2, "Benefits of Modeling Using Business Objects".

In a Sales Quote example you can identify the following business entities:

- Quote
- License Terms
- Product Item
- Approval Flow
- Contract

Each of these entities groups a set of highly related data. This data is represented in the attributes of an business object. The attributes define and describe the same business entity. The value of these attributes defines the state of the business object.

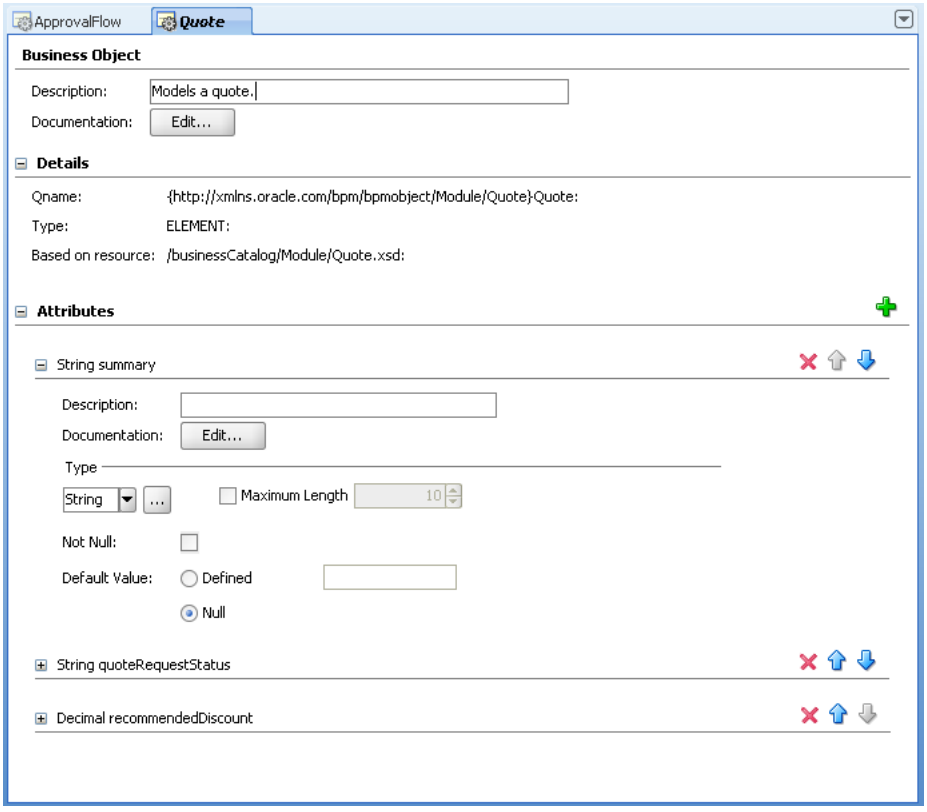
The Business Objects you define in your BPM project are stored in user-defined modules in the business catalog. When you open a business object, its editor shows you its description and the attributes that compose it.

Oracle BPM Studio provides an editor to view and edit the structure of a business object. The editor enables you to:

- Add a description
- Add documentation
- Add, edit, and remove attributes.
- View the namespace information

Figure 13–1 show the Business Object Editor editing a Quote object created manually.

Figure 13–1 Business Object Editor



This figure shows a Quote business object in the Business Object Editor. The Quote object contains two String attributes that specify the summary and the request status of the quote, and a decimal attribute to specify the recommended discount.

****

13.1.1 Types of Business Objects

The way you create a business object determines its characteristics and functionality.

The following are different ways of creating a business object:

- **Manually**: You can build a business object manually by creating it and then adding attributes and documentation.
- **Based on an XML schema element or complex type**: The resulting business object contains one or more attributes that map the selected schema element or complex type. You cannot remove these attributes, but you can add new attributes.
- **By customizing a synthetic type in the Types**: You can customize the types that the business catalog adds to the Types predefined module when you add a Service or a Reference that require them as arguments. When you customize a type you can store it in a user-defined module, change its name and add attributes to it.

13.1.2 Benefits of Modeling Using Business Objects

Using business objects to manage the data in your process provides you the following benefits:

- **Simpler Processes:** Using business object reduces the quantity of process data objects in your process. This makes your process simpler and easier to read.
- **Coupling Reduction:** If your process has fewer data objects, the subprocesses and activities that compose it, require less parameters.
- **Re-use**: You can use a business object you defined for a particular process in other processes that do not necessarily belong to the same project. Reusing business objects can dramatically reduce the development time of your project.
- **Easy Maintenance:** If you update or fix a bug in a business object all the processes using it benefit from those changes.
- **Parallel Development:** After you agree on a certain interface for the business objects in your process, some members of your team can work on the development of those business objects while others work on the development of the process.
- **Unit Testing:** You can test each of the business objects in your process separately. Unit Testing reduces the complexity of your test cases and improves significantly the quality of your project.

13.1.3 Naming Conventions for Business Objects

When you name a business object you should respect the following rules:

- Use one or more nouns, or nouns modified by adjectives.
- Do not start the name with a number.
- Use capital letters only to distinguish internal words.
- Keep names simple and descriptive.
- Use whole words, avoid using acronyms unless they are widely known.

Note: Studio forces the first letter of the name of a business object to uppercase.

13.2 Working with Business Objects

You can add business objects to your BPM project to store data related to the processes it contains. The business objects you add are stored in the business catalog, for more information about the business catalog, see Chapter 12, "Using the Business Catalog".

When developing a business object you can modify it, rename it, or delete them. You can also add documentation that helps you identify the functionality of the business object or describes how to use it.

13.2.1 How to Add a Business Object

You can add business objects to the business catalog to model the business entities to store the data in your BPMN process.

To add a business object:

1. Right-click a user-defined module in the business catalog.
2. Select **New** and then select **Business Object**.
3. Enter a name to identify the new business object.

Note: You cannot repeat a name within the same module. However you can assign the same name to business objects in different modules.

4. Click **OK**.

13.2.2 What Happens When You Add a Business Object

The business object appears in the business catalog. You can use this business object to define the type of the following elements in your BPMN process:

- Arguments in data associations
- Process data objects
- Project data objects

13.2.3 How to Modify a Business Object

You can modify an existing business object by:

- Adding attributes

  See Section 13.6.1, "How to Add a Business Object Attribute".
- Deleting attributes

  See Section 13.6.2, "How to Delete a Business Object Attribute".
- Adding documentation

  See Section 13.6.3, "How to Document a Business Object Attribute".

13.2.4 How to Delete a Business Object

You can delete a business object that you do not use or need. If your project contains flow objects or data associations that use the deleted business object, then you must remove them manually.

To delete a business object:

1. In the BPM Project Navigator, right-click the business object you want to delete.
2. Select **Delete**.

   A confirmation message appears.
3. Click **OK**.

13.2.5 What Happens When You Delete a Business Object

Oracle BPM Studio remove the business object from the business catalog. If there are any flow objects in your process that use the removed business object, then you must remove these references manually.

13.2.6 How to Document a Business Object

You can add documentation to a business object for other process developers to understand its functionality and data structure.

To Document a business object:

1. Edit the business object.
2. In the business object editor, in the business object Editor, click the Edit button next to the Documentation field.
3. Add the documentation for the business object.

   See Section 5.4.1, "Introduction to the Documentation Editor", for details on how to create and edit documentation.
4. Click **Close**.

13.2.7 What Happens When You Document a Business Object

The documentation is available for other process developers to read and modify.

13.3 Using a Business Object in a Process

You can use business objects to store data related to your process. To use a business object in your project, add a process data object to your process and set its type to the business object you created. You can update the information in this data object from any of the activities in the process.

13.3.1 How to Use a Business Object in a Process

You can create a complex data object in your process that defines its type using a business object.

To use a business object in a Process:

1. Add a process data object to your process. Use the business object as the type of the data object.

   See Section 8.3.1, "How to Add a Process Data Object", for information on how to add a process data object.

   Note: When selecting the type of the data object use the Browse More Types... button to display the complete list of types. Then select <Component> to display the list of available business objects.
2. Initialize the value of the data object in the process using a Script Task or Data Associations.

13.3.2 What Happens When You Use a Business Object in a Process

The data object you defined has the structure defined in the business object. The type of the data object is the name of the business object. For example, if you define a business object *SalesQuote* and then create a data object that uses this business object as its type, then the type of the data object is *SalesQuote*.

You can assign values to the data objects that use these types using data associations and script tasks.

13.4 Adding Business Objects Based on a XML Schema Element or Type

You can create a business object based on an XML schema element or complex type. The XML schema element or complex type you use to create your business object has to be part of your BPM Project. You can add an XML schema that contains the element or complex type to your project, or you can use a use a type defined inline in a WSDL file. For the latter you must add the WSDL file to your project by adding an SOA Adapter of type Web Service.

When you create a business object using an XML schema element, the selected element becomes an attribute of the resulting business object.

When you create a business object using an XML schema element, the selected element becomes an attribute of the resulting business object.

If you create a business object based on a schema contained in a WSDL file, then you cannot use the resulting business object as the type of an attribute of another business object.

13.4.1 How to Add a Business Object Based on a XML Schema Element or Type

Before following this procedure ensure that the business catalog contains the XML schema you want to use as a base for your Business Objects.

To add a business object based on an XML schema or complex type:

1. Right-click a user-defined module.
2. Select **New** and then select **Business Object**.

   The Create Business Object Dialog appears.
3. Enter a name to identify the new business object.
4. Select **Based on External Type**.
5. Click the **Browse** button next to the **External Type** field or add a new XML schema following the procedure described in Section 13.4.3, "How to add an XML Schema to Your BPM Project".
6. Select the external type from which to create the new business object.

13.4.2 What Happens When You Create a Business Object Based on an XML Schema Element or Type

You cannot modify or add attributes to the business object. The structure of the business object is based on the structure of the XML schema element or type.

13.4.3 How to add an XML Schema to Your BPM Project

From the Create Business Object dialog you can add an XML schema to your project.

To add an XML schema to your BPM project:

1. In the Create Business Object Dialog, click the **Schema Browser** button.

   The Type Chooser dialog appears.
2. In the upper right corner, click the **Import Schema Files** button.

   The Import Schema File dialog appears.
3. Click the **Browse Resources** button next to the URL field.

The SOA Resource Browser appears.

4. Browse your file system and select a schema file.
5. Select **Copy to Project**.
6. Click **OK**.

   If the XML schema contains references to other types a dialog to confirm their import appears.

   The Browse Resources dialog closes and the Type Chooser dialog appears.
7. Select an element to use as the base of your business object.

13.4.4 What Happens When You Add a Schema File to Your Project

The Schema Browser copies the selected XML schema to the *xsl* directory in your project. You can use it to create new business objects without having to re-add it.

13.5 Introduction to Business Object Attributes

Attributes store data that defines and describes the business object. The attributes in a business object are equivalent to instance variables in Object Orientation.

In the Sales Quote example you can identify the following attributes in the Quote object:

- Summary
- Product Items
- Quote Request Status
- License Terms
- Recommended Discount

These attributes describe the product and are relevant to the process. The ID or SKU serves to identify the chosen product. The description is probably used to show the user what the product does. And the price is used to show the customer how much the product costs and later in the process to calculate the total amount due.

When you define an attribute you must specify:

- Name: used to identify the attribute.
- Type: that defines the type of data you can store in the attribute. Attributes support simple types or other defined business objects.

Additionally you can define the following:

- Description: provides details about the attribute that help other process developers to understand its use.
- Documentation. See Section 13.6.3, "How to Document a Business Object Attribute".
- Custom default value
- Not null constraint.

13.5.1 Supported Data Types for Business Object Attributes

The following table describes the supported data types for an attribute in a business object:

Data Type	Description
Bool	True or false values
Int	Integer numbers
Decimal	Decimal numbers with defined precision
Real	Real numbers
String	Alphanumeric values
Time	Units of time
Interval	Intervals of time
Binary	Binary values (For example: images, files)
Array	A collection of elements of a specified data type
Complex Types	Other business objects

13.5.2 Naming Conventions for Business Object Attributes

When you name a attribute of a business object you should respect the following rules:

- Use one or more nouns, or nouns modified by adjectives
- Use capital letters only to distinguish internal words
- Keep names simple and descriptive
- Use whole words, avoid using acronyms unless they are widely known
- Do not start the name with symbols
- Use short but meaningful names
- Avoid using one-character names

> **Note:** Studio forces the first letter of the name of an attribute to lowercase.

13.6 Working with Business Object Attributes

To model a business object you must add it attributes. These attributes store the data related to your process. You can add, modify and delete attributes as necessary.

You can also add them documentation that describes the data they store and provides any necessary information to the user of the business object.

13.6.1 How to Add a Business Object Attribute

To model a business object that you created from the start, you must add attributes.

To add an attribute to an existing business object:

1. In the BPM Project Navigator Right-click the business object where you want to add the attribute.

2. Select **New** and then select **Attribute**.

> **Note:** Another alternative for the previous steps is editing the business object and clicking the Add button in the Attributes section.

3. Enter a name to identify the new attribute.
4. From the type list, select a type for the new attribute, or click the **Browse More Types...** button to select a complex type.
5. Click **OK**.

13.6.2 How to Delete a Business Object Attribute

To delete an attribute from an existing BPM Object:

1. Edit the Business Object that contains the attribute you want to remove.
2. In the Attributes section, click the **Remove Attribute** button next to the attribute you want to remove.

   A confirmation message appears.
3. Click **OK**.

13.6.3 How to Document a Business Object Attribute

You can add documentation to a business object attribute for other process developers to understand its functionality.

To document a business object Attribute:

1. Edit the business object that contains the attribute you want document.
2. In the Attributes section, expand the business object attribute you want to document.
3. Click the **Edit...** button next to the **Documentation** field.

   The Documentation Dialog appears.
4. Add the text to document the functionality of the selected attribute.

   See Section 5.4.1, "Introduction to the Documentation Editor", for details on how to create and edit documentation.

13.6.4 What Happens When You Document a Business Object Attribute

The documentation is available for other process developers to read and modify

14

Using Human Tasks

This chapter describes how to implement BPMN user tasks using Human Tasks. You can use an existing Human Task component created using the SOA Human Task editor, or you can create a new Human Task using the simplified interface Oracle BPM Studio provides.

This chapter includes the following sections:

For detailed information about Human Tasks, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

14.1 Introduction to Human Tasks in BPM

The implementation of user tasks requires you to define a Human Task. You can use an existing Human Task or define a new one.

If your project contains Human Tasks, then they automatically appear in the business catalog under the *HumanTasks* predefined module.

You can add new Human tasks to your project in the following ways:

- Using the simplified interface Oracle BPM Studio provides
- From the SOA New Gallery
- From the SOA Composite Editor

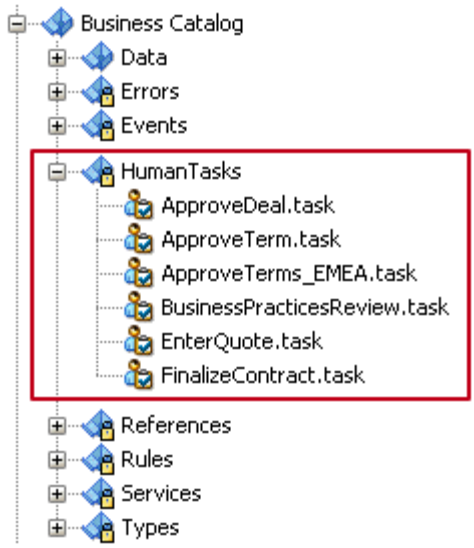
When you double click a Human Task component in the business catalog, Oracle BPM Studio opens the SOA Human Task editor. You can edit the Human Task using this editor.

Figure 14–1 shows a Human Task component in the Sales Quote example.

For more information on how to define Human Tasks using Oracle SOA Suite, see the following chapters in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*:

- *Designing Human Tasks*
- *Designing Task Forms for Human Tasks*

Figure 14–1 Human Task components in the Business Catalog



This figure shows the Human Tasks components in the Sales Quote example. The business catalog stores all the Human Tasks components in the BPM project within the predefined module *HumanTasks*.

****

At run time, when a token arrives at a user task control is passed from the BPMN process to the Oracle Human Workflow. Although both are part of Oracle BPM run time, control is not passed back to the BPMN process until the Human Tasks is completed.

After the workflow is complete, control is passed back the BPMN process, any required data objects are passed back to the user task, and the token moves to the next sequence flow of the process.

However human tasks are independent from BPMN processes. If you terminate a BPMN process while it runs a user task, the associated human tasks keeps running independently. For more information see Section 23.1.1, "Understanding the Relationship Between SOA Composites and SOA Components".

If the process instance leaves the user task before the human tasks is completed, the human task continues running and can you can still access it. This is because human tasks are independent from the BPMN process. Any changes you make to a human task after the process instance left the corresponding user task, do not appear in the audit trail.

Note: When you define a human task in BPM the callback is implicitly defined.

14.1.1 Typical Design Workflow

There are two approaches to working with Human Tasks in Oracle BPM:

- Creating the Human Task using the SOA Human Task editor
- Creating the Human Task using the simplified interface Oracle BPM provides

The approach you choose depends on how you plan your work, how you divide it between the developers in your team and the complexity of the Human Tasks you are developing.

> **Note:** When you create a Human Task using Oracle BPM Suite, the *enableAutoClaim* property is set to true by default.

Creating the Human Task Using the SOA Human Task Editor:

- Create a Human Task using the SOA Human Task
- Create the corresponding taskflow using SOA Suite
- Create a BPMN process with user tasks
- Implement the user tasks in the BPMN process using the defined Human Tasks

Creating the Human Task Using Oracle BPM Human Task Editor

- Create a BPMN process
- Add a user task. From the user task implementation properties dialog, create a Human Task.
- Create the corresponding taskflows using SOA Suite

14.2 Assigning an Existing Human Task to a User Task

You can create a Human Task using the SOA Human Task editor and then assign that Human Task to the implementation of a user task.

You must also define how the data objects in your BPM process map to the input and output arguments of the Human Task. You can do this using data associations or transformation. For more information on data associations and transformations, see Chapter 8, "Handling Information in Your Process Design".

14.2.1 How to Assign an Existing Human Task to a User Task

You can implement a user task using an existing Human Task that you created for another user task or using the SOA Human Task editor.

To assign an existing Human Task to a user task:

1. Open the BPMN process.
2. Right-click the user task.
3. Select **Properties**.

   The Properties - User Task dialog appears.
4. Click the **Implementation** tab.
5. Click the **Browse** button next to the Human Task field.

   The Browse Human Tasks dialog appears.
6. Select a Human Task from the list.
7. Click **OK**.

   The Browse Human Tasks dialog closes and the selected Human Task appears in the Human Task field.

8. Click **OK**.

14.2.2 What Happens When You Assign an Existing Human Task to a User Task

The user task uses the existing Human Task for its implementation.

The SOA Composite displays the relationship between the BPMN process and the Human task by adding a wire between them.

When the BPMN Service Engine runs the user task implementation it invokes the Human Workflow Service with the parameters defined in the data association of the user task. When the Human Workflow Service finishes running the Human Tasks it provides the result to the BPMN Service Engine using the defined data association.

14.2.3 How to Associate the Process Payload to the Human Task Payload

To associate the process payload to the Human Task payload you must configure the Human Task, the user task and start events in the BPMN process and create a business object based on the payload xsd file.

To associate the process payload to the human task payload:

1. Create a business object using the Based on External Schema option.
   a. Right-click a module.
   b. Select **New** and then select **Business Object**.
   c. Select **Based on External Schema**.
   d. Click the **Browse** button.
   e. Select **Copy to Project**.
   f. Click **Browse Resources**.

      The Type Chooser dialog opens.
   g. Select the type of the business object from the payload xsd file.
2. Edit the start event in your BPMN process.
3. Define a custom argument using the business object you created as its type.
4. Add a process data object to your process using the business object you created as its type.
5. Define the data associations between the custom argument and data object.
6. In the Human Task editor, click the **Data** tab.
7. Select Add other payload from the list in the Add button.
8. Select the payload element in the Type Chooser dialog.
9. In the Process editor, right-click the user task and select **Properties**.
10. Define the data associations between the process data object and the task payload.

14.3 Creating a Human Task from Oracle BPM Studio

You can create a simple Human Task using Oracle BPM Studio. The simplified interface Oracle BPM Studio provides hides the complexity of the Human Task editor by exposing only those fields that are relevant to Oracle BPM. After you create the

Human Task using the simplified editor, you can edit it using the SOA Human Task editor if needed.

The simplified Human Task editor Oracle BPM Studio provides, enables you to define the following properties:

- **Title**

  Defines the name of the Human Task that is displayed to end-users in the Oracle BPM WorkSpace and WorkList applications.

- **Priority**

  Specifies a priority for the Human Task. Valid values are between 1 (highest priority) and 5 (lowest priority). The default value is 3.

- **Re-Initiate**

  Restarts the approval process from the beginning

- **Outcomes**

  Specifies the outcome 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.

- **Parameters**

  Define the Human Task payload. The Human Task data association is based on the parameters of the Human Task. The data association maps the data objects as input arguments.

- **Outcome Target**

  Specifies a String data object to store the outcome argument of the Human Task. You can only select one data object.

The Advanced User Task Properties enable you to define the following properties:

- **Initiator**

  Specifies the user who initiates a task. The initiator can view their created tasks from Oracle BPM Worklist and perform specific tasks, such as withdrawing or suspending a task.

- **Owner**

  Specifies the User ID of the task owner

- **Identification Key**

  Defines a user-defined ID for the task. For example, if the task is meant for approving a purchase order, the purchase order ID can be set as the identification key of the task. Tasks can be searched from Oracle BPM Worklist using the identification key. This attribute has no default value.

- **Identity Context**

  This field is required if you are using multiple realms. You cannot have assignees from multiple realms working on the same task.

- **Application Context**

  Specifies the name of the application that contains the application roles used in the task. This indicates the context in which the application role operates.

14.3.1 How to Create a Human Task from Oracle BPM Studio

You can create a Human Task from the User Task Properties dialog in Oracle BPM Studio.

To create a human task from Oracle BPM Studio:

1. Edit the BPMN process.
2. Right-click the user task.
3. Select **Properties**.

   The Properties - User Task dialog appears.
4. Click the **Implementation** tab.
5. Click the **Add** button next to the Human Task field.

   The Create Human Task dialog appears.
6. In the name field, enter a name to identify the Human Task.
7. From they Priority List, select a priority.
8. If you want to use an approval management pattern, then you must select one from the Pattern list.
9. In the Title Field, enter a title for the WorkList to display.
10. Optionally, you can configure the following:
    - The outcome

      See Section 14.3.2, "How to Configure the Outcome of a Human Task" for information on how to configure the outcome of a Human Task.
    - The parameters

      See Section 14.3.3, "How to Add a Parameter to Human Task" for information on how to configure the outcome of a Human Task.
    - The outcome target

      See Section 14.3.4, "How to Configure the Outcome Target of a Human Task" for information on how to configure the outcome of a Human Task.
11. Click **OK**.

    The Create Human Task dialog closes and the Human Task field in the User Task Properties dialog shows the Human Task you created.
12. Click **OK**.

    The User Task Properties closes and saves the implementation you configured for the user task.

14.3.2 How to Configure the Outcome of a Human Task

When you create a Human Task from Oracle BPM Studio you can configure the outcome of the Human Task. The outcome values you configure appear as the available actions of the Human Task in Oracle Worklist.

To configure the outcome of a Human Task:

1. In the Create Human Task dialog, click the **Browse** button next to the Outcomes field.

The Outcomes dialog appears.

2. Select one or more outcomes, or click the **Add** button to add a new custom outcome.
3. Optionally click **Outcomes Requiring Comment**, to select those outcomes that require comments.
4. Click **OK**.

   The Outcomes dialog closes and the selected outcomes appear in the Create Human Task dialog, in the Outcomes field.

14.3.3 How to Add a Parameter to Human Task

You can add multiple parameters to a Human Task to build the Human Task payload. Oracle BPM Studio uses this parameters to create the data association of the user task that uses the Human Task.

To add a parameter to a Human Task:

1. In the Create Human Task dialog, click the **Add** button in the Parameters table.

   The Data Objects dialog appears.
2. Select a data object from the Data Objects dialog and drop it on the Parameters table.

   The selected data object appears in the Parameters table.
3. Close the **Data Objects** dialog.
4. Optionally you can mark the parameter as editable by selecting the Editable column in the Parameters table.

14.3.4 How to Configure the Outcome Target of a Human Task

When you create a Human Task you must define an outcome target. The outcome target maps the result of the Human Task to a String data object in your BPM project.

To configure the outcome target of a Human Task

1. In the Create Human Task dialog, click the **Add** button next to the Outcome Target field.

   The Data Objects dialog appears.
2. Select a String data object from the Data Objects dialog and drop it on the Outcome Target field.

   The selected data object appears in the Outcome Target field.
3. Close the **Data Objects** dialog.

14.3.5 What Happens When You Create a Human Task from Oracle BPM Studio

The Human Task automatically appears in the *HumanTasks* predefined module in the business catalog. You can use the Human Task to implement the user task you are editing or other user tasks in the BPM project.

You can edit the created Human Task using the SOA Human Task editor to configure implementation details.

14.4 Creating a Human Task Using the SOA Human Task Editor

You can create and edit complex Human Tasks using the SOA Human Task editor. For more information on how to create Human Tasks using Oracle SOA Suite, see chapter "Designing Human Tasks" in Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite.

14.4.1 How to Specify an e-mail Address for the Recipient of a Notification

When using the SOA Human Task editor in a BPM Suite installation, you can specify an e-mail address for the recipient of a Notification.

To specify an e-mail address for the recipient of a notification:

1. Open the **Human Task** editor.
2. Click the **Notification** tab.
3. Double-click the **Recipient** list.

   The Recipient list is an editable list, when you double click it, it becomes a text field.
4. Enter the recipient's e-mail address.

   Optionally you can use the buttons next to the Recipient text field to look up the e-mail address in an application server or to specify the e-mail address using XPath.

Note: When sending a notification to a recipient specified using an e-mail address, the notification service uses the user context of an assignee to obtain the task information to include in the notification.

14.5 Editing a Human Task from Oracle BPM Studio

You can edit a human task using Oracle BPM Simplified editor or the SOA Human Task editor. Generally you use the SOA Human Task editor for complex human tasks.

14.5.1 How to Edit a Human Task Using the Oracle BPM Simplified Editor

To edit a Human Task using the Oracle BPM Simplified Editor:

1. Open the BPMN process that contains the user task implemented with a Human Task.
2. Right-click the user task.
3. Select **Properties**.

   The user task properties dialog box appears.
4. Click the **Implementation** Tab.
5. Make changes to the properties in the **Human Task Attributes** and **Advanced** sections.

14.5.2 How to Edit a Human Task Using the SOA Human Task Editor

To edit a Human Task using the SOA Human Task Editor:

1. Open the BPMN process that contains the user task implemented with a Human Task.
2. Right-click the user task.
3. Select **Open Human Task**.

   The SOA Human Task editor appears.
4. Make changes to the Human Task.

14.6 Using Human Task Patterns in Oracle BPM

Human task patterns allow you to use a predefined flow to create the Human Task. These predefined patterns contain standard process flows that are common to all business processes.

Oracle BPM supports the following Human Tasks patterns:

- Complex
- FYI
- Group
- Initiator
- Management
- User

You can add a Human Tasks that uses patterns by selecting the specific user task in the Interactive Activities section in the Component Palette, or you can add a generic user task and when you create the Human Task select the pattern you want to use.

For more information about Human Task patterns, see Chapter 26, "Using Approval Management".

15

Working with Services and References

This chapter describes the different service and reference components that you can use in Oracle BPM. It describes how these components appear in the business catalog and how the components in the business catalog relate to the SOA composite that defines these services and references. It also describes how to customize these components to make them easier to understand and more appropriate for business analysts.

This chapter includes the following sections:

15.1 Introduction to Services and References

Some flow objects in Oracle BPM require you to define a service or a reference to implement them. You can define services and references in the SOA composite in your BPM project.

The business catalog displays the services, components, and references that appear in the SOA composite. When you add a new component in the Exposed Services or External References areas in the composite, it automatically appears in the corresponding predefined module in the business catalog.

The following SOA components appear as services or references in the business catalog:

- SOA service adapters
- SOA mediators
- BPEL processes

> **Note:** When you define a web service to implement a service task, message events, or send and receive tasks, ensure that the operations it contains do not define arguments of XML types defined within a WSDL. The arguments in the operations in the web service must be primitive types or types defined within an XSD file.

15.1.1 Introduction to Services

Services are those components that you can use to implement certain activities and events in your BPMN process.

The Services predefined module stores the components that display a service handle in the SOA Composite.

You can use services to implement the following flow objects:

- Service tasks
- Message events
- Send and receive tasks

15.1.2 Introduction to References

References are the interfaces that you can use to define the interface of your BPMN processes.

The References predefined module stores the components that display a reference handle in the SOA composite.

You can use references to define the process interface using the following flow objects:

- Message events
- Send and receive tasks

15.1.3 Introduction to Callbacks

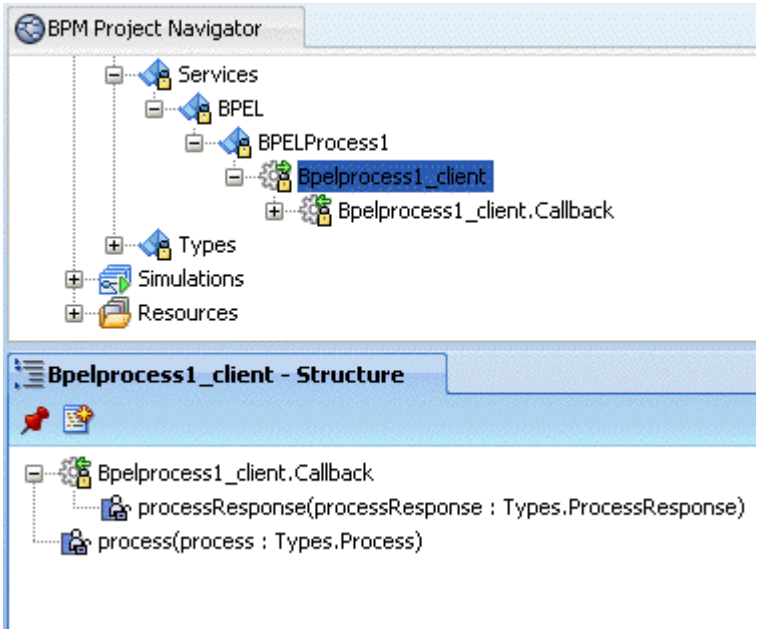
If a service is asynchronous and contains a callback interface, then the component in the business catalog contains a callback inner component. The callback inner component groups all the callback operations in the service.

After selecting the service component in the business catalog, you can view a list of the operations in the callback component in the Structure window.

The implementation of message events and receive tasks configured to wait for a callback from the service only enable you to select an operation from the callback inner component of the corresponding service.

Figure 15–1 shows a service with a callback interface in the business catalog.

Figure 15–1 Service with Callback Interface



This figure shows the service component that represents an asynchronous BPEL process with a callback interface. The BPM Project Navigator displays the callback inner component under the service component. The Structure window displays the operations in the service component and the inner callback component. The inner callback component is expanded to view the list of operations it contains.

****

15.2 Introduction to Service Adapters in Oracle BPM

Service adapters enable you to integrate with other applications and external services. Oracle BPM supports the use of service adapters to integrate your BPMN process with external applications, legacy applications, and external services such as FTP or databases.

Oracle BPM supports service adapters for the following technologies:

- ADF-BC Service
- Advanced Queuing
- B2B
- Oracle BAM Adapter
- A variety of databases
- Direct binding
- EJB service
- Files
- FTP
- HTTP
- JMS
- MQSeries

- Oracle applications
- Sockets
- Third-party adapters
- Web services

For a detailed description of service adapters see:

- "Getting Started with Binding Components" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*
- *Oracle Fusion Middleware User's Guide for Technology Adapters*

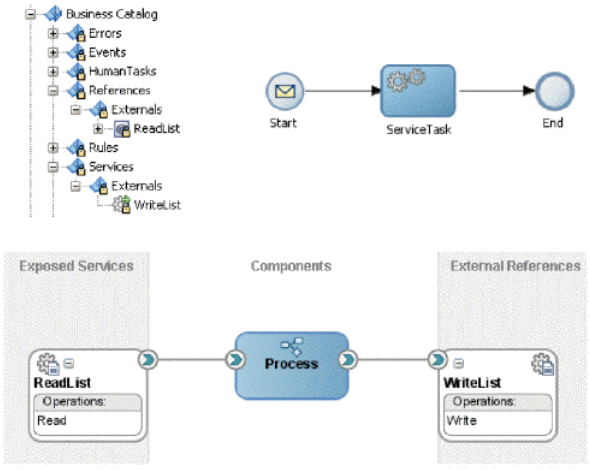
When you add a SOA service adapter to the SOA composite of a BPM project, the service adapter automatically appears in the business catalog. The business catalog stores the SOA service adapters in different modules depending on the swimlane of the SOA composite where you added the SOA service adapter:

- If you add the service adapter in the External References swimlane, then the business catalog displays this adapter in the Services predefined module.
- If you add the Service Adapter in the Exposed Services swimlane, then the business catalog displays this adapter in the References predefined module.

Depending on the nature of the service adapter, the SOA composite enables you to add the components in the different swimlanes. For example, you must add a file adapter that contains a read operation in the Exposed Services swimlane, but if the file adapter contains a write operation, then you must add it in the External References swimlane.

Figure 15–2 shows a SOA composite that contains a file adapter with read operation and a file adapter with a write operation. Note that the file adapter that contains the write operation appears under the Services predefined module in the business catalog, while the file adapter that contains the read operation appears under the References predefined module.

Figure 15–2 Adapter Services in Oracle BPM



This Figure shows an SOA Composite that contains a file adapter with a read operation and another file adapter with a write operation. The file adapter with the read operation appears in the business catalog under the References predefined module, while the file adapter with the write operation appears under the Services predefined Module. The BPMN process contains a message start event that defines the interface using the ReadList interface and invokes the Write operation in the WriteList

file adapter using a service task. In the SOA Composite, the wires between the BPMN process and the file adapters indicate the described relationship.

****

The business catalog stores service adapters under the External module within the Services or References predefined modules. The service adapter is represented as a node. If you select a service adapter in the BPM Project Navigator, then the Structure window displays the operations it contains. If the service adapter is configured as asynchronous, then the Structure window also displays the callback inner object.

Note: The operations in the service adapters that do not return output arguments may define an element structure Empty as the output argument. This element structure appears in the data association of the flow object that uses the service adapter. Defining output data associations is optional, so are not required to provide a mapping for this element structure.

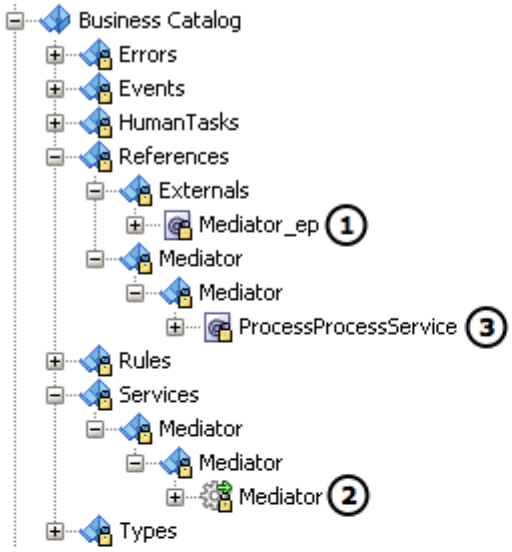
15.3 Introduction to Oracle Mediator in Oracle BPM

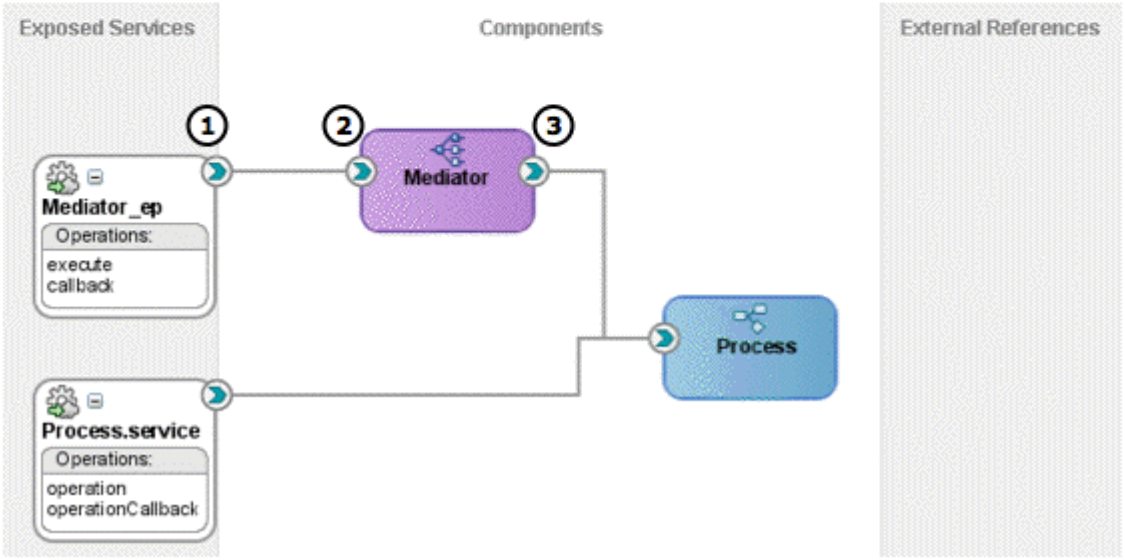
Oracle Mediator facilitates the communication among the components within a composite application. These components include BPMN processes.

You can use Mediator in a BPM project in the following use cases:

- BPMN processes can invoke other components in the SOA composite through a Mediator component. The BPMN process invokes the mediator with certain input data. The mediator transforms this data to adjust to the requirements of the other component and invokes the component.
- The components in the SOAcComposite can invoke a BPMN process through a Mediator component. If the mediator exposes an interface, then external components can also invoke a BPMN process through the mediator. The component or external component invokes the mediator with certain input data. The mediator transforms this data to adjust to the requirements of the BPMN process and in turn invokes the component.

Figure 15–3 Mediator Components in the Business Catalog



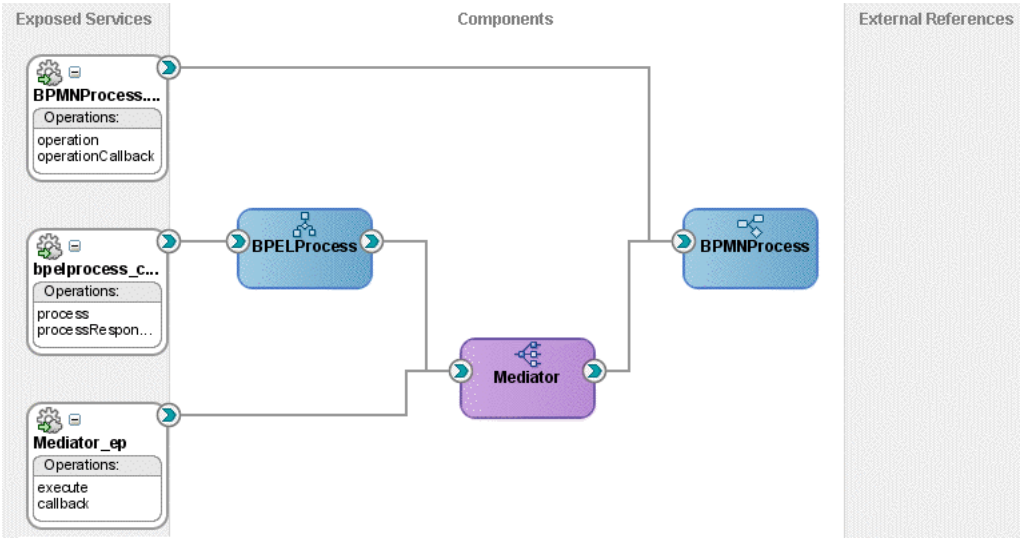


This figure shows the components the business catalog generates when you add a Mediator component to the SOA Composite.

****

Figure 15–4 shows a BPEL process that invokes a BPMN process through a mediator. Note that the service handle of the mediator connects to the BPEL process and the reference handle connects to the BPMN process.

Figure 15–4 BPEL Process Using a Mediator to Invoke a BPMN Process



This figure shows a BPEL process that invokes a BPMN process through a Mediator.

****

The Mediator Service

When you add a mediator to the SOA composite the business catalog generates a mediator service.

This component represents the mediator service. It contains the operations you invoke to communicate with the mediator. You can invoke the operations defined in this service using service tasks, message events, or send and receive tasks, depending on the type of operation.

The business catalog stores mediator services in the Mediator module located in the Services predefined module. It creates a separate module for each of the mediator service components. The name of this module is the name of the component.

Item 2 in Figure 15–3 shows the exposed service Mediator_ep for the mediator component shown in the SOA composite

The Mediator Interface

If you select the SOA binding option when creating the mediator, then Oracle JDeveloper creates the service interface. This interface defines the signature of the operations you can use to access the mediator from outside the SOA composite. You can configure your BPMN process to use this interface, so that the BPMN and the mediator have the same interface.

The business catalog stores service interfaces in the Externals module located in the References predefined module.

Item 1 in Figure 15–3 shows the exposed service interface component Mediator_ep for the mediator component shown in the SOA composite.

For information on how to use an interface to define the process interface, see Section 21.5, "Using Message Events with an Interface from the Business Catalog to Define Your Process Interface" and Section 21.9, "Using Send and Receive Tasks with an Interface from the Business Catalog to Define Your Process Interface".

The Reference Interfaces

The SOA composite shows an interface for each component the mediator adapts.

The business catalog stores service interfaces in the Mediator module located in the References predefined module. It creates a separate module for each of the Mediator service components. The name of this module is the name of the component

Item 1 in Figure 15–3 shows the reference interface component ProcessProcessService for the mediator component shown in the SOA composite.

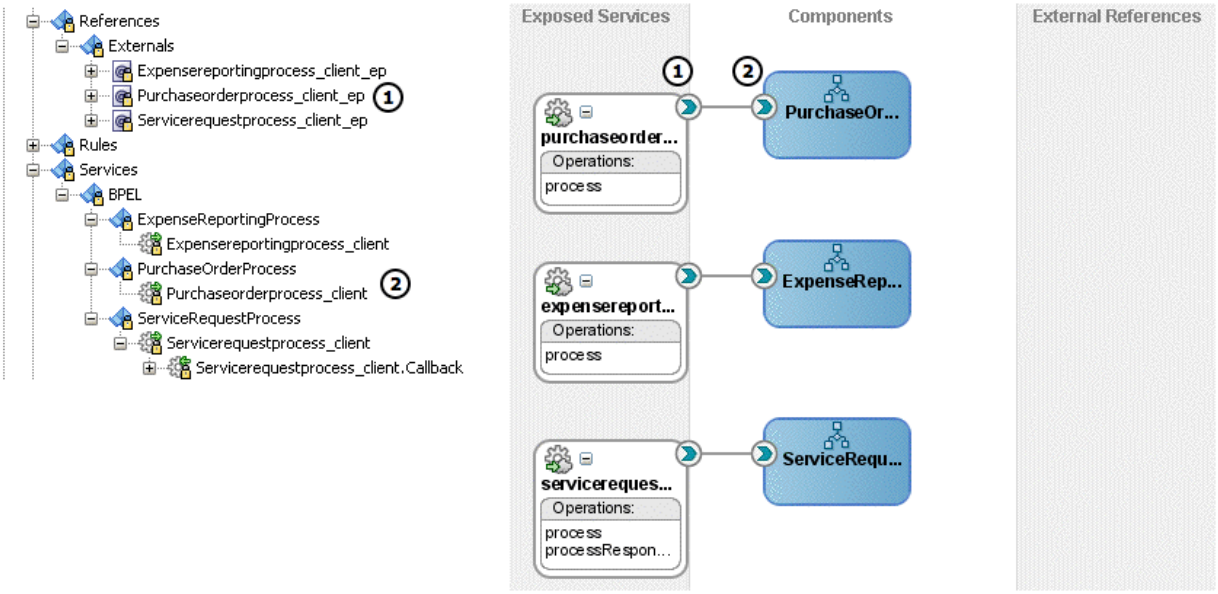
For more information on SOA mediators, see part "Using the Oracle Mediator Service Component" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

15.4 Introduction to BPEL Processes in Oracle BPM

BPEL processes enable you to model a business process using a standard different from BPMN. Depending on the nature of the process, some processes might be easier to implement in a certain technology. Oracle BPM enables you to integrate the BPEL and BPMN processes in your project, getting the best of the two standards.

Figure 15–5 shows an SOA composite that contains multiple BPEL processes and how the business catalog displays them.

Figure 15–5 BPEL Process Components in Business Catalog



This figure shows multiple BPEL processes in the SOA Composite and how the business catalog displays this components in Oracle BPM.

****

The BPEL Service Component

When you add a BPEL process to the SOA composite, the BPEL service component appears in the business catalog.

This component represents the BPEL process service. You can use this component to implement service tasks or message events, or send and receive tasks, depending if the BPEL process is synchronous or asynchronous.

Figure 15–5 shows how the business catalog displays a BPEL process and its corresponding exposed service interface.

The business catalog shows the BPEL process service in the BPEL module located in the Services predefined module. It creates a separate module for each of the BPEL process service components. The name of this module is the name of the BPEL process.

Oracle BPM treats BPEL processes as services. It does not make a distinction between other types of services and BPEL processes.

For more information on how to invoke synchronous and asynchronous services from a BPMN process, see Section 20.1, "Introduction to Communication with Other BPMN Processes and Services".

The Exposed Interface

If you selected the SOA binding option when creating the BPEL process, then the exposed interface appears in the business catalog.

This interface enables external components to invoke BPEL processes. If you are designing a BPMN process to replace a BPEL process, then you might want to use this interface to define a BPMN process to ensure that you can replace one for the other.

For information on how to use an interface to define the process interface, see Section 21.5, "Using Message Events with an Interface from the Business Catalog to Define Your Process Interface" and Section 21.9, "Using Send and Receive Tasks with an Interface from the Business Catalog to Define Your Process Interface".

For more information on BPEL Processes, see Using the BPEL Process Service Component in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

15.5 Using Services in Oracle BPM

The following flow objects require you to define a service to implement them:

- Service tasks

  Service tasks enable you to invoke synchronous services. To implement a service task you must specify a synchronous service in the implementation properties. See Section 20.5, "Using Service Tasks to Invoke Synchronous Operations in Services and BPMN Processes" for information on how to invoke a synchronous service using a service task.

- Message events

  Message events enable you to invoke asynchronous services. To implement message events you must specify an asynchronous service in the implementation properties. See Section 20.3, "Using Message Events to Invoke Asynchronous Services and Asynchronous BPMN Processes" for information on how to invoke an asynchronous service using a message events.

- Send and receive tasks

  Send and receive tasks enable you to invoke asynchronous services. To implement a send task or a receive task, you must specify an asynchronous service in the implementation properties. See Section 20.7, "Using Send and Receive Tasks to Invoke Asynchronous Services and Asynchronous BPMN Processes" for information on how to invoke a synchronous service using a service task.

15.6 Using References in Oracle BPM

The following flow objects enable you to define an interface using a reference component from the business catalog:

- Message events

  See Section 21.5, "Using Message Events with an Interface from the Business Catalog to Define Your Process Interface" for more information on how to use an interface from the business catalog to define a process interface using message events.

- Receive task

  See Section 21.9, "Using Send and Receive Tasks with an Interface from the Business Catalog to Define Your Process Interface" for more information on how to to use an interface from the business catalog to define a process interface using receive tasks.

15.7 Customizing Services and References

The interfaces of some services and references you use in your process might be too complex or use names that do not clearly convey their use. These interfaces are not appropriate for a process analyst. You can customize these services and references to hide their complexity and make them more suitable for a business analyst. You might also customize a service or a reference to make your process easier to understand for other process developers.

Customizing a service or a reference enables you to:

- Change the name of the service or reference
- Store the service or reference in a user-defined module
- Add a description for the service or reference
- Hide the operations that you do not use
- Add a display name for each of the operations
- Add a description for each of the operations

When you customize a service or a reference, the service or reference disappears from the predefined modules where they were stored and Oracle BPM Studio replaces their uses by the customized component.

If you delete the customized service or reference, then the service or reference appears back in the corresponding predefined module, unless you remove either of them from the SOA composite.

Note: If you delete the original service or reference, Oracle BPM Studio does not delete the customized service or reference. You must delete the customized service or reference manually or create a new service or reference with the same name as the one you deleted.

15.7.1 How to Customize a Service or a Reference

You can customize a service or a reference to make it more suitable for a business analyst and easier to understand for process developers.

To customize a service or a reference:

1. In the BPM Project Navigator, right-click the service or the reference.
2. Select **Customize Service**.

   The Customize Adapter Service dialog box appears.
3. In the **Name** field, enter a name for the customized service or reference.
4. Click **Browse** and select a module to store the customized service or reference.
5. Optionally, enter a description for the customized service or reference.
6. From the operations list, select the operations to appear in the customized service or reference.
7. Edit the operations to customize them.

   See Section 15.7.2, "How to Customize an Operation" for information on how to customize an operation.
8. Click **OK**.

15.7.2 How to Customize an Operation

When you customize a service or a reference, optionally you can customize the operations it contains.

To customize an operation:

1. From the Operations table, select an operation.
2. Click **Edit**.

   The Edit Operations dialog appears.
3. In the **Display Name** field, enter the customized name for the operation.
4. In the **Description** field, enter a description for the operation.
5. If the operation requires input or output arguments, then you can provide a description for them.
6. Click **OK**.

15.7.3 What Happens When You Customize a Service or a Reference

The customized service or reference appears in the module you chose to store it in. The component in the Services or References predefined module disappears.

If there are any BPMN processes that use the component you customized, Oracle BPM Studio automatically updates the implementation of the activities in those processes to use the customized service or reference.

If you delete the customized service or reference, then it appears back in the corresponding predefined module.

16

Using Business Rules

This chapter describes how to implement business rule tasks in Oracle BPM. You can use an existing business rule component created using the SOA Business Rule editor, or you can create a new business rule component using the simplified interface Oracle BPM Studio provides.

This chapter includes the following sections:

- Section 16.1, "Introduction to Business Rules in Oracle BPM"
- Section 16.2, "Assigning an Existing Business Rule to a Business Rule Task"
- Section 16.3, "Creating a Business Rule from Oracle BPM Studio"

For detailed information about Oracle Business Rules, see *Oracle Fusion Middleware User's Guide for Oracle Business Rules*.

16.1 Introduction to Business Rules in Oracle BPM

The business rule task requires you to define a business rule to implement it. You can use an existing business rule or define a new one.

If your project contains business rules, then they automatically appear in the business catalog.

You can add new business rules to the business catalog in the following ways:

- Using Oracle BPM
- From the SOA New Gallery
- From the SOA Composite editor

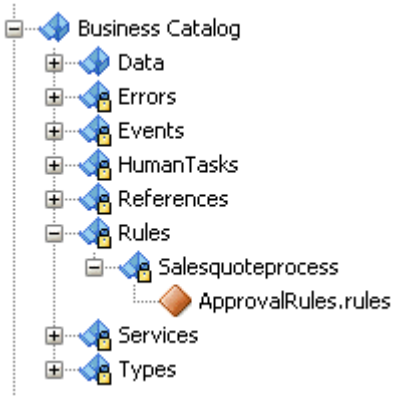
The business catalog displays the business rules in your project in the predefined module Rules. It stores each rule in a module named as the package of the business rule dictionary.

When you double-click a business rule component in the business catalog, Oracle BPM Studio opens the SOA Business Rules editor. You can edit the business rule using this editor.

For information on how to define Oracle Business Rules, see *Oracle Fusion Middleware User's Guide for Oracle Business Rules*.

Figure 16–1 shows a business rule component in the Sales Quote example.

Figure 16–1 Business Rules Components in the Business Catalog



This figure shows the Rules predefined module in the business catalog. The Rules predefine module contains a module *Salesquoteprocess* that in turn contains a business rule component named *ApprovalRules*.

****

16.1.1 Using Business Rules in a BPMN Process

Business rules enable you to determine the flow of your processes based on a group of rules you define.

The business rule task enables you to associate the following:

- Data object values to the input arguments of a business rule
- The output arguments of a business rule component to data objects

When a token arrives at a business rule task, the BPMN Service Engine invokes Oracle Business Rules Engine using the input arguments defined in the data association of the business rule task. The business rules engine evaluates the defined rules and returns output that contains the result. The BPMN Service Engine maps the output from the business rules engine to the data objects in the process using the data association defined for the business rule task.

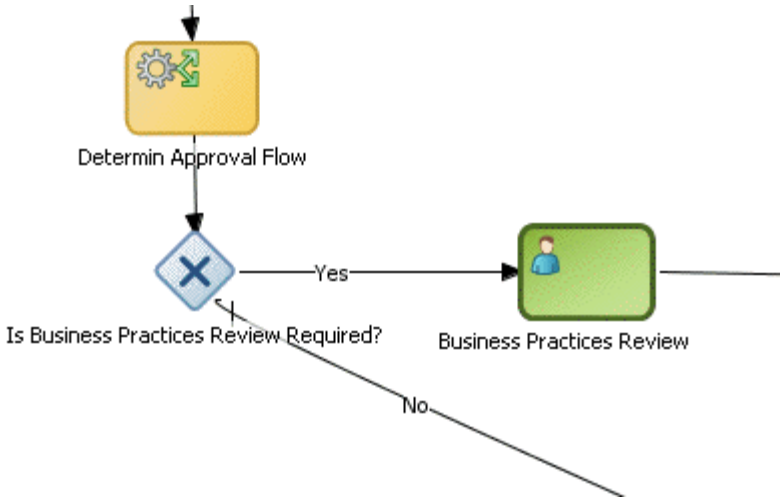
After a business rule task, you can add an exclusive gateway that determines the flow of the process based on the value of a data object that contains the result of running the business rule task.

In the Sales Quote example, the business rule task determines the approval flow for each sales quote in the following way:

- The business rule task invokes the business rule component providing a Quote business object as an input argument.
- The business rule component evaluates the defined rules using the provided input.
- The business rule component returns an ApprovalFlow business object that contains the result of evaluating the defined rules.
- The business rule task data association maps the result of running the business rule to the approvalFlow process data object.
- The exclusive gateway Is Business Practices Review Required determines the flow based on the value of the approvalFlow process data object.

Figure 16–2 shows a business rule task in the Sales Quote example.

Figure 16–2 Business Rule Task in the Sales Quote Example



This figure shows a business rule task in the Sales Quote example process. An exclusive gateway follow the business rule task. The exclusive gateway is based on the result of running the business rule task.

****

16.2 Assigning an Existing Business Rule to a Business Rule Task

You can assign existing business rules to a business rule task. You can implement a business rule task using a business rule that you created using the Business Rule wizard or that existed in the SOA project you used as the basis for the BPM project.

All the business rule components that your project contains appear in the Rules predefined module in the business catalog. They also appear in the SOA composite. If there are business rule tasks in your BPMN processes that use a business rule component, then the SOA composite shows a wire between them.

16.2.1 How to Assign an Existing Business Rule to a Business Rule Task

You can reuse existing business rules to implement the business rules tasks in your BPMN processes.

To assign an existing business rule to a business rule task:

1. Right-click the **business rule task**.
2. Select **Properties**.
3. Click the **Implementation** tab.
4. Click the **Browse** button next to the **Business Rule** field.

   The Type dialog box appears.
5. Select a business rule from the list or enter the name or part of the name in the **Search** field to search for a business rule.
6. Click **OK**.

   The Type dialog box closes, and the **Business Rule** field shows the business rule you selected.

7. From the Decision Function list, select a decision function.
8. Click **OK**.

   The Business Rule Task Properties dialog box closes and saves the implementation you selected for the business rule task.

16.2.2 What Happens When You Assign an Existing Business Rule to a Business Rule Task

The business rule task implementation uses the selected business rule component and the selected decision function.

When the BPMN Service Engine runs the business rule task, it invokes the Oracle Business Rules Engine using the input arguments defined in the business rule task data association. The Oracle Business Rules Engine evaluates the rules using the provided input argument and returns an output argument that contains the result of this evaluation.

16.2.3 How to Edit the Business Rule Associated to a Business Rule Task

You can launch the Business Rule Editor from a business rule task, to edit the associated business rule.

To edit the business rule associated to a business rule task:

1. Right-click the **business rule task**.
2. Select Open Business Rule.

   The Business Rule Editor appears.

16.3 Creating a Business Rule from Oracle BPM Studio

You can create a business rule using the simplified interface Oracle BPM Studio provides. You can access this interface from the business rule task configuration dialog box.

The simplified business rule creation interface enables you to create a business rule with one decision function. When you create the business rule, you can configure the following properties:

- **Name of the Business Rule**

  Oracle BPM Studio uses this name to create the business rule component.

- **Input and output**

  Specifies the input and output parameters for the default decision function Oracle BPM Studio automatically adds to the business rule component.

  Oracle BPM Studio uses these parameters to create the data association for the business rule task. The parameters of the decision function must be complex data objects created based on an XML schema. The XML schema must contain only one element. You must not use types from the WSDL.

- **Dictionary package**

  Specifies the Java package to which your rule dictionary belongs, for example, com.example.

- **Name of the decision function**

Oracle BPM Studio uses this name to add a default decision function to the business rule you create.

After you create the business rule with the simplified interface you can edit it using the editor included in Oracle SOA Suite.

16.3.1 How to Create a Business Rule from Oracle BPM Studio

You can create a business rule component from Oracle BPM Studio from the Implementation Properties dialog box of a business rule task.

To create a business rule from Oracle BPM Studio:

1. Right-click the **Business Rule** task.
2. Select **Properties**.
3. Click the **Implementation** tab.
4. Click the **Add** button next to the **Business Rule** field.

   The Create Business Rule dialog box appears.
5. In the **Name** field, enter a name to identify the business rule.
6. Configure the input and output of the business rule.

   See Section 16.3.2, "How to Add Input and Output Arguments When Creating a Business Rule Component" for more information on how to configure the input and output of a business rule.
7. Optionally, configure the advanced properties of the business rule.

   See Section 16.3.3, "How to Configure the Advanced Properties When Creating a Business Rule Component" for more information on how to configure the advanced properties of a business rule.
8. Click **OK**.

   The Create Business Rule closes and creates the business rule. The **Business Rule** field in the Business Rules Task Properties dialog box shows the business rule you created.
9. From the Decision Function list, select a decision function.
10. Click **OK**.

    The Business Rule Task Properties dialog box, closes and saves the implementation you created for the business rule task.

16.3.2 How to Add Input and Output Arguments When Creating a Business Rule Component

The data objects you add as input or output arguments must use business objects based on external types as their types.

To add input and output arguments when creating a business rule component:

1. In the Input and Output Data Objects section, click the **Add** button.
2. Select the type of argument you want to add.

   The Data Object dialog box appears.
3. Select a data object from the Data Object dialog box and drag it to the table.

The input or output argument appears in the table.

16.3.3 How to Configure the Advanced Properties When Creating a Business Rule Component

To configure the advanced properties when creating a business rule component:

1. Click the **Advanced** tab.
2. In the **Package** field, enter the name of the package in which to store the rules dictionary.
3. In the **Decision Function** field, enter a name for the decision function that the simplified interface creates in the business rule component.

16.3.4 What Happens When You Create a Business Rule Task from Oracle BPM

Oracle BPM Studio creates a business rule component. You can edit this business rule component using the SOA Business Rule editor in the same way you edit a component created using Oracle SOA Suite.

The business rule task uses the business rule component for its implementation.

Part VI

Controlling the Process Flow

This part describes how to implement the different BPMN flow object that you can use to control the process flow. It also describes how to communicate with other BPMN processes and external services.

This part contains the following chapters:

17 Controlling the Process Flow

This chapter briefly describes the different flow objects you can use to control flow in a process. It contains links to the chapters that describe these flow objects with more detail. It also contains a description of the markers you can define for subprocesses.

This chapter includes the following sections:

17.1 Introduction to Controlling the Process Flow

Oracle BPM provides different structures to control the flow of a process. These structures enable you to decide which path a process instance takes based on different conditions.

The structures that allow you to control the flow of a process are:

- Gateways
- Timer Events
- Errors
- Message Events
- Send and Receive Tasks
- Loop Markers
- Multi-Instance Markers

17.1.1 Gateways

Gateways are flow objects that enable you to fork the flow of a process. Depending on the type of gateway the instance follows one ore more outgoing sequence flows coming out of a gateway, or multiple copies are created to run these branches in parallel.

For more information about gateways, see Section 6.7, "Controlling Process Flow Using Gateways".

17.1.2 Timer Events

Timer events enable you to define the path a process instance takes based on a time condition. For more information about timer events, see Chapter 18, "Adding Delays, Deadlines, and Time Based Cycles to Your Process".

17.1.3 Errors

Error events enable you to define how a process handles an abnormal situation. You can use error events to define different process flows for each of the errors that may occur in a business process. For more information about error events, see Chapter 19, "Handling Errors".

17.1.4 Message Events

Message events enable you to define a process flow based on the occurrence of a certain event. Generally you use message events to asynchronously invoke an external service or another BPMN process. For more information about message events, see Chapter 20, "Communicating With Other BPMN Processes and Services".

17.1.5 Send and Receive Tasks

Message events enable you to define a process flow based on the occurrence of a certain event. Generally you use message events to asynchronously invoke an external service or another BPMN process. For more information about message events, see Chapter 20, "Communicating With Other BPMN Processes and Services".

17.1.6 Loop Markers

Loop markers enable you to run a subprocess multiple times based on a certain condition. For more information about loop markers, see Section 17.2, "Introduction to Loop and Multi-Instance Markers in Subprocesses".

17.1.7 Multi-Instance Loop Markers

Multi-instance loop markers enable you to run a subprocess for each of the elements in a set of data. For more information about loop markers, see Section 17.2, "Introduction to Loop and Multi-Instance Markers in Subprocesses"

17.2 Introduction to Loop and Multi-Instance Markers in Subprocesses

You can configure subprocesses to run multiple times using loop and multi-instance markers.

To configure loop and multi-instance makers you must define expressions and conditions that specify how to repeat the subprocess.

Loop Markers

Loop markers enable you to run a subprocess multiple times based on condition. You can configure the loop marker to evaluate the condition before or after running the subprocess. You can also configure the loop marker to stop after a certain number of repetitions.

To configure a loop maker you must write a Loop Condition that determines if the BPMN Service Engine must continue to repeat the subprocess.

Multi-Instance Markers

Multi-Instance markers enable you to run a subprocess for each of the elements on a set of data. When the BPMN Service Engine runs a subprocess with a multi-instance loop marker it creates a set of instances, one for each element on the set of data. You can configure the multi-instance marker to process these instances in parallel or sequentially.

The following fields in a multi-instance loop marker require you to write an expression:

- **Loop Cardinality**

  This expression defines the number of tokens to create in the subprocess.

- **Completion Condition**

  This expression determines when to stop repeating the subprocess. The BPM Service Engine evaluates this condition every time a token completes the subprocess. If the condition evaluates to true, it considers the subprocess completed and the instance moves to the next flow object in the process.

17.2.1 How to Configure Loop Markers

You can configure a loop marker to run a subprocess multiple times.

To configure loop markers:

1. Right-click the subprocess.
2. Select **Properties**.
3. Click the **Loop Characteristics** tab.
4. Select **Loop**.
5. Specify the Loop Condition:
   1. Select the expression language.

      Possible options are Simple or XPath.

   2. In the text area below, write the condition that drives the loop.

      Optionally you can write the condition using the Expression Builder. To launch the Expression Builder click the **Expression Builder** button next to the text area.
6. Optionally, you can specify a maximum number of times for the loop to run:
   1. Select **Loop Maximum**.
   2. Specify a number.
7. Select **before** to evaluate the condition before running the flow object, or deselect it to evaluate the condition after running the flow object.
8. Click **OK**.

17.2.2 How to Configure Multi-Instance Markers

You can configure a multi-instance marker to run subprocess multiple times based on a set of data.

To configure multi-instance markers:

1. Right-click the subprocess.
2. Select **Properties**.
3. Click the **Loop Characteristics** tab.
4. Select **MultiInstance**.

5. Specify the Loop Cardinality:
    1. Select the expression language.

       Possible options are Simple or XPath.
    2. In the text area below, write the specifies the loop cardinality.

       Optionally you can write the condition using the Expression Builder. To launch the Expression Builder click the **Expression Builder** button next to the text area.
6. Optionally, you can specify the Completion Condition:
    1. Select the expression language.

       Possible options are Simple or XPath.
    2. In the text area below, write the condition that determines if the loop is completed.

       Optionally you can write the condition using the Expression Builder. To launch the Expression Builder click the **Expression Builder** button next to the text area.
7. Click the **Browse** button next to the Loop Data Output field, to specify the data output.

   You can select a data object or an attribute in a complex data object to pass to the subprocess. Generally the selected data object is a collection of items.
8. Click the **Browse** button next to the Loop Data Input field, to specify the data input.

   Select a data object or an attribute in a complex data object to assign the result of the subprocess.
9. Optionally, check the Is Sequential check box to specify that the each token must complete the subprocess before the next token starts to run the subprocess.
10. Click **OK**.

18

Adding Delays, Deadlines, and Time Based Cycles to Your Process

This chapter describes how to use timer events to add time conditions to your BPMN process. It describes how to use the different timer events to add delays and deadlines, and to run additional activities.

This chapter includes the following sections:

18.1 Introduction to Timer Events

Timer events enable you to control the flow of your process using a time condition.

You can use timer events for:

- 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
- Start a process
- Trigger a process periodically

Timer events are not based on the business calendar definitions.

Oracle BPM enables you to configure timers using:

- A specific date and time

  You can configure a timer event to fire on a certain date. You can specify a specific date or use a function to calculate the it.

- A relative time

You can configure a timer event to fire after an elapsed time. You can specify the elapsed time or use a function to calculate it. If the timer event is a start event or a non-interrupting boundary event, then it fires multiple times.

When you define a timer event as a boundary event you can choose to configure it as interrupting or non-interrupting.

When an interrupting timer event fires, the token leaves the main process flow to follow the flow the timer defines. The flow an interrupting event defines, can resume the main process flow

When an non-interrupting event fires, the BPMN Service Engine creates a copy of the token that is running the main process flow and routes that copy through the flow the timer event defines. The flow a non-interrupting event defines cannot resume the main process flow.

18.2 Adding a Delay to the Process Flow

You can add a delay to the process flow by adding an intermediate timer catch event. When the token arrives to the timer event it waits the time specified in the timer event before moving to the next activity in the process.

For example, in a process that updates multiple data bases you might want to add a timer activity that delays the process a few minutes, to ensure that all databases are updated when the process continues.

You can configure the intermediate timer catch event to wait until a specific date or to wait for a certain period. In both cases you can choose to use a fixed value or to use an expression that specifies the corresponding date or interval.

When you configure a timer intermediate event as a cycle, the timer event only runs one time. It waits until the specified interval passes and then the token continues moving through the rest of the process flow.

Figure 18–1 Delaying the Process Flow

This diagram shows an intermediate timer catch event that adds a delay to the process flow. The token remains in the timer activity until the time specified in the timer event passes.

****

18.2.1 How to Add a Delay to the Process Flow

You can add a delay between to flow objects.

To create a delay until a specified date in the process flow:

1. Locate the point in your process where you want to add the delay.
2. From the Component Palette, from the Catch Events section, select **Timer**.
3. Drop the timer event in the point where you want to add the delay.

4. If you want to delay the process until a specific date, then you must configure the timer event as time date. If you want to delay the process for a certain period, then you must configure the timer start event as cycle.

   See Section 18.7.1, "How to Configure a Timer Event To Use a Specific Date and Time" for more information on how to configure a timer event as time date.

   See Section 18.7.3, "How to Configure a Timer Event to Use an Interval" for more information on how to configure a timer event as cycle.

18.2.2 What Happens When You Add a Delay to the Process Flow

A token that arrives to the intermediate timer event remains in the timer event until the time specified by the timer event arrives. If you configure the timer event to use a date, then the token remains in the timer event until the specified date. If you configure the timer event to use a cycle, then the token remains in the timer event until the specified time passes.

18.3 Designing a Process to Start Based on a Time Condition

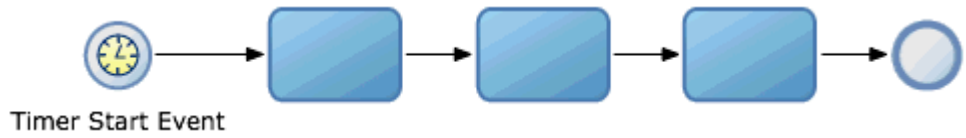
You can add a timer start event to your process to configure it to be triggered based on a time condition. When the time condition specified in the timer start event evaluates to true, the BPMN Service Engine creates a new instance in the process.

For example, in a process to report working hours you can add a timer start event that creates an instance in the process one time a day.

You can configure your process to start on a specific date or to periodically create an instance. In both cases you can choose to use a fixed value or to use an expression that specifies the corresponding date or interval

When deploying a process containing a timer start event specifying a past date, the BPMN Service Engine automatically creates an instance of the process.

Figure 18–2 Starting a Process Based on a Time Condition



This diagram shows a process that starts with a timer event. The timer start event creates an token each time its time condition evaluates to true.

****

18.3.1 How to Design a Process to Start Based on a Time Condition

You can design your process to start when a specific date arrives or to periodically start after a certain elapsed time.

To design a process to start based on a time condition:

1. Open the BPMN process.
2. If you want your process to have a single start event, then you must right-click the start event and select **Change Trigger Type** and then **Timer**.

If you want your process to have multiple start events, then you must select a timer start event from the Start Events section in the Component Palette. Drop the timer start event on you process. Right-click the timer start event and select **Properties**.

3. If you want the process to start on a specific date, then you must configure the timer start event as time date. If you want the process to start after a certain period, then you must configure the timer start event as cycle.

   See Section 18.7.1, "How to Configure a Timer Event To Use a Specific Date and Time" for more information on how to configure a timer event as time date.

   See Section 18.7.3, "How to Configure a Timer Event to Use an Interval" for more information on how to configure a timer event as cycle.

18.3.2 What Happens When You Design a Process to Start Based on a Time Condition

The BPMN Service Engine creates an instance in the process each time the time condition in the timer start event evaluates to true. If you configure the timer start event to use a specific date, then the BPMN Service Engine creates an instance when the specified date arrives. If you configure the timer start event to use a cycle, then the BPMN Service Engine periodically creates an instance in the process.

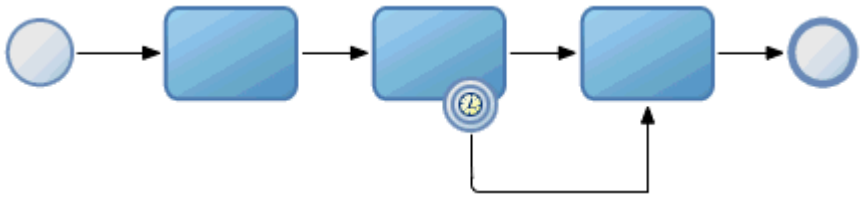
18.4 Configuring a Deadline for an Activity

You can configure a deadline for an activity using an interrupting timer catch event configured as a boundary interrupting event that leads to another point of the process. If the token remains in the activity for longer than expected or beyond a certain date, then the timer catch event gets triggered and interrupts the process flow.

You can configure the deadline to happen on a specific date, or after the token spends a certain time in the activity. In both cases you can specify a fixed date or interval or an expression that calculates the corresponding date or interval.

For example, in an purchase order process, you might want to configure the activity that gets the credit card approval to wait the approval for a day. And if the approval takes longer, then direct the token to an activity that sends a message to the customer.

Figure 18–3 Activity Deadline



This diagram shows a BPMN process that contains an activity that has a deadline. If the token stays in the activity longer than the timer event specifies, then the timer event fires and the token moves to the next activity in the process.

****

18.4.1 How to Configure a Deadline for an Activity

You can configure a deadline for an activity so that the token moves to another activity after the deadline expires. You can specify to which activity the token moves after the deadline expires.

To configure a deadline for an activity:

1. Locate the activity in your process for which you want to configure a deadline.
2. From the Component Palette, from the Catch Events section, select **Timer**.
3. Drop the timer event over the activity.

   The timer event becomes a boundary event. A sequence flow coming out from the boundary timer catch event appears.
4. Place the cursor over an end event and click to drop the sequence flow there.
5. If you want the deadline to happen on a specific date, then you must configure the boundary timer catch event as time date. If you want the deadline to happen after a certain period, then you must configure the boundary timer catch event as cycle.

   See Section 18.7.1, "How to Configure a Timer Event To Use a Specific Date and Time" for more information on how to configure a timer event as time date.

   See Section 18.7.3, "How to Configure a Timer Event to Use an Interval" for more information on how to configure a timer event as cycle.
6. In the Implementation tab, in the Timer Properties dialog, select **Interrupting Event**.

18.4.2 What Happens When You Configure a Deadline for an Activity

If the activity is still running when the timer event fires, then the token quits the activity and move to a different point in the process. The timer event fires because a certain date arrives or because the specified period passes, depending on how you configured the timer event.

18.5 Configuring a Deadline for a BPMN Process

You can configure a process deadline for your process using an event subprocess that starts with an interrupting timer start. After a certain time passes or a date arrives, the timer event fires. If the token is still in the process then it moves to the event subprocess.

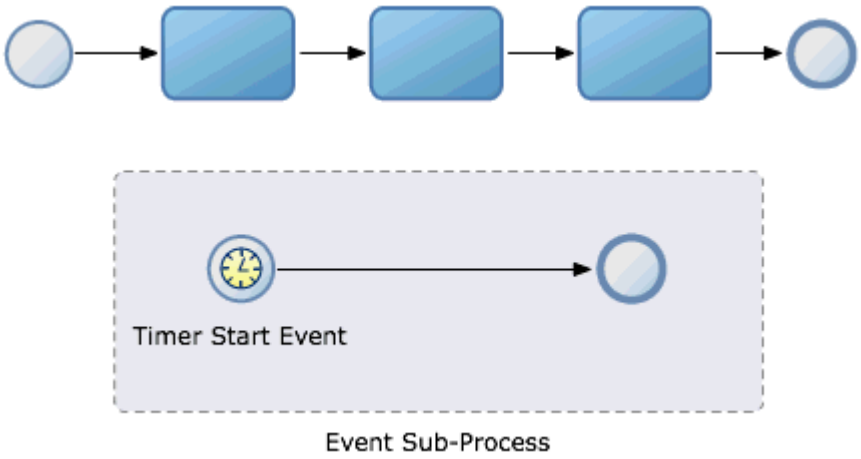
The timer event is only active while the token remains in the process.

You can configure the deadline to happen on a specific date, or after the token spends a certain time in the activity. In both cases you can specify a fixed date or interval or an expression that calculates the corresponding date or interval.

For example, in a purchase order process, you can configure the process so that if the token stays in the process for more than three months, then it automatically ends the process.

You might want to use an error end event in the event subprocess, so that the process does not finish running successfully.

Figure 18–4 Process Deadline



This diagram shows a process that contains a process deadline. If the token stays in the process longer than the timer start event in the event subprocess specifies, then the timer start event in the event subprocess fires. When the timer start event in the subprocess fires, the token moves to the event-sub process.

****

18.5.1 How to Configure a Deadline for a BPMN Process

You can configure a deadline for a BPMN process. You can choose to terminate the process flow or to run a group of flow object when the deadline expires.

To configure a deadline for a BPMN process:

1. Open the BPMN process.
2. From the Component Palette, from the Activities section, select **Event Subprocess**.
3. Drop the event subprocess in the process.
4. Configure the start event in the event subprocess to be a timer event:
    1. Right-click the start event in the event subprocess.
    2. Select **Properties**.
    3. Click the **Implementation** tab.
    4. From the Implementation Type list, select **Timer**.
    5. Select **Interrupting Event**.
    6. If you want the deadline to happen on a specific date, then you must configure the timer event as time date. If you want the deadline to happen after a certain period, then you must configure the timer event as cycle.

       See Section 18.7.1, "How to Configure a Timer Event To Use a Specific Date and Time" for more information on how to configure a timer event as time date.

       See Section 18.7.3, "How to Configure a Timer Event to Use an Interval" for more information on how to configure a timer event as cycle.

18.5.2 What Happens When You Configure a Deadline for a BPMN Process

If the token stays in the process longer than specified by the interrupting timer event, then the timer event fires. When the timer start event in the event subprocess fires the token leaves the process and moves to the event subprocess.

18.6 Running Additional Activities

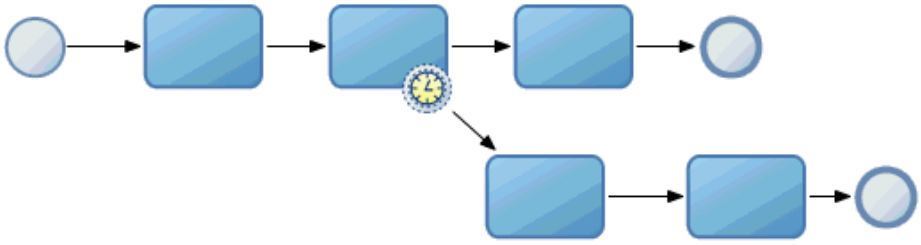
While running an activity or a process you can run additional activities based on a time condition. You can choose to trigger the additional activities periodically or on a certain date.

Typically you run additional activities when the activity you are currently running takes a long time to finish. For example, if you run a service that takes twenty hours to update a database, then you might want to send an e-mail to inform progress of the update to the interested parties.

The timer event is only active while the token remains in the activity.

You can also run additional activities while a process is running. These activities run in parallel to the main process flow.

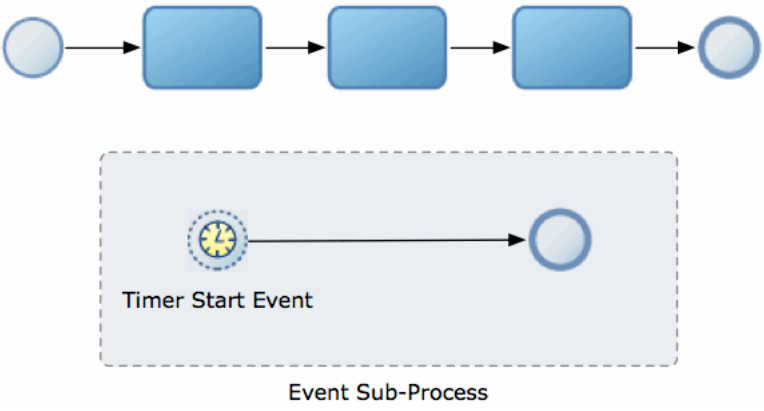
Figure 18–5 Running Additional Activities While an Activity is Running



This diagram shows a process that contains an activity that runs additional activities while it is running. If the token is still in the activity when the non-interrupting timer event fires, then the BPMN Service Engine creates a copy of the token and routes it through the process flow the timer event defines.

****

Figure 18–6



This diagram shows a process that runs activities in parallel to the main process flow. The timer start event in the event subprocess may fire multiple times while the process

is running. When the timer start event fires, the BPM service engine creates a copy of the token and routes it thought the event subprocess flow.

****

18.6.1 How to Run Additional Activities While an Activity is Running

You can run a parallel process flow while an activity is running. Generally you design a parallel process flow to trigger after a certain time when you know that the main activity might take long to complete.

To run additional activities while an activity is running:

1. Locate the activity to run in parallel to the additional activities.
2. Create an additional process flow by adding activities connected by sequence flows, outside the main process flow.
3. Add a timer event as a boundary to the activity.

   A sequence flow for you to connect to an activity appears.
4. Connect the sequence flow to the additional process flow you created.

18.6.2 What Happens When You Run Additional Activities While an Activity is Running

If the token is still in the activity when the non-interrupting fires, then the BPMN Service Engine creates a copy of that token and routes it through the flow that the timer event defines. The timer might fire multiple times while the activity in the main process flow is running.

18.6.3 How to Run Additional Activities While a Process is Running

You can run additional activities while the main process flow is running. Generally you design a parallel process flow to trigger after a certain time when you know your process might take long to complete.

To run additional activities while a process is running:

1. Add a subprocess event to your process.
2. Right-click the start event in the subprocess event.
3. Select **Properties**.
4. Click the **Implementation** tab.
5. From the Implementation Type list, select **Timer**.
6. Verify that the Interrupting Event option is not selected.
7. If you want the additional activities to run on a specific date, then you must configure the timer event as time date. If you want the additional activities to run periodically, then you must configure the timer event as cycle.

   See Section 18.7.1, "How to Configure a Timer Event To Use a Specific Date and Time" for more information on how to configure a timer event as time date.

   See Section 18.7.3, "How to Configure a Timer Event to Use an Interval" for more information on how to configure a timer event as cycle.
8. Add the additional activities to the subprocess event.

18.6.4 What Happens When You Run Additional Activities While a Process is Running

When the timer start event in the event subprocess fires, the BPMN Service Engine creates a copy of the token in the main process flow. The copy of the token in the main process flow follows the additional process flow the subprocess event defines. The timer start event may fire multiple times while the main process flow is running.

18.7 Configuring Timer Events

You can configure timer event to fire on a specific date and time, or to fire after a certain time passes. In both cases you can choose to provide a fixed time value or an expression that calculates it.

18.7.1 How to Configure a Timer Event To Use a Specific Date and Time

You can configure a timer event to use a specific date and time. You can provide the date and time or use an expression to calculate it.

To configure a timer event to use a specific date and time:

1. Right-click the timer event.
2. Select **Properties**.
3. Click the **Implementation** tab.
4. Select **Time Date**.
5. Provide a date.

   The following options are available to provide a date:

   - Click the calendar button next to the Date field. Select a date and enter a time and close the calendar dialog.
   - Enter the date in the Date field. For example: Jan. 18, 2010 4:31:10 PM
   - Select Use Expression and provide an expression that returns a Date.

     See Section 22.4, "Writing Expressions in Timer Events" in Chapter 18, "Adding Delays, Deadlines, and Time Based Cycles to Your Process" for more information.

     Note: The date and time you specify correspond to the time zone the BPMN Service Engine uses.

6. Click **OK**.

18.7.2 What Happens When You Configure a Timer Event to Use a Specific Date and Time

The timer event fires on the specified date and time. If you used an expression to specify the date and time, then the engine evaluates this expression to determine when to fire the timer event.

18.7.3 How to Configure a Timer Event to Use an Interval

You can configure a timer event to use an interval. You can specify the interval or use an expression to calculate it.

To configure a timer event to use an interval:

1. Right-click the timer event.
2. Select **Properties**.
3. Click the **Implementation** tab.
4. Select **Cycle**.
5. Provide a time interval or select Use Expression and write an expression that returns an Interval.

   See Section 22.4, "Writing Expressions in Timer Events" for more information
6. Click **OK**.

18.7.4 What Happens When You Configure a Timer Event to Use an Interval

The timer event fires periodically, waiting the time the interval specifies. If the timer event is a start event or a non-interrupting boundary event, then it fires multiple times. If the timer event is an intermediate timer event or an interrupting boundary event, then it waits for the specified interval before firing, but it fires only one time.

19

Handling Errors

This chapter describes how to handle errors that occur when running a business process. Oracle BPM provides you with an exception component that enables you to model errors and multiple BPMN structures that you can use to handle those errors while running the process.

This chapter includes the following sections:

19.1 Introduction to Error Handling

There might be situations when an unexpected problem occurs causing your process to fail. There are two types of errors: system errors and process errors.

System errors are the consequence of a failure in the software or hardware infrastructure where the BPMN Service Engine is running. A system error can have many causes. The following are examples of problems that can cause a system error:

- Failure in the database connection
- Connectivity loss
- Problem with the hard disk

To recover from system errors within the process flow you can use system exceptions.

If you do not handle a system exception in your process, you can recover from them using the fault recovery system provided by Oracle Enterprise Manager. See *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite* for more information about the Oracle Enterprise Manager fault recovery system.

Process errors are problems that interfere with the regular development of your process. For example, in a purchase order process, if there is no stock for the requested item then you cannot continue with the regular process flow. You can handle these

unexpected situations within the process flow. One way to handle the situation in this example by letting the customer cancel the order or save it for later.

The following are typical examples of unexpected situations within a process:

- Lack of stock
- Workload limit exceeded
- Expense limit exceeded
- Feedback past due
- Credit card authentication problems

When an exception occurs in a process, it affects the state of the SOA composite that contains that BPMN process. For more information on how exceptions affect the state of the SOA composite, see Section 23.1.4, "How Do BPMN Errors Affect the SOA Composite Status".

19.1.1 Handling Errors Using Exceptions

Oracle BPM uses business exceptions to represent unexpected situations that can occur while running a business process.

You can design how to handle an exception as part of the business process, but it is something that occurs outside of the usual flow of a process. The use of business exceptions enables you to create less complicated processes where the main flow follows the typical use cases, and there is a separate flow to handle the process exception.

19.2 Using Business Exceptions

Business exceptions are considered a normal part of the process design, rather than an error.

When you add a component to the business catalog, if the services in the component specify that they can produce errors, then these errors appear as business exceptions in the business catalog in the Errors predefined module.

An exception can arise when you invoke a service. You can handle these exceptions using a boundary error catch event or an event subprocess.

You can also define business exceptions in the business catalog. Then, you can use those business exceptions in an error end event that is triggered under a certain condition. The error end event generates the exception, and the parent process can handle the exception.

19.3 Using System Exceptions

System exceptions represent low level errors that may occur while running a process. In some cases you may require to handle this low level errors within your process.

To handle a system exception within the process flow you must catch the exception and configure the error catch event to use system exceptions.

System exceptions may occur while running a service or another BPMN process. You also design your process to throw certain system exceptions. The only exception that you can use in a throw or end event is Rollback. All the other supported system exceptions are only available for start of catch error events.

System exceptions contain an errorInfo attribute of type Any. You can assign any value to this attribute. Because its type is Any this value can belong to any type. Generally you use this attribute to store the cause of the exception or important information for troubleshooting the application.

You can only view the list of available system exceptions from the Implementation Properties of an error event.

Table 19–1 describes the supported system exceptions. It also specifies the module where the system exception resides and the error events that can use the specified system exception.

Table 19–1 System Exceptions

System Exception	Module	Description	Error Event
AssertFailure	Bpel	Indicates that the specified assertion failed.	Catch, Start
BindingFault	Bpel	Indicates that the preparation of the operation invoked in a flow object failed. For example, the WSD loading failed. You cannot retry the invocation after a BindingFault, recovering from this error generally requires human intervention.	Catch, Start
InvalidVariables	Bpel	Indicated that the variables used are not valid.	Catch, Start
RemoteFault	Bpel	Indicates that there was a problem invoking a service in a flow object. For example, the remote service returned a SOAP fault.	Catch, Start
Timeout	Soap	Indicates that the service exceeded the response time out period.	Catch, Start
ConflictingReceive	Soap	Indicates that there are multiple receive activities to respond to the invoked operation.	Catch, Start
ConflictingRequest	Soap	Indicates that there are multiple requests on the same partner link for the invoked operation.	Catch, Start
CorrelationViolation	Soap	Indicates that the message does not provide the required correlation information.	Catch, Start
ForcedTermination	Soap	Indicates the service terminated because a SOAP fault occurred.	Catch, End
InvalidReply	Soap	Indicates that the reply does not contain the correlation information required by the corresponding receive.	Catch, Start
MismatchedAssignmentFailure	Soap	Indicates the assigned types are incompatible.	Catch, Start
RepeatedCompensation	Soap	Specifies that a compensation handler is invoked multiple times.	Catch, Start
SelectionFailure	Soap	Indicates there was an error running a selection operation.	Catch, Start
UninitializedVariable	Soap	Indicates that the variable you are accessing is not initialized.	Catch, Start
Rollback	Soap	Enables the receiver of the exception to rollback the current JTA transaction from within the process flow.	Throw, End

19.4 Typical Flow of an Exception

The flow of a system or a business exception depends on where the exception occurred.

Exceptions can occur while running the following:

- a task
- a subprocess
- a reusable process

19.4.1 Typical Flow of an Exception Thrown in a Task

The following describes what happens when the BPMN Service Engine runs a task that causes an exception.

1. The BPMN Service Engine runs a task that starts a service that can throw an exception.
2. The task fails with a SOAP error that the BPMN Service Engine converts into an exception.
3. If the task has a boundary catch error event attached, then the instance follows the flow defined by the boundary catch error event to handle the exception. The exception handling flow may resume the main process flow. If it does not resume the main process flow, then the process ends in the boundary catch error event.

   If the task does not have a boundary catch error event associated with it, then the exception propagates to the process level.
4. At the process level, the following options are possible:
   - If the process does not contain an event subprocess that can catch the exception and you did not define a fault policy, then the BPM Service Engine logs this error to the Oracle Enterprise Manager fault recovery system. See *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite* for more information about the Oracle Enterprise Manager fault recovery system.
   - If the process contains an event subprocess with a start event of type error configured to catch that exception, then the instance continues through the exception handling flow. When the instance completes the exception handling flow, the process ends.

19.4.2 Typical Flow of an Exception in a Subprocess

The following sequence describes what happens when the BPMN Service Engine runs a subprocess that causes an exception.

1. The BPM Service Engine runs a subprocess that contains a task that invokes a service that can throw an exception, or an end error event.
2. One of these events happens:
   - The task throws an exception.
   - The subprocess ends with an error event.
3. If the exception occurs in a task and the task has a boundary catch error event or the subprocess contains an event subprocess that can handle the exception, then the exception does not propagated to the parent process.

   If the subprocess ends with an error event, or the exception occurs in a task and is not handled, then the exception propagates to the parent process.
4. The parent process can handle the exception if:
   - the subprocess has a boundary catch event attached.
   - it contains an event subprocess configured to catch the exception.

   If the parent process cannot handle the exception, then it propagates it to its parent process. If there is no parent process, then the exception is logged to the Enterprise Manager fault recovery system. See *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*, for more information about the Enterprise Manager fault recovery system.

19.4.3 Typical Flow of an Exception in a Reusable Process

The following sequence describes what happens when the BPMN Service Engine runs a call activity that invokes a reusable subprocess that causes an exception.

1. The BPM Service Engine runs a reusable process that contains a task that invokes a service that can throw an exception, or an end error event.
2. One of these events happens:
   - The task throws an exception.
   - The reusable process ends with an error event.
3. If the exception occurs in a task and the task has a boundary catch error event or the reusable process contains an event subprocess that can handle the exception, then the exception does not propagated to the parent process.

   If the subprocess ends with an error event, or the exception occurs in a task and is not handled, then the exception propagates to the parent process.
4. The parent process can handle the exception if:
   - the call activity has a boundary catch event attached.
   - it contains an event subprocess configured to catch the exception.

   If the parent process cannot handle the exception, then it propagates it to its parent process. If there is no parent process, then the exception is logged to the Enterprise Manager fault recovery system. See *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*, for more information about the Enterprise Manager fault recovery system.

19.5 Handling Exceptions in a Business Process

You can handle the exceptions that occur in an activity using the following:

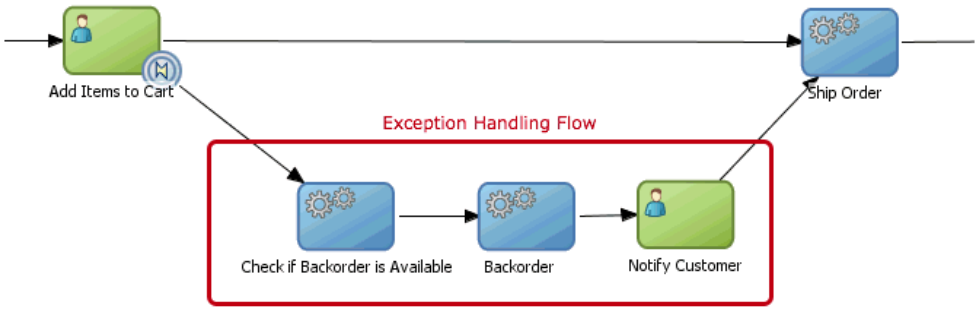
- A boundary error catch event
- An event subprocess

Boundary error catch events enable you to resume the main process flow after handling the exception.

If you want to reuse the exception handling flow for multiple tasks in your process, then event subprocesses are more efficient than boundary catch events. Event subprocesses enable you to define a cleaner process with less effort because the catch error event is located within the event subprocess. To reuse an exception handling flow using boundary catch events, you must define a boundary catch event for each of the tasks, and then connect those boundary events to the exception handling flow.

Figure 19–1 shows a process that handles an error using a boundary error catch event.

Figure 19–1 Boundary Error Catch Event



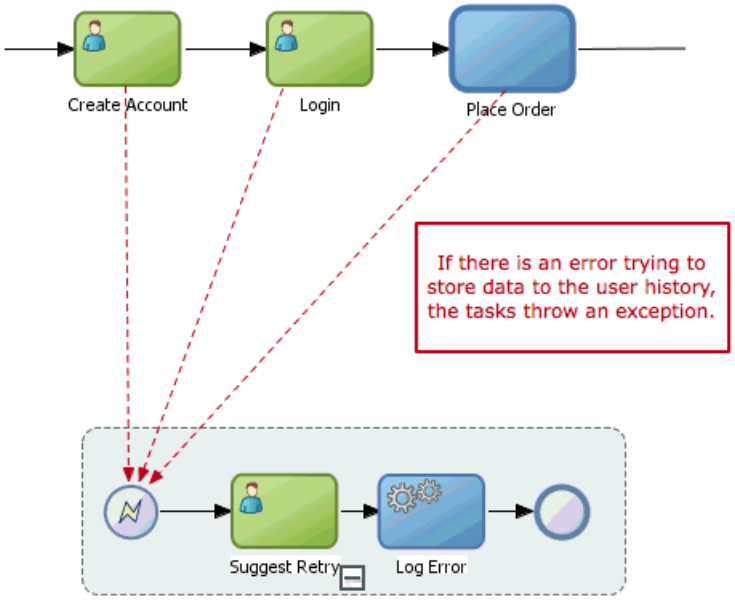
This figure shows a business process that throws an exception when there is no stock for any of the items added to the cart. The user task has a boundary catch error event that handles the exception by back ordering the product and then resumes the main process flow.resume

****

Event subprocesses also enable you to define data objects that you can access only from within the event subprocess, in the same way that subprocesses enable you to define their own data objects.

Figure 19–2 shows a process that handles an error using a an event subprocess.

Figure 19–2 Event Subprocess with a Start Error Event



This figure three tasks that throw an exception when they are unable to write data to the user history. The process contains an event subprocess that handles the exceptions by suggesting the user to retry later and logging the error so that it is available for the system administrator to review it.

****

19.5.1 How to Handle an Exception Using a Boundary Error Catch Event

If you know that running a flow object can cause an exception, then you can design your process to handle the exception using a boundary error catch event.

To handle an exception using a boundary error catch event:

1. Create an exception handling flow.

   After handling the exception, this flow can resume the main process or end the process.

2. From the Component Palette, from the Catch Events section select **Error Event**.

3. Drop the error event over the task that throws the exception.

   You can place the event in any part of the border of the task.

   When you drop the error event, a sequence flow appears that you can connect to the exception handling flow.

4. Connect the sequence flow to the exception handling flow.

5. Right-click the boundary catch error event.

6. Select **Properties**.

7. Click the **Implementation** tab.

8. Configure the implementation properties to catch a business or system exception.

   For information on how to configure the implementation properties to catch business exceptions, see Section 19.5.5, "How to Configure an Error Event to Catch Business Exceptions".

   For information on how to configure the implementation properties to catch system exceptions, see Section 19.5.6, "How to Configure a Catch Event to Catch System Exceptions".

19.5.2 What Happens When You Handle an Exception Using a Boundary Catch Event

If the BPMN Service Engine encounters an error while running a task that has a boundary error catch event attached, then it follows the flow defined by the boundary error catch event. The exception handling flow defined by the boundary error catch event can re-join the main process flow or end the process.

19.5.3 How to Handle an Exception Using an Event Subprocess

You can use an event subprocess to handle an exception that can occur while running any of the flow objects in your BPMN process.

To handle an exception using an event subprocess:

1. From the Component Palette, from the Activities section, select **Event Subprocess**.

2. Drop the event subprocess in the process.

3. Right-click the start event of the event subprocess.

4. Select **Properties**.

5. Click the **Implementation** tab.

6. From the **Implementation Type** list, select **Error**.

7. Configure the implementation properties to catch a business or system exception.

For information on how to configure the implementation properties to catch business exceptions, see Section 19.5.5, "How to Configure an Error Event to Catch Business Exceptions".

For information on how to configure the implementation properties to catch system exceptions, see Section 19.5.6, "How to Configure a Catch Event to Catch System Exceptions".

19.5.4 What Happens When You Handle an Exception Using an Event Subprocess

If the exception handled in the event subprocess occurs while running any of the tasks in the process, then the BPMN Service Engine continues running the exception handling flow defined in the event subprocess.

19.5.5 How to Configure an Error Event to Catch Business Exceptions

You can configure an error event to catch business exceptions. To configure an error event to catch business exceptions you must edit the error event implementation properties.

To configure the implementation properties of an error event to catch business exceptions:

1. If you want to handle all the business exceptions that can occur while running this process, then select **Catch All Business Exceptions**.

   If you want to catch a specific business exception:

   a. Click the **Browse** button next to the **Exception** field.

      The Type dialog box appears.

   b. Enter the name of the exception or select it from the tree.

   c. Click **OK**.

      The Type dialog box closes and the selected exception appears in the **Exception** field.

19.5.6 How to Configure a Catch Event to Catch System Exceptions

You can configure an error event to catch system exceptions. To configure an error event to catch system exceptions you must edit the error event implementation properties.

To configure the implementation properties of an error event to catch system exceptions:

1. If you want to handle all the system exceptions that can occur while running this process, then select **Catch All System Exceptions**.

   If you want to catch a specific business exception:

   a. Click the **Browse** button next to the **Exception** field.

      The Type dialog box appears.

   b. Select **Show System Faults**.

      The tree shows the available system faults. For a list of the supported exception for the different error events, see Table 19–1.

   c. Enter the name of the exception or select it from the tree.

d. Click **OK**.

The Type dialog box closes and the selected exception appears in the **Exception** field.

19.6 Throwing Exceptions in Subprocesses or Reusable Processes

You can only throw business exceptions using an error end event, thus only parent processes can catch these exceptions.

You can configure your process to throw custom high level exceptions instead of throwing the low-level exceptions that occur while running the task. To throw a high level exception, connect the boundary events in your activities to the end event that throws the error, or finish the subprocess event with an error end event.

19.6.1 How to Throw an Exception

You can use an error end event to configure your BPMN process to throw a business exception.

To throw an exception:

1. If you want to throw a custom exception, create a business exception.

   You can also throw existing business exceptions or system exceptions.

   See Section 19.6.3, "How to Create a Business Exception" for more information on how to create a business exception.
2. Identify the point in your process where you want to throw the exception.
3. Branch the flow of the process using one of these options:
   - Add a gateway to create a branch in the flow of the process.
   - Add a boundary event.
4. From the Component Palette, drag **Error End Event** and drop it in the process.
5. Add a sequence flow to link the gateway or boundary event, and the error end event.
6. Right-click the error end event.
7. Select **Properties**.
8. Click the **Implementation** tab.
9. Click the **Browse** button next to the **Exception** field.

   The Type dialog box appears.
10. If you want to throw a system exception, Select **Show System Faults**.

    The tree shows the available system faults. For a list of the supported exception for the different error events, see Table 19–1.
11. Enter the name of the exception or select it from the tree
12. Click **OK**.

    The Type dialog box closes and the selected exception appears in the Exception field.
13. Click **OK**.

19.6.2 What Happens When You Throw an Exception

The BPMN Service Engine interrupts the process and throws the exception to the parent process. If the subprocess has an error catch boundary event attached or the parent process has an event subprocess that can handle the error event, then the parent process can handle the exception. Otherwise the parent process throws the exception to its parent process. If it does not have a parent process, then the BPMN Service Engine logs the exception to the Oracle Enterprise Manager fault handling system.

19.6.3 How to Create a Business Exception

You can create a business exception and use it to implement the error events in your BPMN process.

To create a business exception:

1. Right-click a module in the Business Catalog.

   If the business catalog does not contain a module, then you must create one.

2. Select **New**.
3. Select **Exception**.
4. Enter a name to identify the exception.
5. Click **OK**.

   The Business Exception Editor opens.

6. Optionally, you can change the `errorInfo` attribute.

   See Section 19.6.5, "How to Configure the ErrorInfo Attribute in a Business Exception" for information on how to modify the ErrorInfo attribute.

19.6.4 What Happens When You Create a Business Exception

The exception appears in the business catalog in the module you selected. You can configure an error end event in your process to throw this exception, or you can configure a boundary error catch event to handle this exception.

19.6.5 How to Configure the ErrorInfo Attribute in a Business Exception

Business exceptions contain an errorInfo attribute that you can use to store relevant information about the situation that caused the exception. You can use the information in this field to help users, process developers, and administrators understand the cause of the error.

To configure the ErrorInfo attribute in a business exception:

1. Open the **Business Exception Editor**.
2. In the Attributes section, expand the errorInfo attribute.
3. Make changes to the errorInfo attribute properties.

   You can modify the following properties:

   - Description
   - Documentation
   - Type

4. Click **Save** or close the **Business Exception Editor**, and save the changes.

19.7 Handling Exceptions in Subprocesses

You can handle exceptions that occur in a subprocess in the same way you handle the exceptions in any other BPMN activity.

19.8 Handling Errors in a Peer Process Using Message Events

When a process communicates with another peer process, running any of the flow objects in the peer process may result in an error. For synchronic operations, the correct form of propagating these errors to the invoking peer process is using message events configured as errors.

A message event configured as an error communicates to the invoking peer process that an error occurred while running the process. However the audit trail indicates that the process ran successfully because this is an expected error.

You must define how the invoking peer process handles the exception using one of these options:

- Add a boundary error catch event to the flow object that invokes the peer process.
- Add an event subprocess that handles the exception to the invoking peer process.

If you do not handle the error in the invoking peer process, the error propagates and the process running does not complete successfully.

Note: You must always define a path for the instance to follow if there are no error. If you do not define a path for the case where there are no errors the project does not build successfully.

Difference Between Using Error Events and Error Message Events

Using error end or throw events to handle errors during interprocess communication is not a good practice. You must only use error events for internal errors that might be handled within the process or propagated to the next level. These errors are not meaningful outside of this process.

The exceptions occurred while running a peer process do not propagate to the invoking peer process. Eventually the invoking peer process receives a time out notification because the peer process stopped responding.

19.8.1 How to Handle Errors in a Peer Process Using Message Events

If you know running an operation in a process that is used for inter-process communication may result in an error, it is advisable to add a message end or throw event to propagate the error to the invoking peer process.

The message error implementation requires you to select a business exception. If your project does not define business exceptions, then you must create a business exception. For more information about business exceptions, see Section 19.2, "Using Business Exceptions".

To handle errors in a peer process using message events:

1. Edit the peer BPMN process.
2. Add a message end or throw event.
3. Add a sequence flow from the flow object that can produce an error to the message end event.

4. Right-click the message end event.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. From the Implementation list, select **Exception**.
8. Click the **Browse** button next to the Exception field, to browse the available business exceptions and select one.
9. Click **OK**.

19.8.2 What Happens When You Handle Errors in a Peer Process Using Message Events

If there is an error in the invoked peer process, the error is communicated to the process that invoked it. The invoking peer process must handle the error using error events or the error is propagated to the next level.

After running the invoked peer process, its status appears as successfully ran because the error message event is part of the expected flow of the process.

20

Communicating With Other BPMN Processes and Services

This chapter describes how to develop a BPMN process that communicates with other BPMN processes and services. It shows you how to invoke other processes or services and how to broadcast a message to multiple process and how to configure your process to wait for a specific broadcast message.

This chapter includes the following sections:

20.1 Introduction to Communication with Other BPMN Processes and Services

Oracle BPM provides multiple ways for BPMN processes to communicate with other processes or services:

- **Messages**

They enable you to invoke asynchronous services or asynchronous BPMN processes. You can also use them to define the interface your process exposes to other processes or services.

See Section 20.2, "Communicating With Other BPMN Processes and Services Using Message Events", for more information about message events.

- **Send and Receive Tasks**

  They are very similar to message events. You can choose to use one or the other.

  The only difference they have with message events is that they support boundary events.

  They enable you to invoke asynchronous services or asynchronous BPMN processes. You can also use them to define the interface your process exposes to other processes or services.

  See Section 20.6, "Communicating With Other BPMN Processes and Services Using Send and Receive Tasks", for more information about send and receive tasks.

- **Signal Events**

  They enable you to broadcast a message to multiple process. The processes waiting for that specific message react to it.

  See Section 20.11, "Communicating Between Processes Using Signal Events", for more information about signal events.

20.1.1 Introduction to Synchronous and Asynchronous Operations

Message events, send and receive tasks, and service task use operations to communicate with other BPMN processes or services. These operations can be synchronous or asynchronous.

The main difference between a synchronous and an asynchronous operation is how they respond when you invoke them.

When you invoke a synchronous operation, you send a message and then wait for an response before proceeding with the process flow.

When you invoke an asynchronous operation, you send a message but do not wait for an answer to proceed with the process flow. The asynchronous operation receives the message and starts running. You can obtain the answer of an asynchronous operation by invoking a callback operation. If you invoke the callback operation before the asynchronous operation finishes running, then you must wait for it to complete before getting the answer.

Message events and send and receive task require you to specify how to associate an operation with its corresponding callback. Conversations allow you to group one or more operations with their callback. A conversation may define multiple operations that you can use to access a BPMN process.

20.2 Communicating With Other BPMN Processes and Services Using Message Events

Message events enable you to communicate with the other BPMN processes and services in your project.

You can use message events to:

- Invoke an asynchronous service.

- Invoke an asynchronous BPMN process.
- Define an interface for other processes to communicate with your process.

> **Note:** The send and receive tasks perform similar functionality to the throw and catch message events. However, it is recommended that you do not mix both within a single process.

The implementation of the different message events varies according to the type of event and their role in the conversation. Table 20–1describes the different implementation of message events.

Table 20–1 Message Event Implementation

Event	Initiates Conversation	Continues Conversation
Message Start	■ Define the interface of the operation ■ Use an interface from the business catalog	Not Available
Message Throw	■ Invoke a Service ■ Invoke a BPMN Process	If it continues a start event or a catch event that define an interface: ■ Define the callback interface for an asynchronous operation ■ Define the output for a synchronous operation ■ Define an exception for a synchronous operation If it continues a message throw that invokes a service or a BPMN process: ■ Invoke an operation from the same service or BPMN process it continues.
Message Catch	■ Define the interface of the operation ■ Use an interface from the business catalog	If it continues a start event or a catch event that define an interface: ■ Use the interface of the initiator event ■ Define the interface of the operation If it continues a throw event that invokes a service or a BPMN process: ■ Invoke the callback of the service or the BPMN process
Message End	Not Available	If it continues a start event or a catch event that define an interface: ■ Define callback for an asynchronous operation ■ Define the output for a synchronous operation ■ Define an exception for a synchronous operation If it continues a throw event that invokes a service or a BPMN process: ■ Invoke an operation from the same service or BPMN process it continues.

20.3 Using Message Events to Invoke Asynchronous Services and Asynchronous BPMN Processes

You can use message events to invoke asynchronous services and asynchronous BPMN processes.

To invoke an asynchronous operation from service or BPMN process you must use an intermediate throw message event configured to initiate a conversation.

When the BPMN Service Engine runs the message throw event, it creates an XML message based on:

- the asynchronous operation
- the input required by the asynchronous operation
- the data association defined for the message throw event

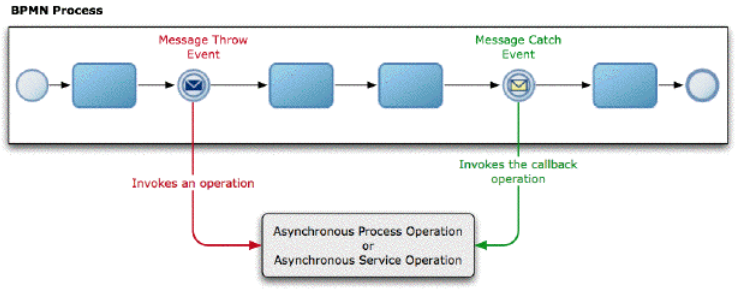
Then it sends the XML message to the service or BPMN process, and continues running the rest of the process flow. It does not wait for the asynchronous service or BPM process to answer.

The asynchronous service or BPMN process receives the message and runs the requested operation. When it finishes it sends a message with the result of the operation to the BPMN process that invoked it. This message is the callback operation of the asynchronous service or BPMN process.

The BPMN process that invoked the asynchronous operation must wait for the callback operation to obtain its results. The BPMN process must define a message catch event that waits for the callback operation. This message catch event continues the conversation and uses the message throw event that invoked the operation as the initiator event.

When a token arrives to the message catch event it might receive an immediate answer if the asynchronous process completed, or might have to wait until the asynchronous process completes to get an answer.

Figure 20–1 Invoking an Asynchronous Service or BPMN Process Using Message Events



This figure shows a BPMN process that invokes an asynchronous operation defined in an external service or in another BPMN process. This BPMN process uses message events to invoke the asynchronous operation.

****

20.3.1 How to Invoke Asynchronous Service Operation Using Message Events

You can invoke an asynchronous service operation using message events.

To invoke an asynchronous service operation using message events:

1. Edit the BPM process where you want to invoke the asynchronous service operation.
2. Locate the point in your process where you want to invoke the asynchronous service operation.
3. Add a message throw event in the point you located in your process.
4. Right-click the message throw event.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. In the Conversation section, select **Initiates**.
8. In the Properties section, select **Service Call** from the Implementation list.

   You must ensure that the service you select is an asynchronous service.
9. Click the **Browse** button next to the Name field.

   The Type dialog appears.
10. Select the asynchronous service you want to invoke.
11. Click **OK**.
12. From the Operation list, select the operation to invoke from the asynchronous service.
13. If the asynchronous service requires arguments, configure the message throw event data association.

    See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.
14. Click **OK**.
15. Follow the procedure described in Section 20.3.2, "How to Receive the Callback Operation of an Asynchronous Service Using Message Events" to invoke the callback operation of the asynchronous process.

20.3.2 How to Receive the Callback Operation of an Asynchronous Service Using Message Events

You can receive the callback operation that pairs with an asynchronous operation using message events.

To receive the callback operation of an asynchronous service using message events:

1. Edit the BPM process where you want to receive the callback of the asynchronous service.
2. Locate the point in your process where you want to receive the callback operation of the asynchronous service.
3. Add a message catch event in the point you located in your process.
4. Right-click the message catch event.
5. Select **Properties**.
6. Click the **Implementation** tab.

7. In the Conversation section, select **Continues**.

   The Properties section changes, and the Initiator Node list appears.

8. From the Initiator Node list, select the activity in your process that invokes the asynchronous service.

   The content of the Properties section changes, and the Name field and the Operation list appear. The Name field shows the asynchronous service that corresponds to the operation the initiator event invokes.

9. Click **OK**.

   The Type dialog disappears and the message catch event properties dialog shows the service you selected in the name field.

10. From the Operation list, select the callback operation to receive.

11. If the asynchronous service requires arguments, configure the message throw event data association.

    See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.

12. Click **OK**.

20.3.3 What Happens When You Invoke an Asynchronous Service Operation Using Message Events

When you invoke an asynchronous service operation using a message throw event, the BPMN Service Engine does not wait for the service to answer. It continues running the flow objects that follow to the message throw event.

The BPMN process can obtain the response of the asynchronous service by invoking the service callback operation using a message catch event.

Even if the service finishes running, the BPMN process does not receive the service response until it invokes the callback operation using a message catch event.

If the service is still running when the BPMN Service Engine runs the message catch event, then the engine waits for the service operation to complete before passing the token to the next flow object in the process.

20.3.4 How to Invoke an Asynchronous BPMN Process Operation Using Message Events

You can invoke a node in an asynchronous BPMN process using message events.

To invoke an asynchronous BPMN process operation using message events:

1. Edit the BPM process where you want to invoke the asynchronous BPMN process.
2. Locate the point in your process where you want to invoke the asynchronous BPMN process.
3. Add a message throw event in the point you located in your process.
4. Right-click the message throw event.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. In the Conversation section, select Initiates.

8. In the Properties section, select **Process Call** from the Implementation list.
9. Click the **Browse** button next to the Process field.

   The Type dialog appears.
10. Select the asynchronous BPMN process you want to invoke.
11. Click **OK**.
12. From the Node list, select the node from the asynchronous BPMN process.
13. If the asynchronous BPMN process requires arguments, configure the message throw event data association.

    See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.
14. Click **OK**.
15. Follow the procedure described in Section 20.3.5, "How to Invoke the Callback Operation of an Asynchronous BPMN Process Using Message Events" to invoke the callback operation of the asynchronous process.

20.3.5 How to Invoke the Callback Operation of an Asynchronous BPMN Process Using Message Events

You can invoke the callback operation that paris with an asynchronous node in a BPMN process using message events.

To invoke the callback operation of an asynchronous BPMN process using message events:

1. Edit the BPM process where you want to invoke the callback of the asynchronous BPMN process.
2. Locate the point in your process where you want to invoke the callback operation of the asynchronous BPMN process.
3. Add a message catch event in the point you located in your process.
4. Right-click the message catch event.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. In the Conversation section, select **Continues**.

   The Properties section changes, and the Initiator Node list appears.
8. From the Initiator Node list, select the activity in your process that invokes the asynchronous process.

   The content of the Properties section changes, and the Process field and the Node list appear. The Process field shows the name of the node that the initiator event invokes.
9. Click **OK**.

   The Type dialog disappears and the message catch event properties dialog shows the service you selected in the name field.
10. From the Node list, select the callback operation to invoke.
11. If the asynchronous BPMN process requires arguments, configure the message throw event data association.

See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.

12. Click **OK**.

20.3.6 What Happens When You Invoke an Asynchronous BPMN Process Using Message Events

When you invoke an asynchronous BPMN process using a message throw event, the BPMN Service Engine does not wait for the BPMN process to answer. It continues running the flow objects that follow to the message throw event.

The invoking BPMN process can obtain the response of the asynchronous BPMN process by invoking the service callback operation using a message catch event.

Even if the asynchronous BPMN process finishes running, the invoking BPMN process does not receive the response until it reaches a message catch event that receives a message from the asynchronous BPMN process.

If the asynchronous BPMN process is still running when the BPMN Service Engine runs the message catch event, then the engine waits for the asynchronous BPMN process to complete before passing the token to the next flow object in the process.

20.4 Using Message Events Configured as Boundary Events

You can use message catch events configured as boundary events to wait for an event while an activity is running. If the message arrives after the activity finishes running, then the event is not triggered.

You can configure a boundary message catch event as interrupting or non-interrupting.

Interrupting boundary message catch events stop running the activity when the expected message arrives. Then the engine starts running the flow defined for the message catch event. The flow defined for interrupting boundary message catch events may resume the main process flow.

Non-interrupting boundary catch events do not stop running the current activity. When the expected message arrives the engine starts running the flow defined for the message catch event in parallel to the current activity. The flow defined for non-interrupting boundary message catch events cannot resume the main process flow.

20.5 Using Service Tasks to Invoke Synchronous Operations in Services and BPMN Processes

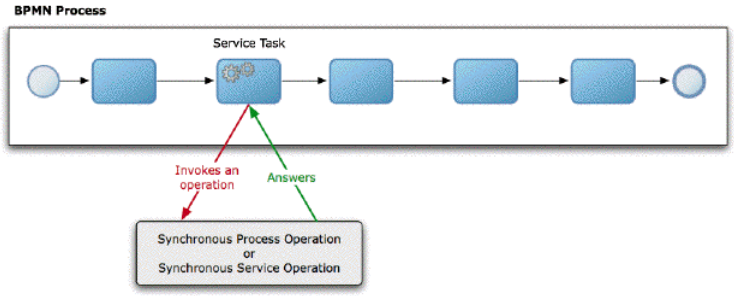
Service tasks enable you to invoke synchronous operations in services and BPMN processes.

When the BPMN Service Engine runs a service task, it invokes the operation specified in the service task and waits for a response. The BPMN Service Engine does not move the token to the next activity until it receives a response from the synchronous service or BPMN process.

The services you can use from a service task include BPEL processes, SOA mediators and SOA adapters that expose synchronous operations. You can also use service tasks to invoke other BPMN processes that expose synchronous operations.

See Section 21.4, "Using Message Events to Define a Synchronous Operation in a BPMN Processes Interface" or Section 21.8, "Using Send and Receive Tasks to Define a Synchronous Operation in a BPMN Process" for more information on how to define synchronous operations in a BPMN process.

Figure 20–2 Invoking a Synchronous BPMN Process or Service Using a Service Task



This figure shows a BPMN process that invokes a synchronous operation defined in an external service or in another BPMN process. This BPMN process uses message events to invoke the asynchronous operation.

****

20.5.1 How to Invoke a Synchronous Service Operation Using a Service Task

To invoke a synchronous service operation you must use a service task.

To invoke a synchronous service operation using a service task:

1. Edit the BPMN process.
2. Locate the point in your process where you want to invoke the synchronous service operation.
3. Add a service task in the point you located in your process.
4. Right-click the service task.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. In the Service Task section, select Service.
8. Click the **Browse** button next to the Name field.

   The Type dialog appears.
9. Select the synchronous service you want to invoke.
10. Click **OK**.
11. From the Operation list, select the operation from the synchronous service to invoke.
12. If the synchronous service requires input data or returns output data, then you must specify how the data objects in the project map to this data, by configuring the service task data association.

    See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.

13. Click **OK**.

20.5.2 What Happens When You Invoke a Synchronous Service Operation Using a Service Task

When the BPMN Service Engine runs a service task, it waits for the service to respond before continuing with the process flow. When the service finishes running, it sends the response to the service task.

If the service operation returns output data, then this data is mapped to the data objects in the project using the service task data association.

20.5.3 How to Invoke a Synchronous BPMN Process Operation Using a Service Task

You must invoke a synchronous BPMN process operation using a service task.

To invoke a synchronous BPMN process operation using a service task:

1. Edit the BPMN process.
2. Locate the point in your process where you want to invoke the synchronous BPMN process.
3. Add a service task in the point you located in your process.
4. Right-click the service task.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. In the Service Task section, select Process.
8. Click the **Browse** button next to the Process field.

   The Type dialog appears.
9. Select the synchronous service you want to invoke.
10. Click **OK**.
11. From the Node list, select the event or activity from the synchronous BPMN process to invoke.
12. If the synchronous BPMN process requires input data or returns output data, then you must specify how the data objects in the project map to this data, by configuring the service task data association.

    See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.
13. Click **OK**.

20.5.4 What Happens When You Invoke a Synchronous BPMN Process Operation Using a Service Task

When the BPMN Service Engine runs a service task, it waits for the synchronous BPMN process to respond before continuing with the process flow. When the synchronous BPMN process finishes running, it sends the response to the service task.

If the synchronous BPMN process returns output data, then this data is mapped to the data objects in the project using the service task data association.

20.6 Communicating With Other BPMN Processes and Services Using Send and Receive Tasks

Send and receive tasks enable you to communicate with the other BPMN processes and services in your project.

The only difference between message events and send and receive tasks is that you can add boundary events to the latter. If you are invoking an asynchronous service and you want to add a deadline using a timer event configured as boundary, then you must use a send and a receive task instead of using message events.

You can use send and receive tasks to:

- Invoke an asynchronous service.
- Invoke an asynchronous BPMN process.
- Define an interface for other processes to communicate with your process.

To use a receive task to define the start operation of a process, you must locate it after a none start event and configure it to create instances.

The implementation of the different message events varies according to the type of event and their role in the conversation. Table 20–1describes the different implementation of message events.

Note: The send and receive tasks perform similar functionality to the throw and catch message events. However, it is recommended that you do not mix both within a single process.

Table 20–2 Send and Receive Tasks Implementation

Task	Initiates Conversation	Continues Conversation
Send Task	■ Invoke a Service ■ Invoke a BPMN Process	If it continues a receive task that defines an interface: ■ Define the callback interface for an asynchronous operation ■ Define the output for a synchronous operation ■ Define an exception for a synchronous operation
Receive Task	■ Define the interface of the operation ■ Use an interface from the business catalog	If it continues a receive task that defines an interface: ■ Use the interface of the receive task it continues ■ Define the interface of the operation If it continues a sent task that invokes a service or a BPMN process: ■ Invoke the callback of the service or the BPMN process

20.7 Using Send and Receive Tasks to Invoke Asynchronous Services and Asynchronous BPMN Processes

You can use send and receive tasks to invoke asynchronous operations in services and BPMN processes.

To invoke an asynchronous operation from service or BPMN process you must use a send task configured to initiate a conversation.

When the BPMN Service Engine runs the send task, it creates an XML message based on:

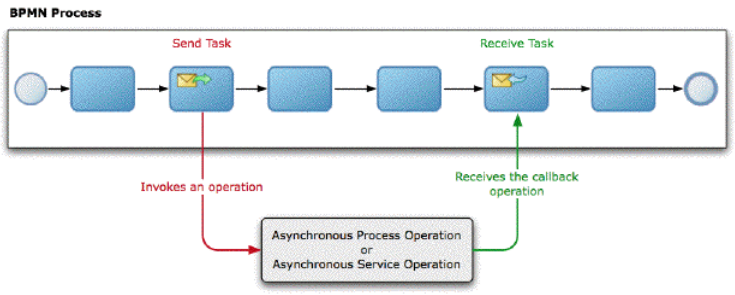
- the asynchronous operation
- the input required by the asynchronous operation
- the data association defined for the message throw event

Then it sends the XML message to the service or BPMN process, and continues running the rest of the process flow. It does not wait for the asynchronous service or BPM process to answer.

The asynchronous service or BPMN process receives the message and runs the requested operation. When it finishes it sends a message with the result of the operation to the BPMN process that invoked it. This message is the callback operation of the asynchronous service or BPMN process.

The BPMN process that invoked the asynchronous operation must invoke the callback operation to obtain its results. When it invokes the callback operation it might receive and immediate answer if the asynchronous process completed or might have to wait until the asynchronous process completes to get an answer.

Figure 20–3 Invoking an asynchronous service or BPMN process using send and receive tasks



This figure shows a BPMN process that invokes an asynchronous operation defined in an external service or in another BPMN process. This BPMN process uses send and receive task to invoke the asynchronous operation

****

20.7.1 How to Use a Send Task to Invoke an Asynchronous Service Operation

You can invoke an asynchronous service operation using a send task.

To invoke an asynchronous service operation using the send task:

1. Edit the BPM process where you want to invoke the asynchronous service.
2. Locate the point in your process where you want to invoke the asynchronous service.
3. Add a send task in the point you located in your process.
4. Right-click the send task.
5. Select **Properties**.

6. Click the **Implementation** tab.
7. In the Conversation section, select **Initiates**.
8. In the Properties section, select Service Call from the Implementation list.
9. Click the **Browse** button next to the Name field.

   The Type dialog appears.
10. Select the asynchronous service you want to invoke.
11. Click **OK**.
12. From the Operation list, select the operation from the asynchronous service to invoke.
13. If the asynchronous service requires input data, then you must specify how the data objects in the project map to this input data, by configuring the send task data association.

    See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.
14. Click **OK**.
15. Follow the procedure described in Section 20.7.2, "How to Use the Receive Task to Invoke the Callback Operation of an Asynchronous Service" to invoke the callback operation of the asynchronous process.

20.7.2 How to Use the Receive Task to Invoke the Callback Operation of an Asynchronous Service

You can invoke the callback operation that pairs with an asynchronous service operation using a receive task.

To invoke the callback operation of an asynchronous service:

1. Edit the BPM process where you want to invoke the callback of the asynchronous service.
2. Locate the point in your process where you want to invoke the callback operation of the asynchronous service.
3. Add a receive task in the point you located in your process.
4. Right-click the receive task.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. In the Conversation section, select **Continues**.

   The Properties section changes, and the Initiator Node list appears.
8. From the Initiator Node list, select the activity in your process that invokes the asynchronous service.

   The content of the Properties section changes, and the Name field and the Operation list appear.
9. Click the **Browse** button next to the Name field to select the asynchronous process.

   The Type dialog appears.
10. Select the asynchronous service whose callback you want to invoke.

11. Click **OK**.

    The Type dialog disappears and the receive task properties dialog shows the service you selected in the Name field.

12. From the Operation list, select the callback operation to invoke.

13. If the callback operation requires input data, then you must specify how the data objects in the project map to this input data, by configuring the receive task data association.

    See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.

14. Click **OK**.

20.7.3 What Happens When You Invoke an Asynchronous Service Using Send and Receive Tasks

When you invoke an asynchronous service operation using a send task, the BPMN Service Engine does not wait for the service to answer. It continues running the flow objects that follow to the send task.

The BPMN process can obtain the response of the asynchronous service by invoking the service callback operation using a receive task.

Even if the service finishes running, the BPMN process does not receive the service response until it invokes the callback operation using a receive task

If the service is still running when the BPMN Service Engine runs the receive task, then the engine waits for the service operation to complete before passing the token to the next flow object in the process.

20.7.4 How to Use the Send Task to Invoke an Asynchronous BPMN Process Operation

You can use a send task to invoke an asynchronous BPMN process operation.

To invoke an asynchronous BPMN process operation:

1. Edit the BPM process where you want to invoke the asynchronous BPMN process.
2. Locate the point in your process where you want to invoke the asynchronous BPMN process.
3. Add a send task in the point you located in your process.
4. Right-click the send task.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. In the Conversation section, select **Initiates**.
8. In the Properties section, select **Process Call** from the Implementation List.
9. Click the **Browse** button next to the Process field.

   The Type dialog appears.

10. Select the asynchronous BPMN process you want to invoke.
11. Click **OK**.
12. From the Node list, select the operation from the asynchronous BPMN process.

13. If the asynchronous BPMN process requires input data, then you must specify how the data objects in the project map to this input data, by configuring the send task data association.

    See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.

14. Click **OK**.

15. Follow the procedure described in Section 20.7.5, "How to Use a Receive Task to Invoke the Callback Operation of an Asynchronous BPMN Process" to invoke the callback operation of the asynchronous process.

20.7.5 How to Use a Receive Task to Invoke the Callback Operation of an Asynchronous BPMN Process

You can use a receive task to invoke the callback operation that pairs with an asynchronous process operation.

To invoke the callback operation of an asynchronous BPMN process:

1. Edit the BPM process where you want to invoke the callback of the asynchronous BPMN process.
2. Locate the point in your process where you want to invoke the callback operation of the asynchronous BPMN process.
3. Add a receive task in the point you located in your process.
4. Right-click the receive task.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. In the Conversation section, select **Continues**.

   The Properties section changes, and the Initiator Node list appears.

8. From the Initiator Node list, select the activity in your process that invokes the asynchronous process.

   The content of the Properties section changes, and the Process field and the Node list appear.

9. Click the **Browse** button next to the Process field to select the asynchronous process.

   The Type dialog appears.

10. Select the asynchronous BPMN process whose callback you want to invoke.
11. Click **OK**.

    The Type dialog disappears and the receive task properties dialog shows the service you selected in the name field.

12. From the Node list, select the callback operation to invoke.
13. If the asynchronous callback operation requires input data, then you must specify how the data objects in the process map to this input data, by configuring the receive task data association.

    See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.

14. Click **OK**.

20.7.6 What Happens When You Invoke an Asynchronous BPMN Process Using Send and Receive Tasks

When you invoke an asynchronous service operation using a send task, the BPMN Service Engine does not wait for the service to answer. It continues running the flow objects that follow to the send task.

The BPMN process can obtain the response of the asynchronous service by invoking the service callback operation using a receive task.

Even if the service finishes running, the BPMN process does not receive the service response until it invokes the callback operation using a receive task.

If the service is still running when the BPMN Service Engine runs the receive task, then the engine waits for the service operation to complete before passing the token to the next flow object in the process.

20.8 Introduction to Invoking a Process Using Call Activities

You can invoke a process from another process using call activities. The invoked process is a child of the process invoking it.

When you run a call activity, the engine does not create a new token for the reusable process. The token in the parent process passes to the reusable process. When the token completes the child process, it returns to the parent process to continue running the activities that follow the call activity.

The child process must be a reusable process. Reusable processes can be invoked from multiple processes. You can only start a reusable process by invoking it from a call activity.

You cannot access reusable process from other SOA components because they are not part of the SOA composite.

The start event of a reusable process must always be of type none. The end event can be a error or a message event.

20.9 Invoking a Process Using Call Activities

You can use call activities to invoke a process from another process. The child process must be a reusable process. You can invoke a reusable process from multiple processes within your BPM project.

20.9.1 How to Invoke a Process Using Call Activities

You can invoke a process from another process using call activities. The invoked process must be a reusable process.

To invoke a process using call activities:

1. Add a call activity to your process.
2. Right-click the call activity.
3. Select **Properties**.
4. Click the **Implementation** tab.

5. From the **Process** list, select a reusable process.

   For information on how to create a reusable process, see Section 5.1.2, "How to Create a New Business Process".

6. If necessary, configure data associations or transformations.

   For more information on data associations, see Section 8.13, "Introduction to Data Associations".

   For more information on transformations, see Section 8.14, "Introduction to Transformations".

7. Click **OK**.

20.10 Introduction to Communication Between Processes Using Signal Events

Signal events allow you to broadcast a message to all the processes in a BPM project. Only the processes configured to listen to that signal react.

In the Sales Quote example you might want to trigger a signal when a quote gets approved to trigger all the process that depend on the approval of a quote.

Mediators and BPEL processes also react when a BPMN process broadcasts a signal and they can also trigger a BPMN process by broadcasting a signal.

Oracle BPM uses Oracle Event Delivery Network (EDN) to send and receive signals. For more information about Oracle EDN see "Using Business Events and the Event Delivery Network" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

For information on how to access Event in Oracle BPM, see Chapter 12.1, "Introduction to the Business Catalog".

The EDN events your SOA project defines automatically appear in the business catalog in the Events predefined module Events. When you add a signal event you can choose which of the events in the business catalog the signal event broadcasts or reacts to.

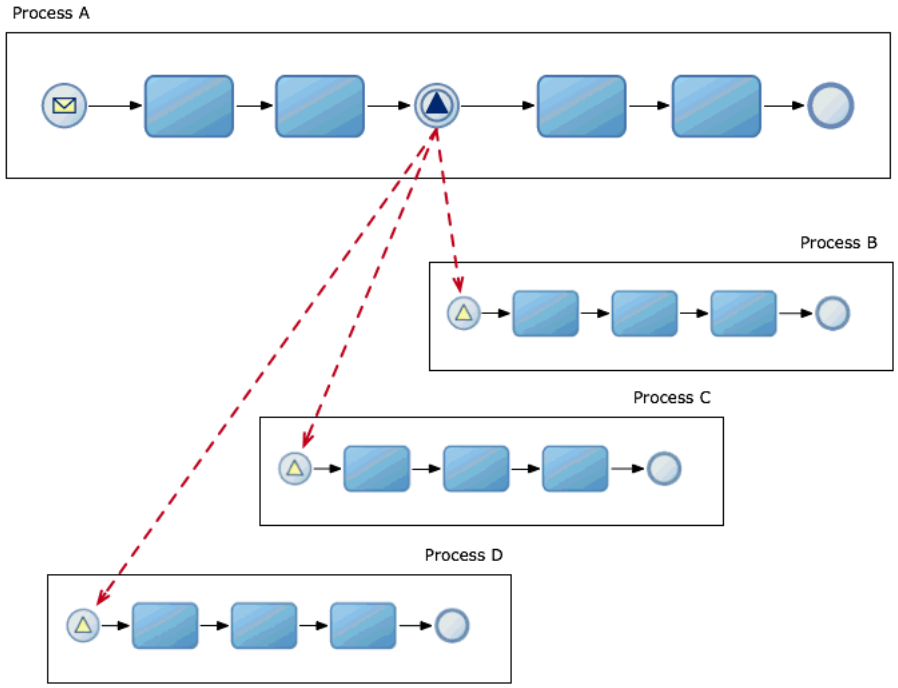
You can broadcast a signal from a throw intermediate signal event or from a signal end event. In a BPMN process you can only receive a signal in a a signal start event in another process.

The process that broadcasts the message has no information about the receivers. You might add or remove processes that react to a signal without impacting the process that broadcasts the signal.

In a similar way, the process that reacts to a specific message has no information about the processes that broadcast that message. If you add a process that broadcast a message to your project, all the process waiting for that specific message react to it without you having to modify them.

The events you use to broadcast a signal contain a payload that you can use to send information to all the processes configured to react to this specific signal. To assign values to the payload in the event you must configure the signal throw event data association. This data association enables you to pass the relevant data stored in the process and project data objects to the event. When the corresponding processes receive the signal, they must obtain the data in the event using another data association. This data association defines which data objects store the data in the event received in the signal start event.

Figure 20–4 Signal Broadcast



This figure shows a process A that broadcasts a signal. Processes B, C and D are configured to react to that signal.

****

20.11 Communicating Between Processes Using Signal Events

You can use signal events to communicate a message to all the processes that are configured to wait for that message.

20.11.1 How to Broadcast a Signal to Multiple Processes

Before following this procedure you must add the events you want to broadcast, to your SOA project.

To broadcast a signal to multiple processes:

1. Locate the point in your process where you want to broadcast the signal.
2. From the Component Palette, from the Throw Events section, select **Signal**.

   If you want to broadcast the signal immediately after the process finished, change the implementation type of the existing end event to signal or add new end event of type signal.
3. Drop the signal event in your process.
4. Right-click the signal event.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. Click the **Browse** button next to the event field.

   The Type dialog appears.

8. Select an event.
9. Click **OK**.

   The Type dialog disappears and the type name appears in the type field.
10. Click **OK**.

20.11.2 What Happens When You Broadcast a Signal

When the BPMN Engine runs a throw or an end signal event, it published an event to Oracle EDN. Oracle EDN delivers this event to all SOA components configured to listen to that specific signal.

20.11.3 How to Configure Your Process React to a Specific Signal

Before following this procedure you must add the events you want to react to, to your SOA project.

To configure your process to react to a specific signal:

1. Change the implementation type of the process start event to signal, or add a new signal start event to your process.
2. Right-click the start event.
3. Select **Properties**.
4. Click the **Implementation** tab.
5. Click the **Browse** button next to the event field.

   The Type dialog appears.
6. Select an event.
7. Click **OK**.

   The Type dialog disappears and the type name appears in the type field.
8. Click **OK**.

20.11.4 What Happens When You Configure a Process To React to a Specific Signal

The process does not start until another BPMN process or SOA component broadcasts a specific signal. When a BPMN process or an SOA component broadcasts this signal using Oracle EDN, the process gets triggered by this signal.

21

Defining the Process Interface

This chapter describes how to configure a BPMN process to expose it as a service for other processes or services to invoke it. Oracle BPM enables you to expose the flow objects in the BPMN process as process operations. Other BPMN processes and services can invoke these operations.

This chapter includes the following sections:

This chapter assumes that you are familiar with SOA Composites. For more information about SOA Composites see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

21.1 Defining the Process Interface

The process interface is a group of operations a BPMN process exposes for other processes or services to use. The SOA Composite shows the BPMN process interface in the Exposed Services section.

You must define an interface for your BPMN process if you want other processes and services to use it. The interface you define contains the operations other processes and services can invoke.

Synchronous process operations define input and output arguments.

When you define an asynchronous processes operation you must also define its corresponding callback operation. The asynchronous operation defines the input arguments and the callback operation defines the output arguments.

You can define the process interface by defining operations in your BPMN Process or you can choose to use an existing interface from the business catalog. You can implement any of these options using message events or send and receive tasks.

21.2 Using Message Events to Define the BPMN Process Interface

The process interface contains the operations that other services and processes can invoke to interact with a BPMN process. These operations may be synchronous or asynchronous.

You can define the process interface using message events or send and receive tasks.

To expose an operation in a BPMN process you can use a message start or message catch event configured as initiators. These message events enable you to define if the operation is synchronous or asynchronous. They also enable you to define the process input.

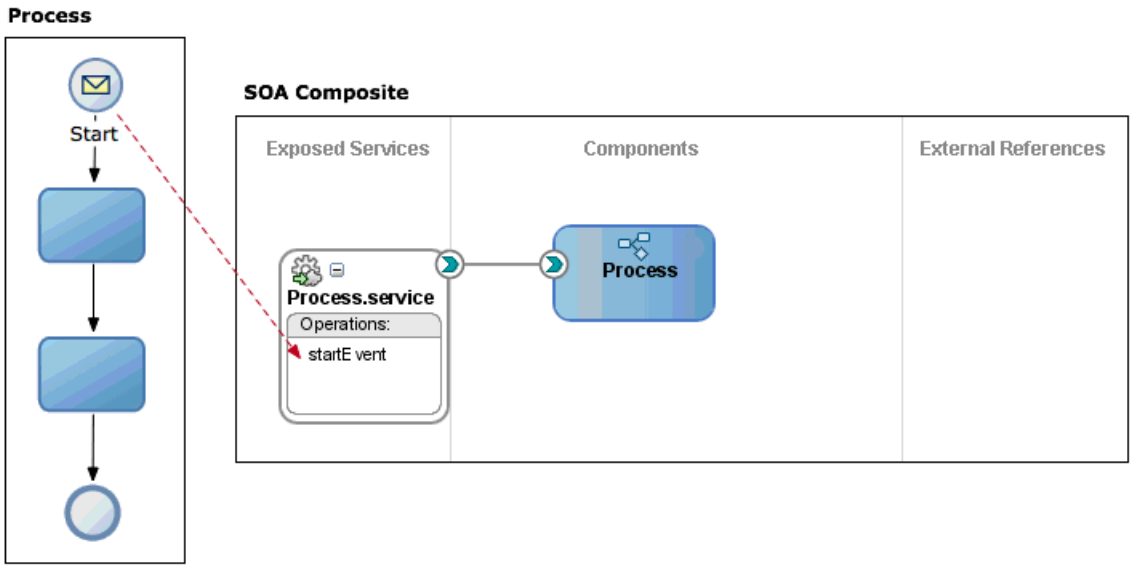
The process interface must always contain an operation that exposes the start event of a BPMN process. A process or service that invokes this BPMN process must always invoke the operation that corresponds to the start event before invoking any of the operations in the process.

To define the process output, you must configure the message throw or message end event that continue the event that defines the operation. If the operation is asynchronous, then these events also define the callback operation.

If an interface contains an asynchronous operation, then it must also define the callback operation that returns the result of this operation. See Section 21.2.1, "Using Message Events to Define the Callback Interface for BPMN Processes" for more information on how to define a callback operation in a BPMN Process.

Figure 21–1 shows a BPMN process that exposes a message start message event in its interface. It also shows how the SOA Composite editor displays this operation.

Figure 21–1 BPMN Process that exposes a message start event as an operation



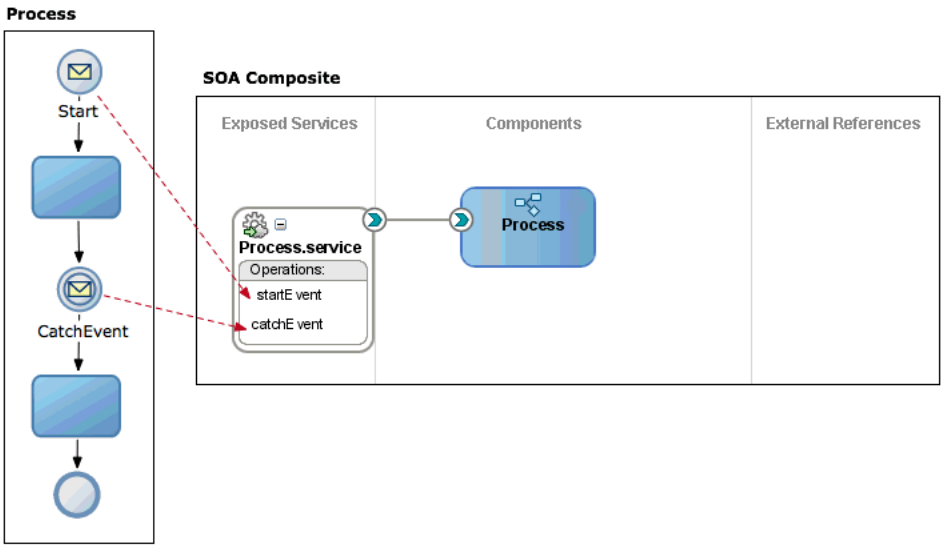
This figure shows a BPMN process that exposes a message start event as an operation, and how this reflects in the SOA Composite editor.

****

In addition, the process interface may contain the operations exposed by the catch message events in the process. Before invoking an operation that corresponds to a catch message event, you must always invoke the operation that corresponds to the message start event.

Figure 21–2 shows a BPMN process that exposes a catch message event in its interface in addition to the message start message event. It also shows how the SOA Composite editor displays this operation.

Figure 21–2 BPMN process that exposes a message start and a message catch event in its interface



This figure shows a BPMN process that exposes a message start event and a message catch event as operation, and how this reflects in the SOA Composite editor.

****

21.2.1 Using Message Events to Define the Callback Interface for BPMN Processes

A BPMN process must expose a callback operation for each of the asynchronous operations it defines.

The callback operation returns the response to the service or process that invoked the asynchronous operation. The callback operation may define output arguments. If it defines output arguments you must map their values to the data objects in the process using data associations.

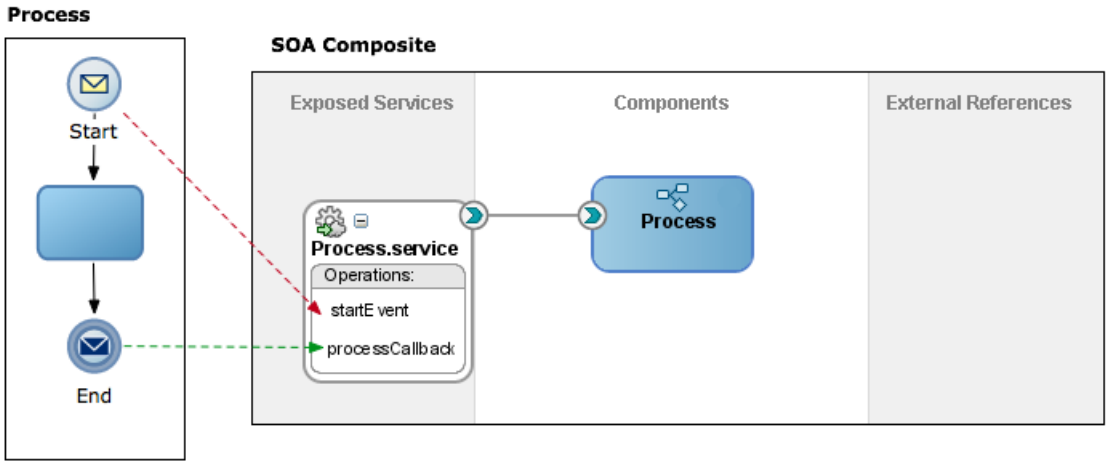
You can define a callback operation using a message throw event or a message end event.

See Chapter 20.3, "Using Message Events to Invoke Asynchronous Services and Asynchronous BPMN Processes", for information on how invoke an asynchronous BPMN process from another BPMN process.

Figure 21–3 shows an end event that exposes the BPMN process callback operation. It also shows how the callback operation appears in the SOA Composite editor.

Note: If you used a send task to expose an operation, then you must use a receive task to define the callback operation. See Section 21.7, "Defining Asynchronous Processes Operations Using Send and Receive Tasks" for more information on how to define a callback operation using send events.

Figure 21–3 Asynchronous BPMN process that exposes a start operation an its corresponding callback



This figure shows a BPMN process that exposes a start operation and its corresponding callback, and how this reflects in the SOA Composite editor.

****

21.3 Using Message Events to Define Asynchronous Operations in a BPMN Processes

You can define asynchronous operations in a BPMN Process using message events. If you expose an asynchronous operation, then you must also expose a start operation. The client invoking the asynchronous service must invoke the start operation first to create an instance in the process. The asynchronous operation runs over the created instance.

You must also specify a callback operation for each of the asynchronous operations you define.

21.3.1 How to Configure the Start Operation of a BPMN Process as Asynchronous Using Message Events

You can expose the start event of a BPMN process as an asynchronous operation.

To configure the start operation of a BPMN process as asynchronous:

1. Edit the BPMN process.
2. Right-click the start activity.
3. Select **Properties**.
4. Click the **Implementation** tab.
5. If the Implementation Type is not message, then change it to Message.

   The Conversation section appears.
6. In the Conversation Properties section, select **Define Interface** from the Implementation list.

7. If your asynchronous BPMN process requires input data, then you must define the process input in the Arguments Definition section.

   For more information on how to define the process input see Section 21.10, "Defining the Process Input and Output".

8. Expand the **Advanced** section.
9. Select **Asynchronous**.
10. Enter a name for the start operation.

    The SOA Composite uses the name you specify for the operation to display it in the SOA Composite.

11. Click **OK**.
12. Follow the procedure described in Section 21.3.2, "How to Define a Callback Operation Using Message Events", to define the callback operation of the asynchronous BPMN process.

21.3.2 How to Define a Callback Operation Using Message Events

You can expose a callback operation that pairs with an asynchronous operation using message events.

To define the callback operation:

1. Edit the BPMN process.
2. Locate the point in your process where you want to return the answer of the corresponding operation.
3. To return the answer before the process finishes, then add an intermediate message throw event to your process.

   Note: To return the answer when the processes finishes, then add a message end event or change the implementation type of the end to message

4. Right-click the message throw event or the message end event.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. If you are editing a throw message event, in the Conversation section, select **Continues**. If you are editing an end message event this is the default selection and you cannot change it.
8. From the Initiator list, select the event to associate with the callback.
9. If you want your asynchronous process to return output data, then you must define the process output in the Argument Definition section.

   For more information on how to define the process output, see Section 21.10, "Defining the Process Input and Output".

10. Expand the **Advanced** section.
11. To change the name of the callback operation, then enter a name.

    The SOA Composite uses the name you specify for the operation to display it in the SOA Composite.

12. Click **OK**.

21.3.3 What Happens When You Configure a BPMN Process Start Operation as Asynchronous Using Message Events

When you invoke the process start event you must not wait for a response before continuing with the process flow. To obtain the response you must invoke the process callback operation.

You can invoke asynchronous BPMN processes using message events or send and receive tasks.

See Section 20.3, "Using Message Events to Invoke Asynchronous Services and Asynchronous BPMN Processes" and Section 20.7, "Using Send and Receive Tasks to Invoke Asynchronous Services and Asynchronous BPMN Processes", for more information on how to invoke an asynchronous BPMN process.

In the SOA Composite, the interface of an asynchronous process shows at least two operations: the operation to start the process and its callback operation.

21.3.4 How to Add an Asynchronous Operation to a BPMN Process Interface Using Intermediate Message Events

You can expose an intermediate message event as an asynchronous operation.

To add an asynchronous operation to a BPMN process interface:

1. Edit the BPMN process.
2. Locate the point in your process where you want to add the new operation.
3. Add an intermediate catch message event.
4. Right-click the catch message event.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. In the Conversation section, select **Initiates**.
8. In the Properties section, select **Define Interface** from the Implementation list.
9. If your want your operation to have output arguments, then define input arguments.

   For more information on how to define output arguments see Section 21.10, "Defining the Process Input and Output".
10. Expand the **Advanced** section.
11. Select **Asynchronous**.
12. To change the name of the callback operation, then enter a name in the Operation Name field.
13. Click **OK**.
14. Follow the procedure described in Section 21.3.2, "How to Define a Callback Operation Using Message Events", to define the callback operation for this asynchronous operation.

21.3.5 What Happens When You Add an Asynchronous Operation to a BPMN Process Interface Using Message Events

The asynchronous operation and the corresponding callback operation are available for other processes to invoke them.

The SOA Composite shows the asynchronous operation and its callback in the BPMN process interface.

21.4 Using Message Events to Define a Synchronous Operation in a BPMN Processes Interface

You can define a synchronous operation in your BPMN process using message events. You define the synchronous operation using a message start or catch event, and a message throw or catch that continue the first. The message start or catch event defines the process input. The message throw or end event defines the process output.

If you use a message catch event to define a synchronous operation, then you must also define a start operation. You must invoke the start operation before invoking the synchronous operation, to create an instance in the process. The synchronous operation runs over the created instance.

See Section 20.5, "Using Service Tasks to Invoke Synchronous Operations in Services and BPMN Processes", for more information on how to invoke a synchronous BPMN process.

Message events enable you to send error message when you define synchronous operations. For more information about error message events, see Section 19.8, "Handling Errors in a Peer Process Using Message Events".

21.4.1 How to Configure the Start Operation of a BPMN Process as Synchronous Using Message Events

You can expose the message start event of a BPMN process as a synchronous operation.

To configure the start operation of a BPMN process as synchronous using message events:

1. Edit the BPMN process.
2. Right-click the start activity.
3. Select **Properties**.
4. Click the **Implementation** tab.
5. If the Implementation Type is not message, then change it to Message.

   The Conversation section appears.
6. In the Conversation Properties section, select Define Interface from the Implementation list.
7. If your synchronous BPMN process requires input data, then you must define the process input in the Argument Definition section.

   For more information on how to define the process input, see Section 21.10, "Defining the Process Input and Output".
8. Expand the **Advanced** section.

9. Select **Synchronous**.
10. Enter a name for the start operation.

    The SOA Composite uses the name you specify for the operation to display it in the SOA Composite.
11. Click **OK**.
12. Configure the end event following the procedure described in Section 21.4.2, "How to Configure the End Event of a Synchronous Process".

> **Note:** When adding a synchronous start event, you must also add an end or catch message event that is part of the same conversation. The end or catch message event continue the start event thus they are also synchronous.

21.4.2 How to Configure the End Event of a Synchronous Process

When you expose a start event as a synchronous operation, you must configure the end event of the process as synchronous.

To Configure the end event of a synchronous process:

1. Right-click the end event.
2. Select **Properties**.
3. Click the **Implementation** tab.
4. If the Implementation Type is not message, then change it to Message.

   The Conversation section appears.
5. If there is no Initiator Node selected, then select the start event of your process from the Initiator Node list.

   The sub-section to define the interface appears in the Properties section.
6. If your synchronous BPMN process returns output data, then you must define the process output in the Argument Definition section.

   For more information on how to define the process output, see Section 21.10, "Defining the Process Input and Output".
7. If your synchronous BPMN process returns output data, then you must specify how the data objects in your project map to the process output.

   For more information on how to configure data associations, see Section 8.13, "Introduction to Data Associations".
8. Click **OK**.

21.4.3 What Happens When You Configure the Start Operation of a BPMN Process as Synchronous Using Message Events

The process start event exposes a synchronous operation. When you invoke the process start event from a client, you must wait for a response before continuing with the process flow. The service task that invokes the synchronous process waits for the synchronous process to finish before the token moves to the next activity in the process.

You must invoke synchronous operations in a BPMN processes using a service tasks.

See Section 20.5, "Using Service Tasks to Invoke Synchronous Operations in Services and BPMN Processes", for more information on how to invoke a synchronous BPMN process.

In the SOA Composite, the interface of a synchronous process only shows one operation for the start event.

21.5 Using Message Events with an Interface from the Business Catalog to Define Your Process Interface

When configuring the message events that define the interface of your process, you can choose to use an existing interface instead of defining an interface.

You can choose any of the operations from the *References* predefined module in the business catalog and use it as the interface for your process operations.

The operation from the reference that you choose to define the interface of your operation, determines if your operation is synchronous or asynchronous.

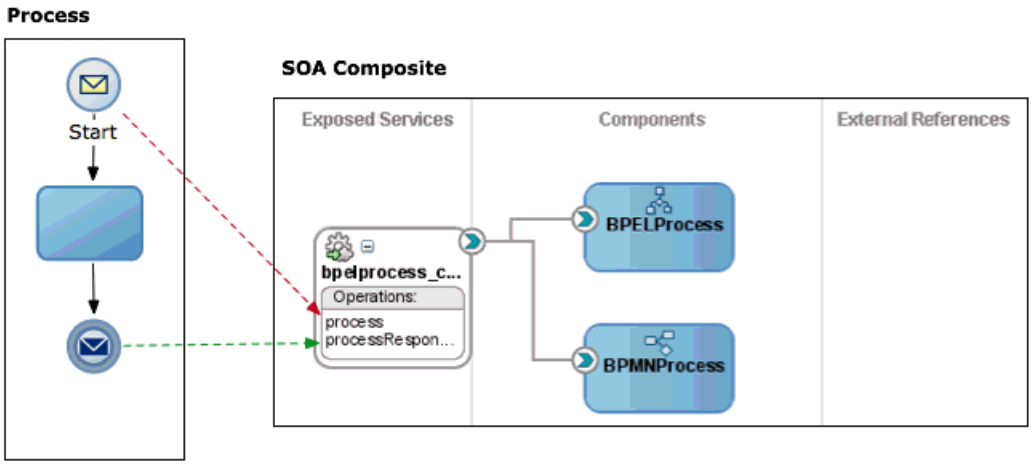
If you define a message start or a message catch event using an interface from the business catalog, then the associated message throw or message end event must also use an interface from the business catalog. If the operation you are defining is asynchronous, then the message throw or message end events can only use callback operations.

Generally you define the process interface using an interface from the business catalog to use a interface that exists in the composite and later on add a wire from this interface to the BPMN process.

You might provide multiple implementations of the same interface. For example you might implement an existing interface in BPEL and BPMN technologies. To implement the BPMN interface you must define the process using an interface from the business catalog.

Figure 21–4 shows how a BPMN process can reuse the interface of a BPEL process to provide a parallel implementation in BPMN. The BPMN process uses the interface of the BPEL process that appears in the business catalog to define its operations. It also shows how the SOA Composite editor indicates that a BPMN process uses another SOA Component to define its interface.

Figure 21–4 Process That Uses an Interface from the Business Catalog



This figure shows a BPMN process that uses an interface from the business catalog and how this reflects in the SOA Composite editor.

****

21.5.1 How to Use an Interface from the Business Catalog to Define an Operation in a BPMN Process Interface Using Message Start and Catch Events

You can use an interface from the business catalog to define the interface of your BPMN process.

To use an interface from the business catalog to define an operation:

1. Edit your BPMN process.
2. Add the start event or catch event to use to define the process interface.
3. Right-click the start or catch event.
4. Select **Properties**.
5. Click the **Implementation** tab.
6. If you are editing a catch message event, in the Conversation section, select Initiates. If you are editing a start event this is the default selection and you cannot change it.
7. In the Properties section, select **Interface from Catalog** from the Implementation list.

   The Properties section changes and the Name and Operation appear.
8. Click the **Browse** button next to the Name field.

   The Type dialog appears.
9. Select the reference you want to use as the process interface.
10. Click **OK**.
11. From the Operation list, select the operation you want to use as the process interface.

12. If the interface you selected requires input data, then you must specify how the data objects in the project map to this input data, by configuring the message event data association.

    For more information on how to configure data associations, see Section 8.13, "Introduction to Data Associations".

13. Click **OK**.

14. Configure an existing message end or message throw event to use an interface from the business catalog or add a new event and configure it, following the procedure described in Section 21.5.2, "How to Configure a Message End or a Message Throw Event to Use an Interface from the Business Catalog Using Message Events".

21.5.2 How to Configure a Message End or a Message Throw Event to Use an Interface from the Business Catalog Using Message Events

You can use an interface from the business catalog to define the interface of your BPMN process.

To configure a message end or message throw event to use an interface from the business catalog:

1. Edit the BPMN process.
2. Right-click the message end or message throw event.
3. Select **Properties**.
4. Click the **Implementation** tab.
5. In the Conversation section, select **Continues**.

   The Properties section changes, the Initiator Node, Name and Operation fields appear.

6. From the Initiator Node list, select the message start or message catch event that defines the process interface.
7. Click the **Browse** button next to the Name field.

   The Type dialog appears.

8. Select the component you want to use as the message catch or message end interface.
9. Click **OK**.
10. From the Operation list, select the operation you want to use as the as the message catch or message end interface.
11. If the interface you selected requires output data, then you must specify how the data objects in the project map to this output data, by configuring the message event data association.

    See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.

12. Click OK.

21.5.3 What Happens When You Use an Interface from the Business Catalog to Define an Operation

The operation you define uses the signature of the operation form the interface in the business catalog. To invoke the operation in the BPMN process you must use the same operation name and input that you use to invoke the operation in the interface from the business catalog. The operation in the BPMN process returns the same output that the operation in the interface from the business catalog.

The SOA composite shows a wire between the BPMN process and the interface used to define its operations.

If you define all the process operations using interfaces from the business catalog, then JDeveloper asks you if it should delete the BPMN process WSDL. Because the BPMN process does not define an interface, but uses existing interfaces, its WSDL is no longer necessary and you can delete it.

21.6 Defining the BPMN Process Interface Using Send and Receive Tasks

The process interface contains the operations that other services and processes can invoke to run a BPMN process. These operations may be synchronous or asynchronous.

You can define the process interface using message events or send and receive tasks.

See Section 21.2, "Using Message Events to Define the BPMN Process Interface", for more information on how to define the process interface using message events.

To expose an operation in a BPMN process you can use a receive task. The receive task enables you to define if the operation is synchronous or asynchronous. It also enables you to define the process input.

The process interface must always contain an operation that exposes a receive task that creates an instance. A process or service that invokes this BPMN process must always invoke this operation before invoking any of the operations in the process.

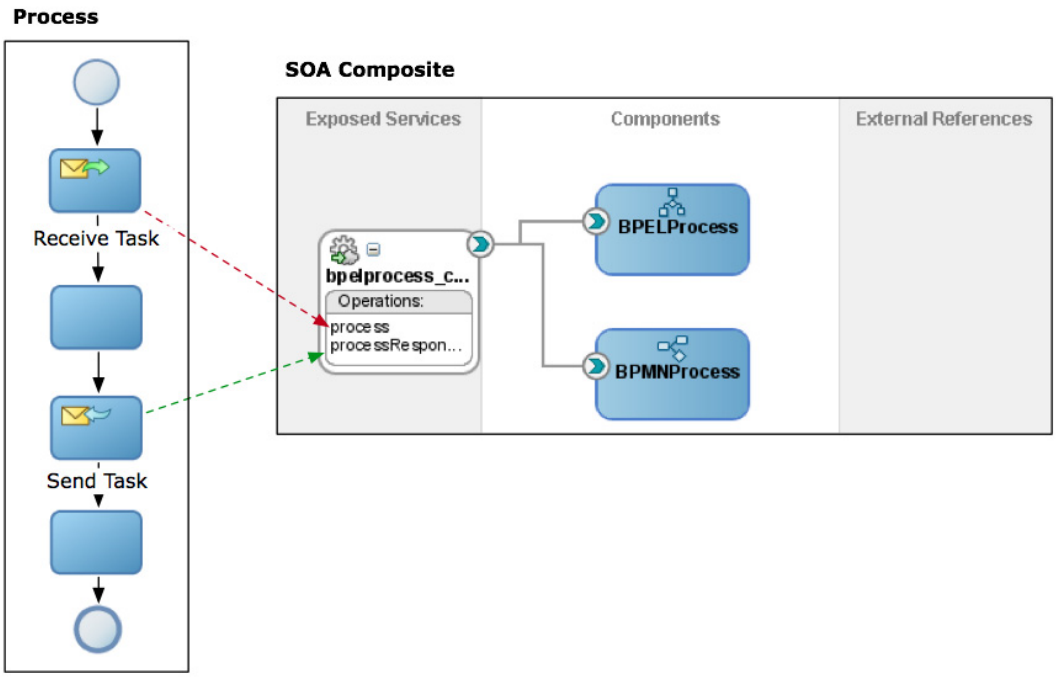
To define the process output, you must configure the send task that continues the receive that defines the operation. If the operation is asynchronous, then the send task also defines the callback operation.

If an interface contains an asynchronous operation, then it must also define the callback operation that returns the result of this operation. See Section 21.6.1, "Defining the Callback Interface for BPMN Processes Using a Send Task" for more information on how to define a callback operation in a BPMN Process.

In addition, the process interface may contain the operations exposed by the receive tasks in the process. Before invoking an operation that corresponds to a receive task, you must always invoke the operation that corresponds to the received task configured to create an instance.

Figure 21–5 shows a BPMN process that exposes a receive task in its interface in addition to the receive tasks that creates the instance. It also shows how the SOA Composite editor displays these operations.

Figure 21–5 BPMN process that exposes an asynchronous operation defined using send and a receive task



This figure shows a BPMN process that exposes an asynchronous operation defined using send and receive tasks, and how this reflects in the SOA Composite editor.

****

If you used a send task to expose an operation, then you must use a receive task to define the callback operation. See Section 21.6.1, "Defining the Callback Interface for BPMN Processes Using a Send Task" for more information on how to define a callback operation using send events.

21.6.1 Defining the Callback Interface for BPMN Processes Using a Send Task

A BPMN process must expose a callback operation for each of the asynchronous operations it defines. You can define a callback operation using a send task.

The callback operation returns the response to the service or process that invoked the asynchronous operation. If the service or process is waiting for the answer, then they receive it immediately. If the service or process is not waiting for the answer yet, then they receive it when they get to the part of the process or code that waits for the answer.

The callback operation may define output arguments. If it defines output arguments you must map their values to the data objects in the process using data associations.

Figure 21–5 shows a receive task that exposes the BPMN process callback operation.

21.7 Defining Asynchronous Processes Operations Using Send and Receive Tasks

You can define asynchronous operations in a BPMN Process using send and receive tasks. If you expose an asynchronous operation, then you must expose a start operation. The process invoking the asynchronous service must invoke the start operation first to create an instance in the process. The asynchronous operation runs over the created instance.

You must also specify a callback operation for each of the asynchronous operations you define.

21.7.1 How to Define an Asynchronous Process Operation Using Send and Receive Tasks

You can define an asynchronous process operation using send and receive tasks.

To define an asynchronous process operation using send and receive tasks:

1. Edit the BPMN process.
2. Change the trigger of the start and end events to None:
   a. Right-click the event.
   b. Select **Properties**.
   c. Click the **Implementation** tab.
   d. From the Implementation Type list, select **None**.
   e. Click **OK**.
3. Add a receive task immediately after the start event.
4. Right-click the receive task.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. Select **Create Instance**.
8. In the Conversation section, select **Initiates**.
9. In the Conversation Properties section, select **Define Interface**.
10. If your asynchronous BPMN process requires input data, then you must define the process input in the Argument Definition section.

    For more information on how to define the process output, see Section 21.10, "Defining the Process Input and Output".
11. Expand the **Advanced** section.
12. Select **Asynchronous**.
13. Enter a name for the start operation.

    The SOA Composite uses the name you specify for the operation to display it in the SOA Composite.
14. Click **OK**.

15. Follow the procedure described in Section 21.7.3, "How to Define a Callback Process Operation Using a Send Task", to define the callback operation of the asynchronous BPMN process.

21.7.2 How to Add an Asynchronous Process Operation to the Process Interface Using a Receive Task

You can expose a receive task as an asynchronous process operation.

To add an asynchronous process operation using a receive task:

1. Edit the BPMN process.
2. Locate the point in your process where you want to add the new operation.
3. Add a receive task in the point you located.
4. Right-click the receive task.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. In the Conversation section, select **Initiates**.
8. In the Conversation Properties section, select **Define Interface**.
9. If your asynchronous BPMN process requires input data, then you must define the process input in the Argument Definition section.

   For more information on how to define the process output, see Section 21.10, "Defining the Process Input and Output".
10. Expand the **Advanced** section.
11. Select **Asynchronous**.
12. Enter a name for the start operation.

    The SOA Composite uses the name you specify for the operation to display it in the SOA Composite.
13. Click **OK**.

21.7.3 How to Define a Callback Process Operation Using a Send Task

You can expose a send task as the callback operation that pairs with an asynchronous process operation.

How to define the callback operation for an asynchronous process using a send task:

1. Edit the BPMN process.
2. Locate the point in your process where you want to return the answer of the corresponding operation.
3. Add a send task to the point you located in your process.

   You must place the send task after the receive task in the process flow.
4. Right-click the send task.
5. Select **Properties**.
6. Click the **Implementation** tab.

7. In the Conversation section, select **Continues**.
8. From the Initiator list, select the receive task to associate with the callback.
9. If you want your asynchronous process to return output data, then you must define the process output in the Argument Definition section.

   For more information on how to define the process output, see Section 21.10, "Defining the Process Input and Output".
10. Expand the **Advanced** section.
11. Select **Synchronous** or **Asynchronous**.
12. To change the name of the start operation, then enter a name.

    The SOA Composite uses the name you specify for the operation to display it in the SOA Composite.
13. Click **OK**.

21.7.4 What Happens When You Define an Asynchronous Operation Using Send and Receive Tasks

The asynchronous operation and the corresponding callback operation are available for other processes to invoke them.

When you invoke the process asynchronous operation you defined, you must not wait for a response before continuing with the process flow. To obtain the response you must invoke the process callback operation.

The SOA Composite shows the asynchronous operation and its callback in the BPMN process interface.

You can invoke asynchronous BPMN processes using message events or send and receive tasks.

See Section 20.3, "Using Message Events to Invoke Asynchronous Services and Asynchronous BPMN Processes" and Section 20.7, "Using Send and Receive Tasks to Invoke Asynchronous Services and Asynchronous BPMN Processes", for more information on how to invoke an asynchronous BPMN process.

21.8 Using Send and Receive Tasks to Define a Synchronous Operation in a BPMN Process

You can define a synchronous operation in your BPMN process using send and receive tasks. You define the synchronous operation using a receive task and send tasks that continues the receive task. The receive task defines the process input and the send task defines the process output.

If you use a send task to define a synchronous operation, then you must also define a start operation. You must invoke the start operation before invoking the synchronous operation, to create an instance in the process. The synchronous operation runs over the created instance.

See Chapter 20.5, "Using Service Tasks to Invoke Synchronous Operations in Services and BPMN Processes", for information on how to invoke a synchronous operation in a BPMN process from another BPMN process.

21.8.1 How to Configure a Process Operation as Synchronous Using Send and Receive Tasks

You can expose send and receive tasks as a synchronous process operation.

To configure a process operation as synchronous:

1. Edit the BPMN process.
2. Change the trigger of the start and end events to None:
   a. Right-click the event.
   b. Select **Properties**.
   c. Click the **Implementation** tab.
   d. From the Implementation Type list, select **None**.
   e. Click **OK**.
3. Add a receive task after the start event.
4. Right-click the receive task.
5. Select **Properties**.
6. Click the **Implementation** tab.
7. Select **Create Instance**.
8. In the Conversation Properties section, select **Define Interface**.
9. If your synchronous BPMN process requires input data, then you must define the process input in the Argument Definition section.

   For more information on how to define the process input, see Section 21.10, "Defining the Process Input and Output".
10. Expand the **Advanced** section.
11. Select **Synchronous**.
12. Enter a name for the start operation.

    The SOA Composite uses the name you specify for the operation to display it in the SOA Composite.
13. Click **OK**.

21.8.2 What Happens When You Define a Synchronous Operation Using Send and Receive Tasks

The asynchronous operation and the corresponding callback operation are available for other processes to invoke them.

You must invoke synchronous operations in a BPMN processes using a service tasks.

See Section 20.5, "Using Service Tasks to Invoke Synchronous Operations in Services and BPMN Processes", for more information on how to invoke a synchronous BPMN process.

In the SOA Composite, the interface of a synchronous process only shows one operation for the receive task.

21.9 Using Send and Receive Tasks with an Interface from the Business Catalog to Define Your Process Interface

When configuring the receive tasks that define the interface of your process, you can choose to use an existing interface instead of defining an interface.

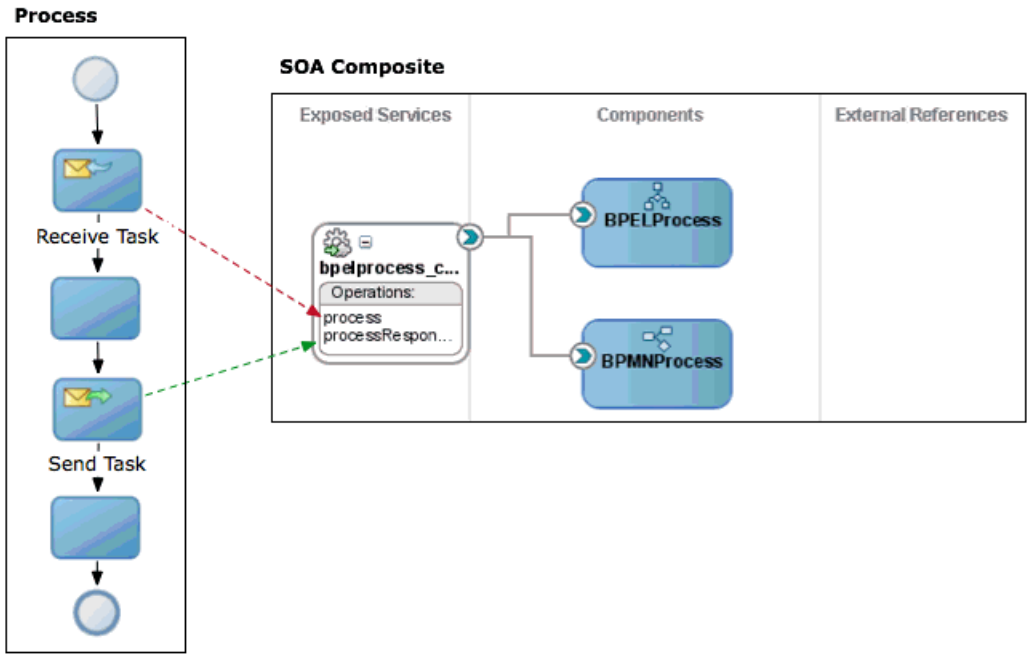
You can choose any of the operations from the components in the business catalog and use it as the interface for your process operations.

The operation from the component in the business catalog that you choose to define the interface of your operation, determines if your operation is synchronous or asynchronous.

If you define a receive task using an interface from the business catalog, then the associated send task must also use an interface from the business catalog. If the operation you are defining is asynchronous, then the message send task can only use callback operations.

Figure 21–6 shows a process that uses a BPEL process from the business catalog to define its operations. It also shows how the SOA Composite editor indicates that a BPMN process uses another SOA Component to define its interface.

Figure 21–6 BPMN Process that uses an interface from the Business Catalog defined using send and receive tasks



This figure shows a BPMN process defined using send and receive task that uses an interface from the business catalog. It also show this reflects in the SOA Composite editor.

****

21.9.1 How to Use an Interface from the Business Catalog to Define an Operation in a BPMN Process Interface Using Send and Receive Tasks

You can use an interface from the business catalog to define your BPMN process interface.

To use an interface from the business catalog to define an operation:

1. Edit your BPMN process.
2. Add the start event or catch event to use to define the process interface.
3. Right-click the start or catch event.
4. Select **Properties**.
5. Click the **Implementation** tab.
6. If you are editing a catch message event, in the Conversation section, select Continues. If you are editing a start event this is the default selection and you cannot change it.
7. In the Properties section, select **Interface from Catalog**.

   The Properties section changes and the Name and Operation appear.
8. Click the Browse button next to the Name field.

   The Type dialog appears.
9. Select the component you want to use as the process interface.
10. Click **OK**.
11. From the Operation list, select the operation you want to use as the process interface.
12. If the interface you selected requires input data, then you must specify how the data objects in the project map to this input data, by configuring the message event data association.

    For more information on how to configure data associations, see Section 8.13, "Introduction to Data Associations".
13. Click **OK**.
14. Configure an existing message end or message throw event to use an interface from the business catalog or add a new event and configure it, following the procedure described in Section 21.5.2, "How to Configure a Message End or a Message Throw Event to Use an Interface from the Business Catalog Using Message Events".

21.9.2 How to Configure a Message End or a Message Throw Event to Use an Interface from the Business Catalog Using Send and Receive Tasks

You can use an interface from the business catalog to define your BPMN process interface.

To configure a message end or message throw event to use an interface from the business catalog:

1. Edit the BPMN process.
2. Right-click the message end or message throw event.

3. Select **Properties**.
4. Click the **Implementation** tab.
5. In the Conversation section, select **Continues**.

   The Properties section changes, the Initiator Node, Name and Operation fields appear.
6. From the Initiator Node list, select the message start or message catch event that defines the process interface.
7. Click the **Browse** button next to the Name field.

   The Type dialog appears.
8. Select the component you want to use as the message catch or message end interface.
9. Click **OK**.
10. From the Operation list, select the operation you want to use as the as the message catch or message end interface.
11. If the interface you selected requires input data, then you must specify how the data objects in the project map to this input data, by configuring the message event data association.

    See Section 8.13, "Introduction to Data Associations", for more information on how to configure data associations.
12. Click **OK**.

21.9.3 What Happens When You Use Send and Receive Tasks with an Interface from the Business Catalog to Define an Operation

The operation you define uses the signature of the operation form the interface in the business catalog. To invoke the operation in the BPMN process you must use the same operation name and input that you use to invoke the operation in the interface from the business catalog. The operation in the BPMN process returns the same output that the operation in the interface from the business catalog.

The SOA composite shows a wire between the BPMN process and the interface used to define its operations.

If you define all the process operations using interfaces from the business catalog, then JDeveloper asks if it should delete the BPMN process WSDL. Because the BPMN process does not define an interface, but uses existing interfaces, its WSDL is no longer necessary and you can delete it.

21.10 Defining the 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.

21.10.1 How to Add Input and Output Arguments to a BPMN Process

When you expose operations using message start and end events, or send and receive tasks, you can define the input and output argument they require.

To add input and output arguments to a BPMN process:

1. In the Argument Definition section, click the Add button.

   The Create Argument dialog appears.

2. Enter a name to identify the argument.
3. Click the **Browse More Types** Button.

   The Browse Type dialog appears.

4. From the Type list, select a basic data type or select **<Component>** to use a complex data type.
5. If you selected <Component> then select a component from the list of available complex data types.
6. Click **OK**.

   The Browse Type dialog disappears and the data type you selected appears in the Type field in the Create Argument Dialog.

7. Click **OK**.

   The argument appears in the Argument Definition table.

21.10.2 How to Edit the Input and Output Arguments of a BPMN Process

You can change the name and the types of the arguments of a BPMN Process.

To edit the input and output arguments of a BPMN process:

1. From the Argument Definition table, select an argument.
2. In the Argument Definition section, click the **Edit** button.

   The Edit Argument dialog appears.

3. Change the name of the type.
4. Click **OK**.

   The argument in the Argument Definition table shows the updated name and type.

21.10.3 How to Delete an Input or Output Argument of a BPMN Process

You can delete input and output arguments that you do not use or need.

To delete an input or output argument:

1. From the Argument Definition table, select an argument.
2. In the Argument Definition section, click the **Remove** button.

   The select argument is removed from the Argument Definition table.

22

Writing Expressions

This chapter describes how to write expressions and conditions for the BPMN elements that require them. Oracle BPM provides you with two different types of expressions editors that adjust to requirements of different users. This chapter describes the expression language used by each of these expression builders and the operations you can use in the expressions you write.

This chapter includes the following sections:

22.1 Introduction to Expressions in Oracle BPM

Some BPM elements require you to write a condition or an expression that defines their behavior. For example, you might want to control the flow of your process using a conditional sequence flow that ensures that all expenses above 500 dollars are approved by a manager.

Oracle BPM provides you two ways of writing these expressions and conditions:

- Using the Simple Expression Builder
- Using the XPATH expression builder

The Simple Expression Builder uses dot notation and its syntax is very similar to Java. The XPATH Expression Builder uses standard XPATH language.

After writing an expression in simple expression language you can convert it to XPath and vice versa. When you convert an expression from one language to another, the expression editor removes any operators and parenthesis that do not affect the meaning of the expression.

Oracle BPM uses expressions to configure the following BPMN elements:

- Conditional Sequence Flows
- Complex Gateways
- Timer Events
- Data Associations
- Loop Markers
- Multi-Instance Markers
- User Task Advanced Properties

The results of the expression vary according to the type of element you are configuring. Table 22–1 describes the expression required by each of the BPM elements.

Table 22–1 Expression Types

BPMN Element	Expression Type
Conditional Sequence Flow	Condition that when evaluated results in a boolean value.
Complex Gateway	Condition that when evaluated results in a boolean value.
Timer Event	Time Date: expression that when evaluated results in a DateTime value. Cycle: expression that when evaluated results in an Interval value.
Data Associations	Expression that when evaluated results in a value of the same type as the argument in the data association.
User Task Advanced Properties	Expression that when evaluated results in a String value.
Loop Marker	Condition that when evaluated results in a boolean value.
Multi-Instance Marker	Loop Cardinality: expression that when evaluated results in an Int value. Completion Condition: Condition that when evaluated results in a boolean value.

The configuration dialogs of the BPM elements that support expressions contain an embedded expression editor and a button to launch the expression builder. The latter is more suitable when you are working with long expressions. Both expression builders enable you to browse the available variables. The XPATH expression builder also enables you to browse the available functions.

22.2 Writing Conditions in Conditional Sequence Flows

To implement a conditional sequence flow you must provide a condition. When the token arrives to the conditional sequence flow, the BPMN Server Engine evaluates the

condition in the conditional sequence flow to determine which sequence flow the token should follow.

Generally the condition is based on the values of the project and process data object, but this is not a requirement. The condition must result in a boolean value when the compiler evaluates it. If you write a condition that does not result in a boolean value, then the Simple Expression Builder prompts an error.

22.2.1 How to Implement a Conditional Sequence Flow

You must define an expression to implement a conditional sequence flow.

To implement a conditional sequence flow:

1. Right-click the conditional sequence flow.
2. Select **Properties**.
3. Click the **Implementation** tab.
4. From the Type List, select **Condition**.
5. In the Expression section, select the type of expression builder to use to write your condition.
6. If your condition is simple, then you can write it in the provided text area.

   If you are working with complex conditions, then you can launch the expression builder by clicking the Launch Expression Builder button next to the text area.
7. Click **OK**.

22.3 Writing Expressions in Complex Gateways

To implement a complex gateway you must provide 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.

Generally the condition is based on the number of tokens that arrived to the complex gateway. For example you might want the gateway to release the tokens after two tokens arrive to the merge gateway.

Example 22–1 shows a condition that configures the gateway to release the tokens that arrived to it after two tokens arrive to the merge gateway.

Example 22–1 Condition in a Complex Gateway

```
activationCount >= 2
```

22.3.1 How to Implement a Complex Gateway

You must define an expression to implement a gateway.

To implement a complex gateway:

1. Right-click the conditional complex gateway.
2. Select **Properties**.
3. Click the **Implementation** tab.

4. In the Expression section, select the type of expression builder to use to write your condition.
5. If your condition is simple, then you can write it in the provided text area.

   If you are working with complex conditions, then you can launch the expression builder by clicking the Launch Expression Builder button next to the text area.
6. Click **OK**.

22.4 Writing Expressions in Timer Events

To implement a timer event you can choose to specify a date or an interval, or to write an expression that calculates the date or the interval.

Generally you use expressions in those cases where the date or the interval are not fixed.

The following examples show expressions that you can use in a timer event to express a date:

- `'now' + '30m'`
- `deadline - ’1day’`
- `arrivalDate.dateTime + '1h'`

The following examples show expressions that you can use in a timer event to express an interval:

- `waitToRetry.interval()`
- `period(deadline)`

22.4.1 How to Use an Expression in a Timer Event

You can use an expression to calculate a date or an interval in the implementation of a timer event.

To use an expression in a timer event:

1. Right-click the timer event.
2. Select **Properties**.
3. Click the **Implementation** tab.
4. Select **Use Expression**.
5. In the Expression section, select the type of expression builder to use to write your condition.
6. If your expression is simple, then you can write it in the provided text area.

   If you are working with complex expressions, then you can launch the expression builder by clicking the Launch Expression Builder button next to the text area. The Expression Builder where you can write the expression appears.
7. Click **OK**.

22.5 Writing Expressions in Data Associations

You can use expressions in data associations to modify the input and output values before associating them with the activity implementation arguments.

Generally you use expressions when there is a mismatch between the data objects and the activity implementation arguments. The following examples describe situations where you can use expressions in a data association:

- A mismatch between the value of the data object and the argument the service requires.

  For example, the service your activity invokes uses a different product ID than the one you use in the process. In this case you can use an expression to adapt the content of the product ID data object to the value your services require.

- A mismatch between the data type of the data object and the data type of the argument the service requires.

  For example, the service your activity invokes uses a String to store the state of the order and your service requires you to specify the state of the order with an Int value. In this case you use an expression that calculates the Int value that corresponds to the state the String specifies.

22.5.1 How to Use an Expression in a Data Association

You can use expressions in data associations to modify the values of the arguments or data objects before mapping them.

To use an expression in a data association:

1. Right-click the activity whose data association you want to modify.
2. Select **Properties**.
3. Click the **Implementation** tab.
4. In the Data Associations section, select **Use Associations**.
5. From the Type list select the type of data association.

   The type of expression you can use depends on the type of data association you select.

6. Click the **Edit** button next to the Type list.

   The Data Associations dialog appears.

7. Locate the input or output argument you want to modify using an expression.
8. If your expression is simple, then you can write it in the provided text area.

   If you are working with complex expressions, then you can launch the expression builder by clicking the Expression Builder button next to the input or output text area. The Expression Builder where you can write the expression appears.

9. Click **OK**.

22.6 Writing Conditions in Loop and Multi-Instance Markers in Subprocesses

You can configure subprocesses to run multiple times using loop and multi-instance markers.

To configure loop and multi-instance makers you must define expressions and conditions that specify how to repeat the subprocess.

Loop Markers

Loop markers enable you to run a subprocess multiple times based on condition. You can configure the loop marker to evaluate the condition before or after running the subprocess. You can also configure the loop marker to stop after a certain number of repetitions.

To configure a loop maker you must write a Loop Condition that determines if the BPMN Service Engine must continue to repeat the subprocess.

Multi-Instance Markers

Multi-Instance markers enable you to run a subprocess for each of the elements on a set of data. When the BPMN Service Engine runs a subprocess with a multi-instance loop marker it creates a set of instances, one for each element on the set of data. You can configure the multi-instance marker to process these instances in parallel or sequentially.

The following fields in a multi-instance loop marker require you to write an expression:

- **Loop Cardinality**

  This expression defines the number of tokens to create in the subprocess.

- **Completion Condition**

  This expression determines when to stop repeating the subprocess. The BPM Service Engine evaluates this condition every time a token completes the subprocess. If the condition evaluates to true, it considers the subprocess completed and the instance moves to the next flow object in the process.

22.6.1 How to Configure Loop Markers

You can configure a loop marker to run a subprocess multiple times.

To configure loop markers:

1. Right-click the subprocess.
2. Select **Properties**.
3. Click the **Loop Characteristics** tab.
4. Select **Loop**.
5. Specify the Loop Condition:
   1. Select the expression language.

      Possible options are Simple or XPath.

   2. In the text area below, write the condition that drives the loop.

      Optionally you can write the condition using the Expression Builder. To launch the Expression Builder click the **Expression Builder** button next to the text area.
6. Optionally, you can specify a maximum number of times for the loop to run:
   1. Select **Loop Maximum**.
   2. Specify a number.

7. Select **before** to evaluate the condition before running the flow object, or deselect it to evaluate the condition after running the flow object.
8. Click **OK**.

22.6.2 How to Configure Multi-Instance Markers

You can configure a multi-instance marker to run subprocess multiple times based on a set of data.

To configure multi-instance markers:

1. Right-click the subprocess.
2. Select **Properties**.
3. Click the **Loop Characteristics** tab.
4. Select **MultiInstance**.
5. Specify the Loop Cardinality:
    1. Select the expression language.

        Possible options are Simple or XPath.

    2. In the text area below, write the specifies the loop cardinality.

        Optionally you can write the condition using the Expression Builder. To launch the Expression Builder click the **Expression Builder** button next to the text area.
6. Optionally, you can specify the Completion Condition:
    1. Select the expression language.

        Possible options are Simple or XPath.

    2. In the text area below, write the condition that determines if the loop is completed.

        Optionally you can write the condition using the Expression Builder. To launch the Expression Builder click the **Expression Builder** button next to the text area.
7. Click the Browse button next to the Loop Data Output field, to specify the data output.

    You can select a data object or an attribute in a complex data object to pass to the subprocess. Generally the selected data object is a collection of items.
8. Click the Browse button next to the Loop Data Input field, to specify the data input.

    Select a data object or an attribute in a complex data object to assign the result of the subprocess.
9. Optionally, check the Is Sequential check box to specify that the each token must complete the subprocess before the next token starts to run the subprocess.
10. Click **OK**.

22.7 Writing Expressions and Conditions Using the Simple Expression Builder

The Simple Expression Builder contains a text area for you to type the expression and a list of variables that you can use.

The Simple Expression Builder supports the following features:

- Syntax Highlighting

  The Simple Expression Builder highlights the syntax in your expressions to make them easier to read and understand. It uses different colors for the different data type values.

- Automatic Code Completion

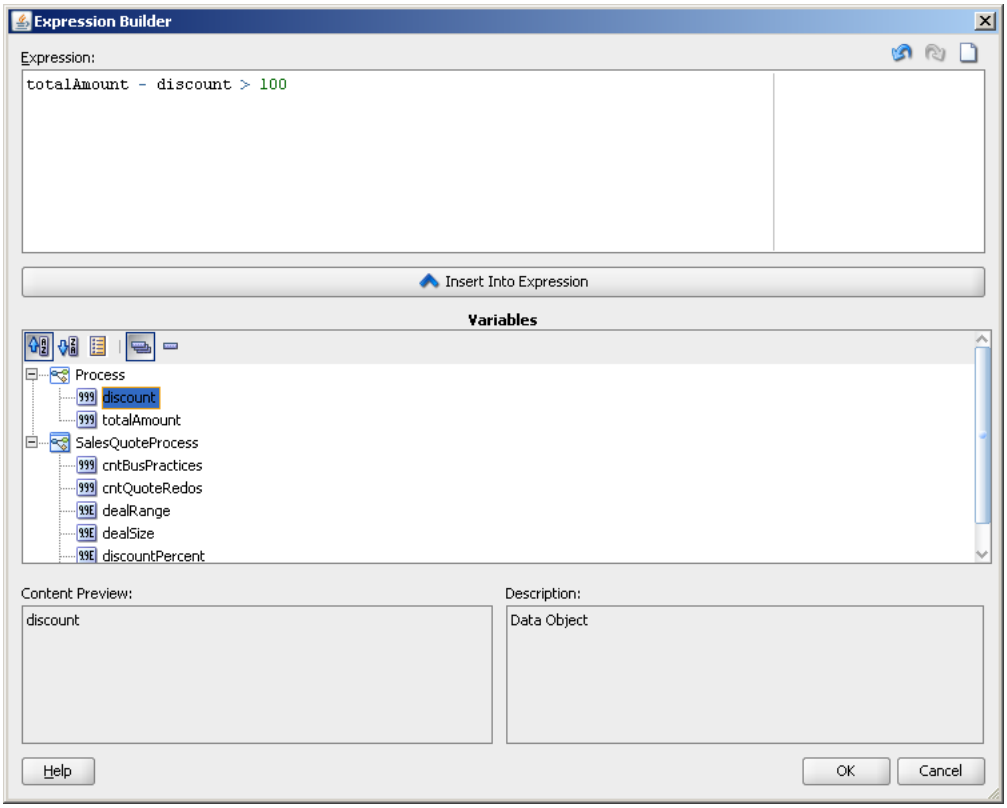
  If you wait a few seconds after you type the dot to invoke a method, then the Simple Expression Builder shows a list with the available functions that you can invoke over that data object. If you want the Simple Expression Builder to complete the expression for you, then you can press `Ctrl + Space`.

- On-the-fly Error Checking

  The Simple Expression Builder checks the expressions as you write. It underlines with a red waved line those expressions that do not compile. To find out the cause of the error place the cursor over the red wavy line and wait for a tooltip with the error description to appear.

Figure 22–1 show the Simple Expression Builder dialog.

Figure 22–1 Simple Expression Builder



22.7.1 How to Use a Data Object in an Expression

You can use a data objects in your expressions to perform calculations based on them.

To use a data object in an expression:

1. Open the **Simple Expression Builder**.
2. Place the cursor where you want to insert the data object.
3. From the variables section, select a data object.
4. Click **Insert Into Expression**.

   The selected data object appears in the Expression text area.

22.7.2 How to Use a Function in an Expression

To use a function in an expression, you can select the expression from the expression list in the simple expression builder, or you can type the function name in the Expression text area. If you write part of the name and press `Crtl+Space`, then the expression builder completes the name of the function.

To use a function in an expression using the simple expression builder:

1. Open the **Simple Expression Builder**.
2. Place the cursor where you want to insert the function.
3. From the **Functions** section, select a type of function.
4. From the **Functions** list, select a function.

   The Description field shows a description of the function.
5. Click **Insert Into Expression**.

   The selected function appears in the Expression text area.

22.8 Simple Expression Builder Supported Operators

The Simple Expression Builder enables you to create expressions using 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 drive 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`
- `deadlineExpired and orderStatus !=complete`
- `activationCount > 3`
- `unitsSold <= 1200`

- `'now' + '2m'`
- `deadline - '1h'`
- `not formComplete`

Table 22–1, Table 22–2, Table 22–3, Table 22–4 and Table 22–5 describe the supported operators in the Simple Expression Builder.

Table 22–2 Arithmetic Operators

Operator	Name	Description
+	Addition	Adds numeric data types. Concatenates Strings. Add an interval value to a DateTime value.
-	Subtraction	Subtracts numeric data types. Subtracts an interval value from a DateTime value.
*	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 22–3 Unary Operators

Operator	Name	Description
+	Plus	Has no effect on the value of the numeric operand. Use it to indicate explicitly that a certain value is positive.
-	Minus	Negates an arithmetic expression. Inverts the sign of a number.
not	Not	Logical complement operator. Negates the value of a boolean expression.

Table 22–4 Equality and Relational Operators

Operator	Name	Description
=	Equal	Returns true if the first operand equals the second operand.
!=	Not Equal	Returns true if the first operand is not equal to the second operand.
>	Greater Than	Returns true if the first operand is greater than the second operand.
>=	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 22–5 Conditional Operators

Operator	Name	Description
and	Conditional And	Returns true if both operands evaluate to true.

Table 22–5 (Cont.) Conditional Operators

Operator	Name	Description
or	Conditional Or	Returns true if one operand evaluates to true.

22.8.1 Operators Precedence

The precedence of the operators indicates the order in which the compiler evaluates them. You can change the precedence of the operators in an expression by using parenthesis.

The precedence of the operators in the Simple Expression Builder is:

- Unary Plus and Unary Minus
- Multiplication, Division, Remainder
- Addition and Subtraction
- Less than, Greater Than, Less Than or Equal to, Greater Than or Equal to
- Equal, Not Equal
- Not
- Conditional And
- Conditional Or

22.9 Simple Expression Builder Supported Functions

The Simple Expression Builder supports functions that you can use to calculate and manipulate your expressions and conditions.

The following sections describe the functions the Simple Expression Builder supports:

- Section 22.9.1, "String Functions"
- Section 22.9.2, "Numeric Functions"
- Section 22.9.3, "DateTime and Interval Functions"

22.9.1 String Functions

These functions enable you to manipulate String variables and literals, and perform calculations based on them.

22.9.1.1 length

Returns the number of characters in this String.

Signature:

```
Int length(String stringToMeasure)
```

Arguments:

-

Examples:

```
name.length()
```

```
length(name)
```

```
name.length
```

22.9.1.2 concatenation

Concatenates one or more Strings.

Examples:

```
name + " " + lastName
```

```
"Oracle " + "BPM"
```

22.9.1.3 contains

Returns true if the String contains the specified String.

Signature:

```
Bool contains(String mainString, String subString)
```

Arguments:

`subString` - The String to find.

Examples:

```
productName.contains("book")
```

```
contains(productName, "book")
```

22.9.1.4 startsWith

Returns true if the String starts with the specified String.

Signature:

```
Bool startsWith(String mainString, String subString)
```

Arguments:

`subString` - The String to find at the beginning of the String.

Example:

```
productId.startsWith("ABC")
```

```
startsWith(productId, "ABC")
```

22.9.2 Numeric Functions

These functions enable you to perform calculations using numeric data types. The available numeric data types are Real, Decimal, and Int.

22.9.2.1 floor

Returns the largest Int value that is smaller than the numeric value used for invoking this function. You can use this function with Real and Decimal data types.

Signature:

```
Int floor(Real number)
```

```
Int floor(Decimal number)
```

Arguments:

-

Examples:

```
number.floor()
floor(number)
number.floor
floor(totalAmount/3)
temperature.floor()
```

22.9.2.2 ceil

Returns the smallest Int value that is greater than the numeric value used for invoking this function. You can use this function with Real and Decimal data types.

Signature:

```
Int ceil(Real number)
Int ceil(Decimal number)
```

Arguments:

-

Examples:

```
number.ceil()
ceil(number)
number.ceil
```

22.9.2.3 round

Returns the closest Int value to this number. If there are two Int values that are equally close, then it returns the greater one. You can use this function with Real and Decimal data types.

Signature:

```
Int round(Real number)
Int round(Decimal number)
```

Arguments:

-

Examples:

```
number.round()
round(number)
number.round
```

22.9.2.4 abs

Returns the absolute value of this number. You can use this function with Int, Real, and Decimal data types.

Signature:

```
Int abs(Int number)
Real abs(Real number)
Decimal abs(Decimal number)
```

Arguments:

-

Examples:

```
number.abs()
abs(number)
number.abs
```

22.9.3 DateTime and Interval Functions

These functions enable you to manipulate time variables and literals, and perform calculations based on them. The available time data types are DateTime and Interval.

22.9.3.1 now

Special notation for the system current date and time.

Examples:

```
setReceivedDate('now')
```

22.9.3.2 addition

Adds an interval to a DateTime variable or value.

Examples:

```
today + '1d3h'
now + 3d
vacationStartingDate + '1M'
```

22.9.3.3 subtraction

Subtracts an interval to a DateTime variable or value.

Examples:

```
today - '2d3h25m'
now - age
expirationDate - '7d'
```

22.9.3.4 year

Returns the year of this DateTime variable.

Signature:

```
Int year(DateTime date)
```

Arguments:

-

Examples:

```
today.year()
year(today)
today.year
```

22.9.3.5 month

Returns the month of this DateTime variable.

Signature:

```
Int month(DateTime date)
```

Arguments:

-

Examples:

```
today.month()
month(today)
today.month
```

22.9.3.6 day

Returns the day of this DateTime variable.

Signature:

```
Int day(DateTime date)
```

Arguments:

-

Examples:

```
today.day()
day(today)
today.day
```

22.9.3.7 hours

Returns the hour of this DateTime variable.

Signature:

```
Int hours(DateTime date)
```

Arguments:

-

Examples:

```
today.hours()
hours(today)
today.hours
```

22.9.3.8 minutes

Returns the minutes of this DateTime variable.

Signature:

```
Int minutes(DateTime date)
```

Arguments:

-

Examples:

```
today.minutes()
minutes(today)
today.minutes
```

22.9.3.9 seconds

Returns the seconds of this DateTime variable.

Signature:

```
Int seconds(DateTime date)
```

Arguments:

-

Examples:

```
today.seconds()
seconds(today)
today.seconds
```

22.9.3.10 timezone

Returns an Interval value that represents the offset from UTC.

Signature:

```
Interval timezone()
```

Arguments:

-

Examples:

```
'1995-02-03 23:30:23-3:30'.timezone()
timezone('1995-02-03 23:30:23-3:30')
'1995-02-03 23:30:23-3:30'.timezone
```

The result of this example is '-3h30m'.

22.10 Writing Expressions Using the XPath Expression Builder

Oracle BPM enables you to write expressions using SOA XPath Expression Builder. This expression builder supports standard XPath language.

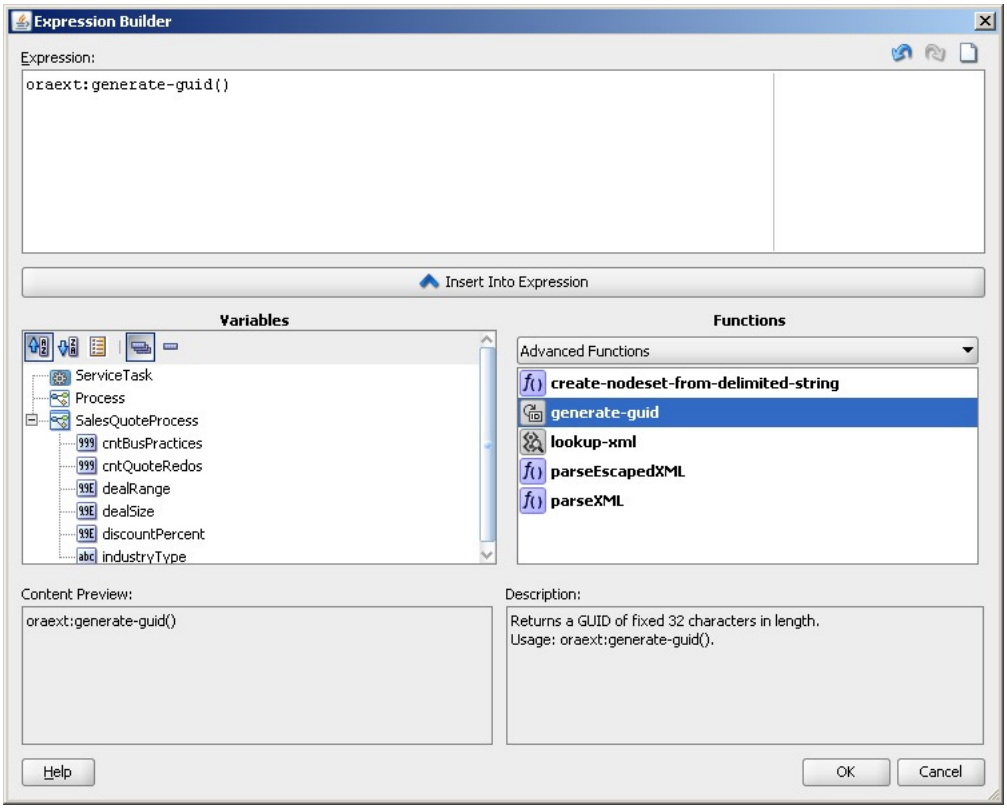
The XPath Expression Builder displays a list of the available variables that you can use in your expression. It also displays a list with the functions you can use in your expressions. When you select a function you can preview its syntax and description before adding it to your expression.

For more information about the functions supported for XPath see "Appendix B XPath Extension Functions" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Additionally Oracle BPM supports a group of BPM Extension Functions.

Figure 22–2 shows XPath Expression Builder.

Figure 22–2 XPath Expression Builder



22.10.1 How to Add a Variable to an XPath Expression

You can add variables to an XPath expression to perform calculations based on them.

To add a variable to an XPath expression:

1. Launch the **XPath Expression Builder**.
2. Place the cursor where you want to insert the variable.
3. From the variables list, select a variable.

4. Click **Insert Into Expression**.

   The selected variable appears in the expression text area.

22.10.2 How to Use a Function in an XPath Expression

You can invoke a function in an XPath expression. The XPath Expression Builder enables you to brow a list of functions grouped by functionality.

To use a function in an XPath expression:

1. Launch the **XPath Expression Builder**.
2. Place the cursor where you want to insert the expression.
3. From the Functions list select a function category.

   The list of available functions for the selected category appears below the Function list.
4. From the list of available functions, select a function.
5. Click **Insert Into Expression**.

   The selected function appears in the expression text area.

22.11 Using Arrays

Some service operations may require or return arguments of type array. These arrays can be collections simple types or complex types.

To provide a parameter of type array to a service operation, you can:

- Invoke an operation that returns an array
- Define an array using array literals

  For more information about defining array literals, see Section 22.12.4, "Using Array Literals".

Generally when a process operation returns an argument of type array you assign one of the elements of the array or an attribute of one of the elements to a data object using data association. For more information about data associations, see Section 8.13.1, "Introduction to the Data Association Editor".

Example 22–2 shows an expression that accesses an element of an array. In this example the name of the array is *persons* and the element to access is the third element in the array.

Example 22–2 Expression accessing an element of an array

```
persons[3]
```

Note that The index of the first element in the array is 1. For example to access the first element in the persons array you must write the following expression: `persons[1]`

22.11.1 Accessing an Attribute of an Element Within an Array

Simple expression language also enables you to access the attribute of the elements within the array. For example you could invoke a service operation that returns an array of persons and access the name attribute of the first element in the array to a process data object, using data associations.

Example 22–3 shows an expression that accesses an attribute of an element of an array. In this example the name of the array is *persons*, the element to access is the third element in the array and the attribute to access is the name of the person

Example 22–3 Expression accessing an attribute of an element of an array

```
persons[1].name
```

22.11.2 Obtaining the Length of an Array

Simple expression language enables you to obtain the length of an array to use it in your process logic. This is typical when defining the loop cardinality of multi-instance markers. For more information about multi-instance markers, see Section 22.6, "Writing Conditions in Loop and Multi-Instance Markers in Subprocesses".

To obtain the length of an array you must invoke the length function.

Example 22–3 shows an expression that obtains the length of an array. In this example the name of the array is *persons*. Note that even if the example uses parentheses after the function name but, you can omit them because this function does not require any parameters.

Example 22–4 Expression obtaining the length of an array

```
persons.length()
```

22.12 Using Literals

Literals enable you to express a certain value. You can use literals to assign a value to data object or to pass a parameter to a method you are invoking. To assign a value to a data object you can use data associations or a script task.

Simple expression language supports the following types of literals:

- String literals
- Time literals
- Interval literals
- Array literals

22.12.1 Using String Literals

You can assign a value to data object of type String using literals. You can also use String literals to provide a parameter of type String to a method.

To define a String literal you must write a word or a set of words enclosed by double quotes. A String can also contain numbers.

The following list shows examples of String literals:

- "Wednesday"
- "marie@oracle.com"
- "500 Oracle Parkway"
- "+1.650.506.7000"

22.12.2 Using Time Literals

You can assign a value to a data object of type Time using literals. You can also use time literals to provide a parameter of type time to a method.

Time literals enable you to define a date using different levels of precision.

The following list uses an example date to show the different time literals that you can use to specify a variable of type Time:

- `'13:30'`
- `'13:30:23'`
- `'13:30:23.001023'`
- `'13:30:23.001023Z'`
- `'13:30:23.001023-05'`
- `'13:30:23.001023-3:30'`
- `'1979-02-19'`
- `'1979-02-19 13:30'`
- `'1979-02-19 13:30:23'`
- `'1979-02-19 13:30:23.001023'`
- `'1979-02-19 13:30:23.001023Z'`
- `'1979-02-19 13:30:23.001023-05'`
- `'1979-02-19 13:30:23.001023-3:30'`
- `'1979-02-19T13:30'`
- `'1979-02-19T13:30:23'`
- `'1979-02-19T13:30:23.001023'`
- `'1979-02-19T13:30:23.001023Z'`
- `'1979-02-19T13:30:23.001023-05'`
- `'1979-02-19T13:30:23.001023-3:30'`
- `'19790219T'`
- `'19790219T133023.001023-330'`

22.12.3 Using Interval Literals

You can assign a value to a data object of type Interval using literals. You can also use interval literals to provide a parameter of type interval to a method.

Interval literals enable you to define a date using different levels of precision.

To define a time literal you must use a combination of values and followed by their time unit enclosed by single quotes.

Table 22–6 shows the available time unit fields.

Table 22–6 Time Unit Suffixes

Time Unit Suffix	Description
Y	Year

Table 22–6 (Cont.) Time Unit Suffixes

Time Unit Suffix	Description
M	Month
d or D	Day
h or H	Hour
m	Minutes
s or S	Seconds
x	Microseconds

Table 22–7 shows examples of interval literals:

Table 22–7 Examples of interval literals

Example	Description
`'1Y1M3h2m1.500s'`	1 year, 1 month, 3 hours, 2 minutes and 1.500 milliseconds
`'1.5h'`	1.5 hours
`'3M15d'`	3 months and 15 days

22.12.4 Using Array Literals

You can assign a value to a data object of type Interval using literals. You can also use interval literals to provide a parameter of type interval to a method.

To define an array literal you must provide a list of values separated by comas and enclosed by brackets. You can also specify the values using literals or attributes from data objects.

The following list shows examples of array literals:

- `[ "One", "Two", "Three" ]`
- `[ 1, 2, 3 ]`
- `[ customer.firstName, customr.lastName ]`

22.13 XPath BPM Extension Functions

The BPM Extension Functions enable you to access the following elements using XPath:

- Process and Project Data Objects
- Arguments
- Activity Instance Attributes

In XPath this is the only way of accessing the value of the described elements in your BPMN process.

22.13.1 getActivityInstanceAttribute

Returns the value of a specific activity instance attribute. See Section 8.4, "Introduction to Activity Instance Attributes" for more information about the supported activity instance attributes.

Signature:

```
bpmn:getActivityInstanceAttribute(activityName, attributeName)
```

Arguments:

`activity name` - The name of the activity that contains the activity instance attribute.

`attributeName` - The name of the activity instance attribute for which you want to find out the value.

Examples:

```
bpmn:getActivityInstanceAttribute(userTask, priority)
```

```
bpmn:getActivityInstanceAttribute(userTask, title)
```

22.13.2 getDataInput

Returns the value of a specific input argument in a data association.

Signature:

```
bpmn:getDataInput(dataInputName)
```

Arguments:

`dataInputName` - String that contains the name of the data input argument.

Examples:

22.13.3 getDataObject

Returns the value of a specific data object.

Signature:

```
bpmn:getDataObject(dataObjectName)
```

Arguments:

`dataObjectName` - String that contains the name of the data object whose value you want to obtain.

Examples:

```
bpmn:getDataObject(discount)
```

```
bpmn:getDataObject(approveTermsOutcome)
```

22.13.4 getDataOutput

Returns the value of a specific data output argument in a data association.

Signature:

```
pmn:getDataOutput(dataOutputName)
```

Arguments:

`dataOutputName` - String that contains the name of the data output argument.

Examples:

22.13.5 getGatewayInstanceAttribute

Returns value of a specific activity instance attribute in a gateway. See Section 8.4, "Introduction to Activity Instance Attributes" for more information about the supported activity instance attributes for gateways.

Signature:

```
bpmn:getGatewayInstanceAttribute(gatewayName, attributeName)
```

Arguments:

`gatewayName` - String that contains the name of the gateway that contains the attribute whose value you want to obtain.

`attributeName` - String that contains the name of the attribute whose value you want to obtain.

Examples:

22.13.6 getProcessInstanceAttribute

Returns value that corresponds to a process activity instance attribute. See Section 8.4, "Introduction to Activity Instance Attributes" for more information about the supported activity instance attributes.

Signature:

```
bpmn:getProcessInstanceAttribute(attributeName)
```

Arguments:

`attributeName` - String that contains the name of the process instance attribute whose value you want to find out.

Examples:

```
bpmn:getProcessInstanceAttribute(owner)
```

Part VII

Using SOA Components

This part provides on overview on how to use SOA Composites with BPM Projects. It also describes how to use other SOA applications with Oracle BPM.

This part contains the following chapters:

23

Using SOA Composites with BPM Projects

This chapter describes how to use SOA Composites to design a BPMN process and integrate it with other SOA components. SOA Composites show the dependencies between a BPMN process and the other components of your BPM project.

This chapter includes the following sections:

For detailed information about SOA Composites, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

23.1 Introduction to SOA Composites

SOA Composites group interrelated components, enabling the integration of different technologies into a single application. The composite provides a single deployment and management model, end-to-end data security, and unified metadata management to the components it contains.

BPM projects use the SOA technology. All BPM projects are layered on top of an SOA Composite. They use this composite to store information that describes the relationship between the different components in your BPM project and the services they expose.

BPMN processes are a component in the SOA Composite. You can view how a BPMN process relates to the rest of the components in the SOA Composite, using the SOA Composite editor.

The SOA Composite of a BPM project shows the following:

- The available SOA components to use in your BPM Project
- The BPMN and BPEL processes in your BPM Project
- The relationship between the SOA components and the processes

If the SOA Composite contains components or external references that expose services, then these appear in the business catalog. See Chapter 12, "Using the Business Catalog", for more information about the business catalog.

When you add a component to the SOA Composite, it automatically appears in the business catalog so that you can use it in your BPM project.

The SOA Composite is the unit that you use to deploy your BPM project. The components and dependencies that appear in the SOA Composite specify how to deploy a project. If you remove a process or a wire from the SOA Composite, then even if they still appear in the BPM Project they are ignored when you deploy the project.

23.1.1 Understanding the Relationship Between SOA Composites and SOA Components

When you run a BPM project the SOA engine creates a SOA composite instance. The SOA composite instance contains an instance of each of the components defined in the SOA composite. For example, if your SOA composite defines a BPMN process and a human Task, then the SOA composite instance contains a BPMN process instance and a human task instance.

The components in a SOA composite instance are independent from each other. When you terminate one of them, this does not affect the other components in the SOA composite instance. In a SOA composite instance that contains an instance of a BPMN process and an instance of a Human Task, terminating the BPMN process instance does not terminate the Human Task. After terminating the BPMN process, the Human Task instance is still available and you can access it using Oracle Worklist application. But completing that Human Task does not have any effect on the terminated process.

In a similar way, when an interrupting timer or message boundary event arrives to a user task, the BPMN process instance leaves the user task but the associated Human Task remains available. Because the interrupting timer or message boundary event arrived before the user completes the user task, the human task remains unfinished, and you can still access it thought the Worklist application. However running that human task does not have any effect on the BPMN process.

23.1.2 Working with SOA Components

All the SOA components and external references that are exposed as services in the SOA Composite appear in the business catalog in your Business Project.

If you created your BPM project based on an existing SOA project, then all the components and external references exposed as services in your SOA project automatically appear in the business catalog.

If there are activities in your BPMN process that use a component in their implementation, then the SOA Composite shows a wire between the BPMN process and the component.

Wires represent a relationship between a service and a reference. When you save a BPMN process Oracle BPM Studio automatically updates the wires between the BPMN process and the components it uses. Services represent the interface a component exposes. References represent the service interfaces a component requires. For more information about services and references, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

To implement the activities in your BPMN process you must assign them an SOA component. To add these components to your BPM Project you must use the SOA

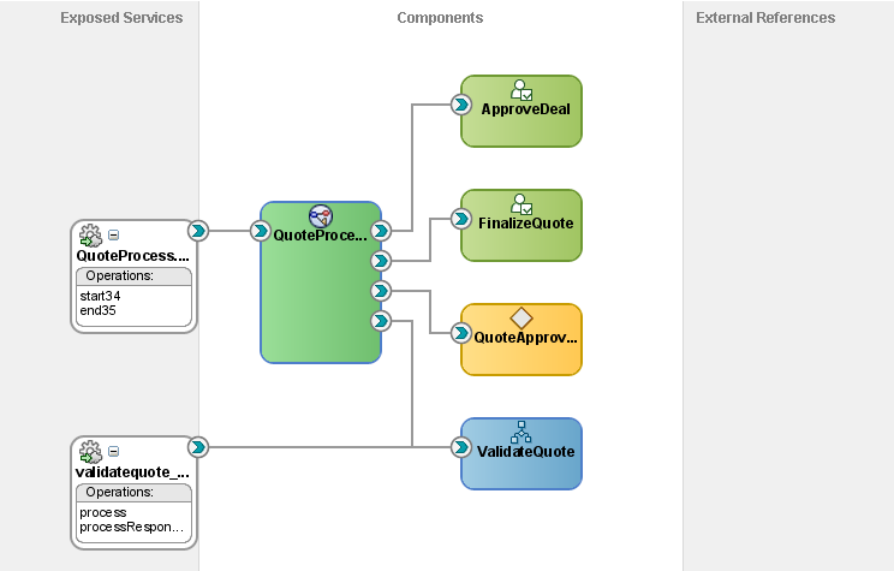
Composite editor. From the SOA Composite editor you can add the following SOA components to your BPM Project:

- Service Adapters
- Human Tasks
- Business Rules
- Mediators
- BPEL Processes

If the SOA component that you added to the SOA Composite exposes itself as a service, then the Component appears in the business catalog. You can use any of the components in the business catalog to implement the activities in your BPMN Process. For more information about how to implement BPMN activities, see the chapters in the following parts:

- Part V, "Working with Business Components"
- Part VI, "Controlling the Process Flow"

Figure 23–1 BPMN process in an SOA Composite



This SOA Composite diagram show a BPMN process that uses Human Tasks, Business Rules and Service Adapters to implement its activities. Note that the BPMN process has a link to each of these SOA components.

****

23.1.3 BPMN Process in SOA Composites

When you add a BPMN process it is automatically added to the SOA Composite. The BPM process appears as a component in the SOA Composite.

If the BPMN process contains a start event of type message, then the interface of the process appears as an exposed service.

The SOA Composite shows how your process depends on the different components your BPM Project uses. If an activity in your project uses a service exposed by an SOA component for its implementation, then the SOA component shows a line between the

exposed service and the BPMN process. This line represents the wire that links the BPMN Process and the exposed service.

23.1.4 How Do BPMN Errors Affect the SOA Composite Status

The status of the components in the SOA composite determine the status of the SOA composite. If an exception occurs in a BPMN process, then the status of the SOA composite is marked as faulted. Even is the BPMN process handles the exception and finishes running successfully, the status of the SOA composite is marked as faulted.

23.2 Opening the SOA Composite in a BPM Project

BPM projects are layered on top of a SOA project. The SOA project contains an SOA Composite. You must use the SOA Composite editor to add SOA components to your BPM project. The SOA components you add to the SOA Composite automatically appear in the business catalog of your BPM project.

23.2.1 How to Open the SOA Composite in a BPM Project

You can open the SOA Composite contained in your BPM project to add new SOA components or edit the existing ones.

To open the SOA Composite in a BPM project:

1. Select the **Application** window.
2. Double-click the *composite.xml* file located in the SOA Content directory of your project.

   The SOA Composite editor opens.

23.3 Opening BPMN Processes from the SOA Composite in a BPM Project

BPM projects use the SOA technology, therefore they contain an SOA Composite. You can use the SOA Composite editor to view the dependencies of your BPM processes with other components in your BPM project, or to add new components to your BPM project.

23.3.1 How to Open a BPMN Process from the SOA Composite in a BMP Project

You can open a BPMN process from the SOA Composite without having to switch to the BPM Project Navigator.

To open a BPMN process from SOA Composite in a BPM Project:

1. Open the **SOA Composite editor**.
2. Double-click the BPMN process you want to open.

   The BPMN process editor appears. Any changes you make to a process appear on the SOA Composite.

23.4 Adding a BPMN Process from the SOA Composite Editor

You can add new BPMN processes directly from the SOA Composite editor without having to switch to the BPM Project Navigator.

23.4.1 How to Add a BPMN Process from the SOA Composite Editor

If you identify the need of a BPMN process while analyzing the business application infrastructure, then you can directly add it without leaving the SOA Composite editor.

To add a BPMN process from the SOA Composite Editor:

1. Open the **SOA Composite editor**.
2. Select **BPMN Process** from the Service Components section in the Component Palette.
3. Drag the selected component to the Components area in the SOA Composite editor.

23.4.2 What Happens When You Add a BPMN Process from the SOA Composite Editor

The BPMN process appears as a component in the SOA Composite editor. The new process appears in the Processes folder in the BPM Project Navigator.

To edit the BPMN process right-click it and select edit, or double click the BPMN process.

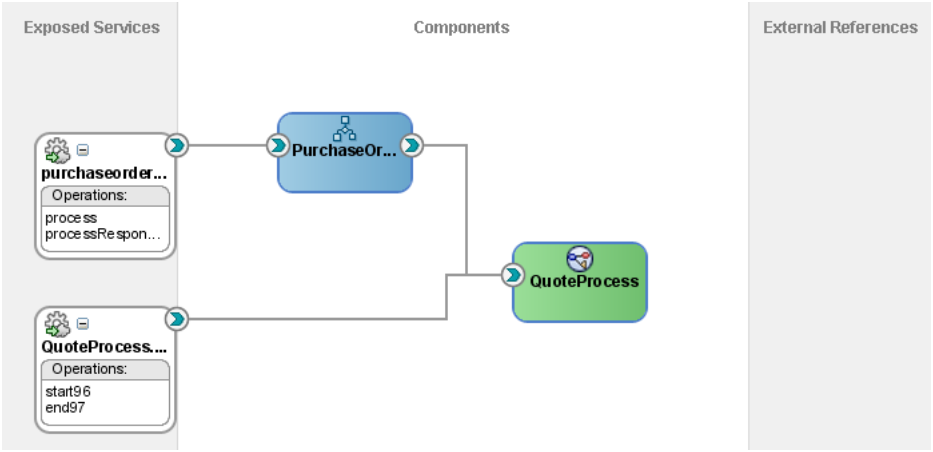
23.5 Integrating with BPEL Processes Using the SOA Composite

The SOA Composite editor shows the BPEL processes and the BPMN processes in your project. You can use the Composite editor to design the integration between a BPEL process and a BPMN process.

To use a BPMN process from a BPEL process you must add the BPMN process as a partner link in the BPEL process. To add the BPM process as a partner link in the BPEL process you must use the SOA Composite editor. After adding the BPMN process as a partner link, you can use the BPEL editor to link the BPMN process to the activities in the BPEL process. For more information about editing BPEL processes, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

To use a BPEL process from a BPMN process you must add the BPEL process to the SOA Composite. After you do this the BPEL process appears in the business catalog. You can use the BPEL processes in the business catalog to implement the activities in your BPMN process.

Figure 23–2 BPMN Process as a Partner Link in a BPEL Process



This SOA Composite diagram shows a BPEL process, Purchase Order, that uses the BPMN process Quote Process. The BPMN process is a partner link in the BPEL process.

****

23.6 Adding a BPMN Process as a Partner Link in a BPEL Process

To use a BPMN process from a BPEL process you must add it as a partner link. Then you can use the BPEL editor to invoke the BPMN process from the activities in the BPEL process.

23.6.1 How to Add a BPMN Process as a Partner Link in a BPEL Process

To use a BPMN process from a BPEL process, you must first add the BPMN process as a partner link in the BPEL process.

To add a BPMN process as a partner link in a BPEL process:

1. Open the **SOA Composite editor.**
2. Place the mouse pointer over the BPEL process component.

   Orange arrows appear to the sides of the BPEL process component. The arrow on the left enables you to add a new service. The arrow on the right enables you to add a new reference.
3. Click the right arrow and drag.

   A green link appears and all the services exposed by the components in the composite, including those exposed by BPMN processes, turn green.
4. Drop the link on the service of the BPMN process you want to add as a partner link.

23.6.2 What Happens When You Add a BPMN Process as a Partner Link in a BPEL Process

The BPMN process appears as a partner link in the BPEL process and you can invoke the BPMN process from the BPEL process.

23.7 Connecting to a BPMN Process Using Web Services

If a BPMN process defines a process interface, then you can connect to that process using web services. All the BPMN processes that define a process interface appear in the SOA Composite. For more information about defining a process interface, see Chapter 21, "Defining the Process Interface".

To connect to a BPMN process using a custom web service client you need the following information:

- Web Service Location

  http://*host*:*port*/soa-infra/services/*partition*/*composite*!*revision*/*process*.service
- WSDL Location

  http://*host*:*port*/soa-infra/services/*partition*/*composite*!*revision*/*process*.service?WSDL
- List of Composites

http://*host*:*port*/soa-infra

Table 23–1 describes the information used to construct the previous URLs.

Table 23–1 URL Field Description

Field	Description
Host	The server where the BPMN Service Engine is running.
Port	The port to connect to the BPMN Service Engine.
Partition	The MDS partition where the SOA composite resides.
Composite	The name of the SOA Composite.
Revision	The revision number that indicates the version of the composite. This field is optional. If you do not specify the revision, then the BPMN Service Engine uses the default revision of the composite.
Process	The name of the BPMN process.

23.8 Building a BPM Project

You must build your BPM project before deploying it to a BPMN Service Engine. You can build your BPM project from Oracle JDeveloper.

After you build the BPM project, the Compiler Log window displays the results. If the build is successful, then you can deploy to the BPMN Service Engine.

If there are any errors, you can select the Compiler tab and click the errors to open the corresponding editor and correct them.

23.8.1 How to Build a BPM Project

To build a BPM project:

1. Open the BPM project.
2. Click the **Application Navigator** window.
3. Right-click the file that corresponds to your project.
4. Select **Make *Project***.

   Oracle JDeveloper compiles the BPM Project. The Compiler Log window displays the results of the compilation.

23.8.2 What Happens When You Build a BPM Project

After you successfully build a BPM Project you can deploy it to a BPMN Service Engine. The process of deploying a BPM Project is identical to deploying a SOA Project.

For more information on how to deploy a SOA Project, see *Deploying SOA Composite Applications* in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

24

Working with Guided Business Processes

This chapter describes how to use Guided Business Processes to organize the activities in your process into milestones. You can use milestones to make your process easier to run for inexperienced users. Guided Business Processes hide the complexity of the process and guide the end-user through the tasks that are relevant to them.

24.1 Introduction to Guided Business Processes

Guided Business Process enable you to group the interactive activities in your BPM 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.

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

Guided Business Processes enable directing end users to complete a business process through a guided set of steps associated with the process. By following the steps outlined in a Guided Business Process, end users require less training to complete a business process, and the results of the process are more predictable.

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 end user has to 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 resume it at a later time.

Figure 24–1 An Activity Guide

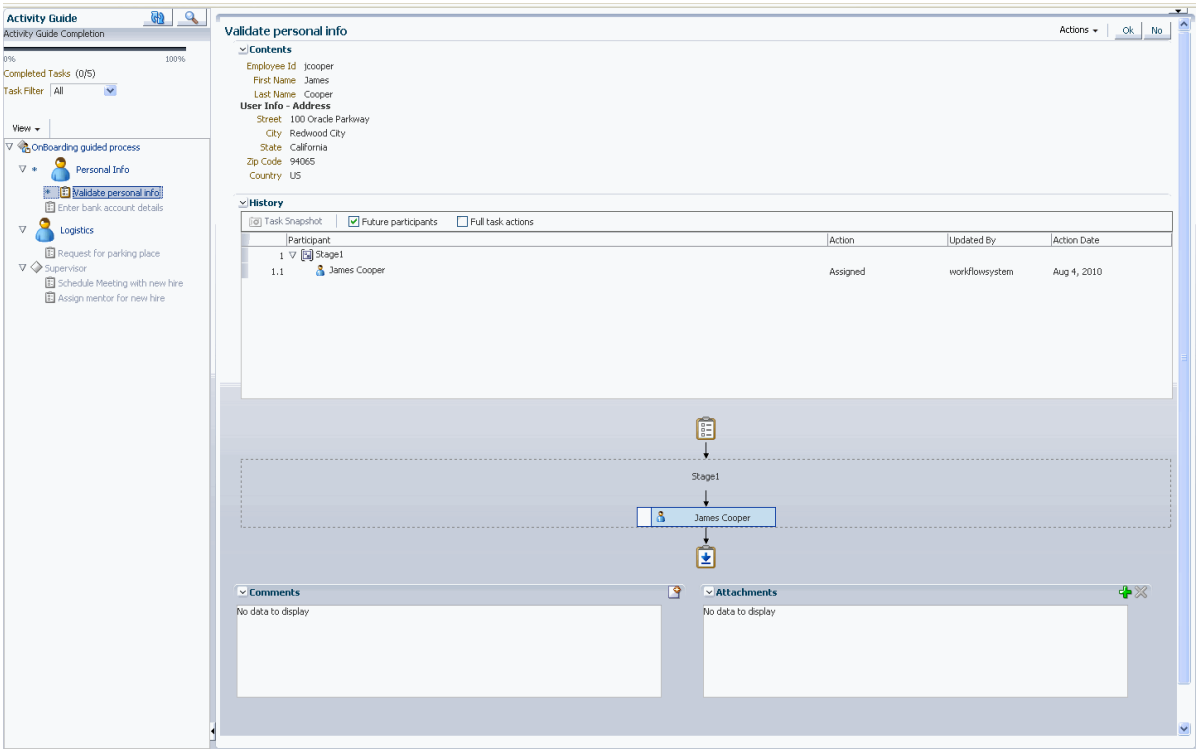
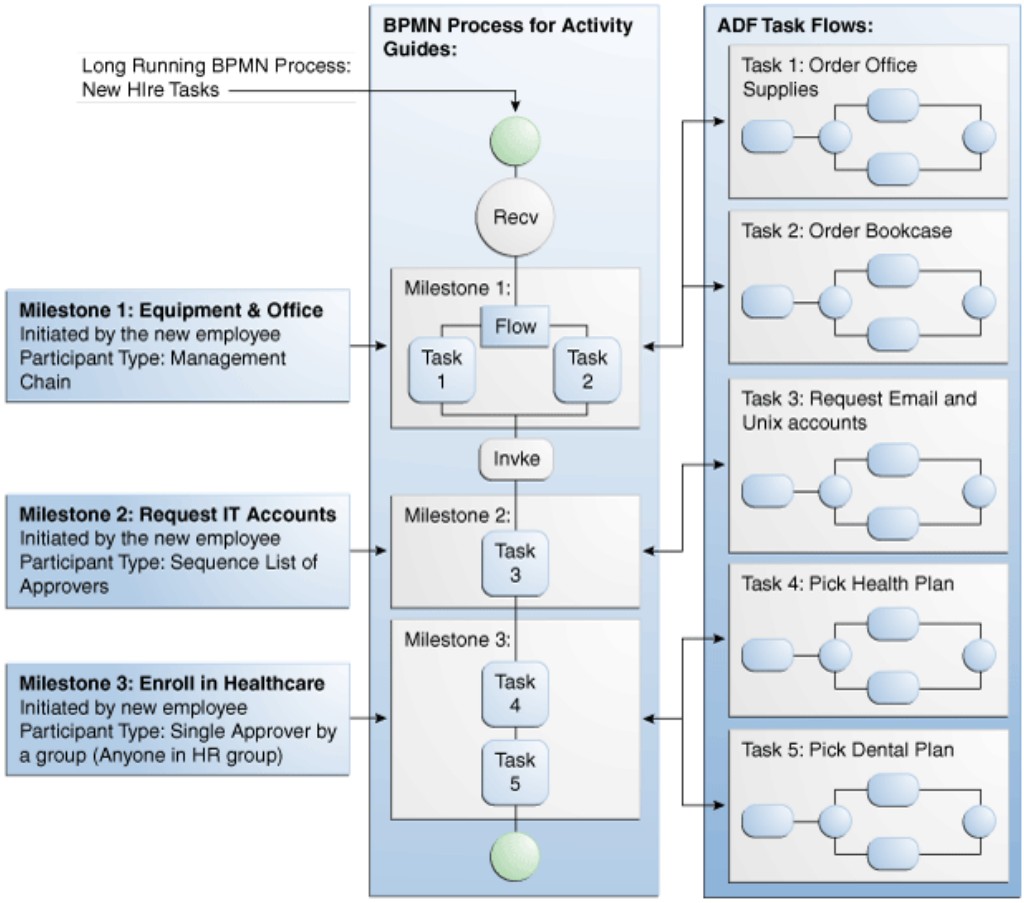


Figure 24–2 An Example of a Guided Business Process



Leveraging Service-Oriented Architecture

Service-Oriented Architecture is the foundation for Guided Business Processes. Guided Business Processes use SOA composite processes, leveraging the following SOA functionality:

- **Reuse of existing investments**: Oracle SOA Suite provides a foundation for and access to re-usable services. Composite processes leverage existing investments by running on the SOA platform and accessing existing services provided by the platform. By enabling the use of existing systems, Oracle SOA Infrastructure increases the usefulness and value of these systems.
- **Process-oriented software**: Service-Oriented Architecture combined with process orchestration infrastructure act as the controlling agent for the business process across multiple disparate systems.

When to Use Guided Business Processes

Guided Business Processes enable running large-scale, long-running, multiuser processes that consume and reuse taskflows built by other teams. For example, the finance and human resources departments of an organization may access the same human taskflows in different business processes. Using Guided Business Processes enables re-using existing taskflows in a large composite, creating a more meaningful business process for end users.

Oracle SOA infrastructure provides access to re-usable services that you can use in your business processes. Guided Business Processes leverage existing services, processes and task flows to create long-running, multiuser processes.

Guided Business Processes provide the following functions and features:

- Re-using tasks and taskflows within a large composite, to avoid redesigning and re-coding tasks and activities.
- Using SOA infrastructure to orchestrate tasks, creating a flow of business processes.
- Using Milestones to modularize tasks into manageable chunks, while presenting to end users a set of related, guided tasks.

  For example, a long process with one hundred tasks can be broken down into ten or twenty Milestones. End users need only step through a few Milestones rather than, say, one hundred individual tasks.

Guided Business Processes: Design Time and Run Time

Guided Business Processes consist of both design time components and run time interfaces.

24.1.1 Guided Business Process Design Time Architecture

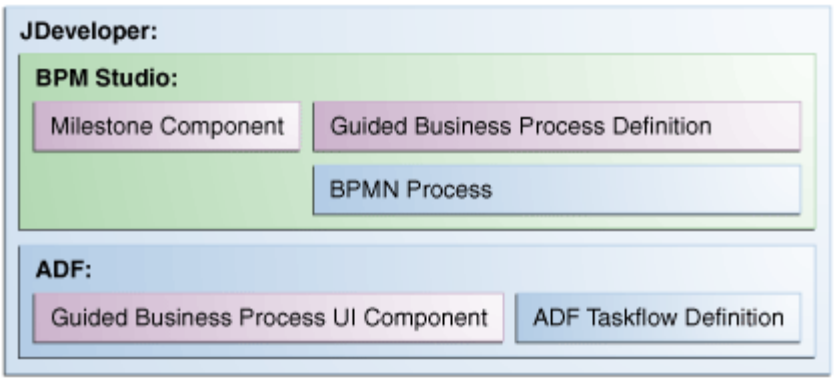
Guided Business Processes use Oracle Business Process Management to provide a comprehensive, standards-based, easy-to-use solution for creating, deploying, and managing composite application business processes with both automated and human workflow steps—all in a service-oriented architecture.

Guided Business Processes leverage features of Oracle Fusion Middleware, such as security, scalability and high availability. The following features enable composite processes to be exposed as Guided Business Processes:

- **Technology abstraction**. By using metadata, the actual implementation of business logic need not rely on any one technology.
- **Declarative development**. Using metadata to define business logic and business processes eliminates the need for coding when creating a Guided Business Process.

Developing Guided Business Processes involves creating a composite application which contains a SOA project with a BPM Project. The BPM process exposed as a Guided Business Process, consists of an Activity Guide that contains milestone activities. A separate client application must also be developed as an end user interface for the Guided Business Process.

Figure 24–3 Guided Business Process Design Time Architecture



You can develop a user interface for Guided Business Processes using any of the following:

- Oracle ADF
- Oracle WebCenter Framework
- Guided Business Process Run-Time Services.

24.1.2 Components of a Guided Business Process

A Guided Business Process is a BPMN process that orchestrates a set of human tasks and provides a common user interface to complete and track these tasks. To define a Guided Business Process you create an Activity Guide that comprises the following components:

- **Milestone**: A milestone is a group of human tasks which must be completed to accomplish a particular goal. A milestone is complete when its human tasks have been completed. Similarly, an Activity Guide is complete when a specific set of Milestones are completed. A milestone can contain Human Task activities and other BPMN activities.
- **Human Tasks**: It is possible to invoke an ADF task flow from an Activity Guide. To do so, a Human Task component is placed in the Activity Guide and bound to an ADF task flow.
- **Other Tasks**: Activity Guides can include other tasks such as service calls. However, these automated tasks do not appear in the Activity Guide tree at run time.

Activity Guides with a simple, sequential process execution must complete all Milestones. Similarly, all Human Task components within a milestone must be completed to complete the milestone.

Activity Guides containing branching and conditional logic may sometimes complete execution without necessarily completing all Milestones. Similarly, some Human Task components within these Milestones may be skipped as well.

24.1.3 Guided Business Process Run-Time Architecture

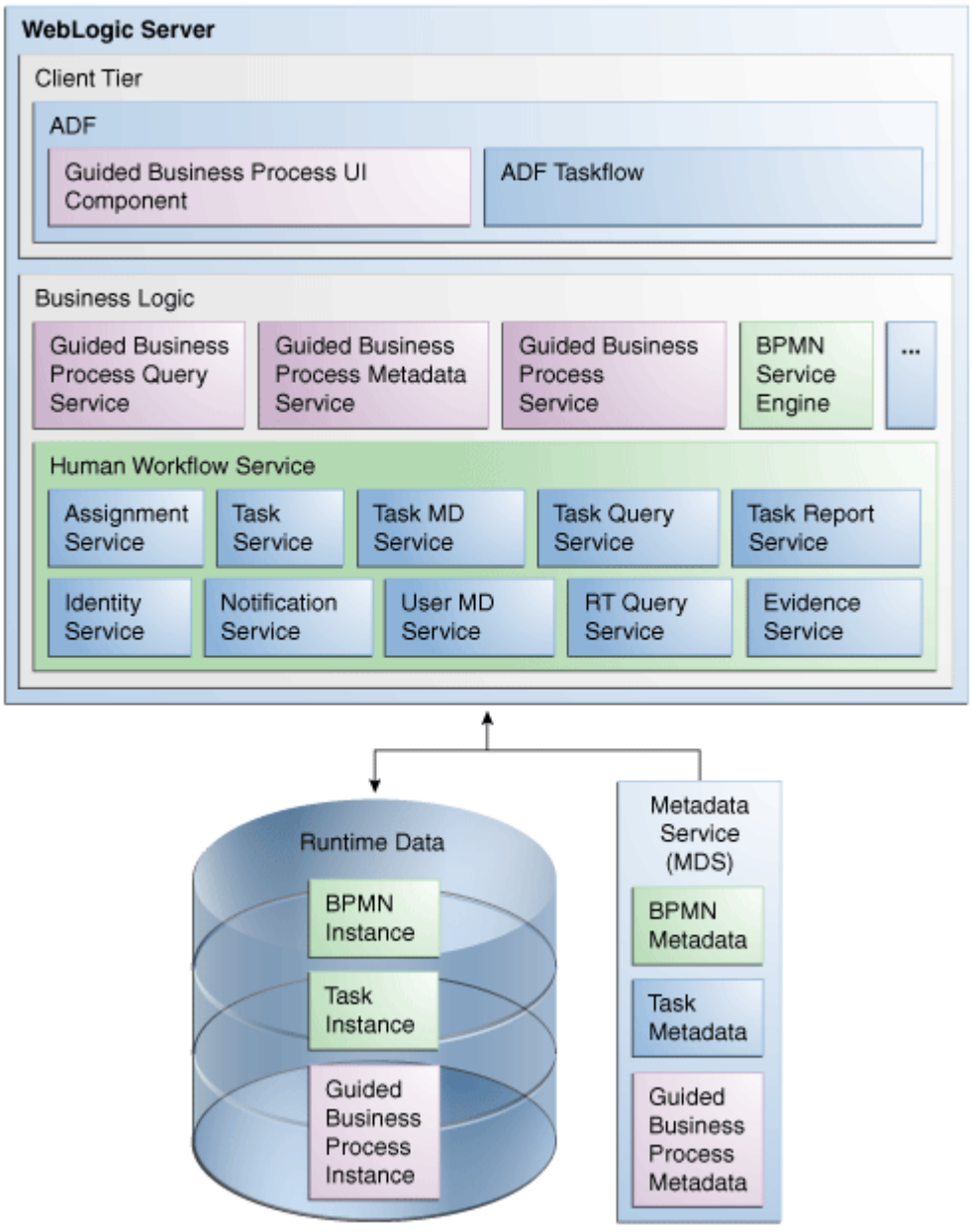
Guided Business Processes rely on Oracle Business Process Management to orchestrate tasks, combining Oracle Business Process Management, worklist applications, and human task flows to link disparate human tasks to a greater long-running process.

At run time, Guided Business Processes manifest as BPMN process instances orchestrating ADF task flows within Milestones. The run-time engine for BPMN guided business processes is the BPMN Service Engine. The BPMN engine delegates all human task operations to Human Workflow services.

You can view the process instances organized into an Activity Guide using a custom application developed with Oracle ADF UI, Oracle WebCenter UI or Guided Business Process access APIs. The process instances that you view in an Guided Business Process client also appear in the Worklist application, but they are not organized into milestones.

As shown in Figure 24–4, the Guided Business Process run-time architecture is composed of the client, business logic and data tiers.

Figure 24–4 Guided Business Process Run-Time Architecture



Run time support includes the Guided Business Process Query, Guided Business Process Metadata and Guided Business Process Instance Management services. The run time components interface with Oracle Business Process Management and the Human Workflow Service.

You can use any of the following as a basis for a Guided Business Process client application:

- Oracle Worklist application
- Oracle ADF
- Oracle WebCenter Framework
- Guided Business Process Run-Time Services.

24.1.3.1 Client Tier

The Guided Business Process run time front end, or client application, enables end users to follow the task flow defined at design time in the Activity Guide. Using the client application, end users can:

- Expand a milestone and drill down to the tasks it includes
- Complete the tasks defined in the Milestones
- View the status and percent completion of the business process as the Guided Business Process runs.

You can manage and access a Guided Business Process using the following:

- **Customized Client Application**: You can create customized client applications to display the Guided Business Process to end users using Oracle ADF UI, Oracle WebCenter UI or Guided Business Process APIs.
- **Worklist Application**: An out-of-the-box interface for displaying the tasks in a Guided Business Process, to users completing the guided tasks.
- **Oracle Enterprise Manager Application Server Control console**: A platform for administering and monitoring Guided Business Process instances. The console is also useful for testing Guided Business Processes following development.

Features of the Run-Time User Interface

Following are some features of the run-time user interface:

- **Auto Focus**: Once a Guided Business Process is deployed, you can generate a new Guided Business Process instance with milestone nodes and task nodes. When you access the Guided Business Process instance from the user interface, auto focus selects the first uncompleted task in the process. On completion of a task, auto focus selects the next uncompleted task in the process. Auto focus also supports optional tasks, when you skip an optional task it selects the next uncompleted task in the process.
- **Future Milestones and Tasks**: When accessing a Guided Business Process, the Milestones display in the Activity Guide tree. The Activity Guide also displays the milestones and tasks to be completed in the future. This provides users with a holistic view of the tasks and milestones to be completed. If the Guided Business Process is to be executed sequentially, future tasks and milestones are grayed out. Future tasks and milestones may be viewed, but not executed.
- **Branching**: Branching involves placing a switch in the business process flow which enables splitting the process into two or more branches.

Conditions are set to determine which branch executes. If the flow branching is determined by a condition, then the milestone branching node displays as an ellipsis ("...") in the Activity Guide tree. The milestones to be completed for the selected branch display only when the switch executes at real time.

- **Parallel Milestones**: Parallel milestones are milestones that can be completed at any time, in any order without following a specific sequence, if previous milestones have completed.
- **Disabling Completed Tasks**: When the end user completes a task, the link to the task is grayed out to avoid distracting the user with links that are no longer active.
- **Progress Indicator**: The progress indicator provides feedback to the end user about their progress completing the entire Guided Business Process. The progress is shown using a bar graphic with the completion percentage of the Guided Business Process.
- **Completed Tasks**: Shows the completed number of tasks out of a total number of tasks.
- **Filtering**: Enables to filter the tasks within a milestone based on their functional state. The possible functional states to use when filtering tasks are: All, In Progress, Completed, Required, Optional.
- **Explicit Refresh**: Activity Guides automatically refresh task flows when you complete a task using Activity Guides. If you complete the task from another application such as Oracle BPM WorkSpace or a custom UI client, you must explicitly refresh the Activity Guide. Explicit refresh is not available by default. To enable explicit refresh you must enable the ShowRefreshButton property.

  For more information on how to configure the ShowRefreshButton property, see Section 24.7, "Configuring Activity Guide Properties".

24.1.3.2 Business Logic Tier

The business logic tier includes the following components:

- Guided Business Process Metadata Service
- Guided Business Process Query Service
- Guided Business Process Instance Management
- Guided Business Process Instance Schema
- Human Workflow Service

Guided Business Process Metadata Service

The SOA composite associated with an Guided Business Process drives it at run time. As such, no run-time environment data is stored in the Guided Business Process metadata. The Guided Business Process Metadata Service locates and retrieves the Guided Business Process definition at run time from the Metadata Service (MDS).

Guided Business Process Query Service

The Guided Business Process query service retrieves the Guided Business Process instance information based on specified search criteria. The query service uses the existing workflow service to query Guided Business Process data. The Guided Business Process query service is registered to the workflow service locator as an additional workflow service.

Guided Business Process Instance Management

Guided Business Process run-time states are maintained as separate objects, enabling Guided Business Processes to have a state separate from the SOA composites with which they are associated. The related SOA composite instances are managed by an SOA composite run-time manager.

Guided Business Process Instance Schema

A schema defines the structure of Guided Business Process instances. The schema represents Guided Business Process data persisted to the database at run time.

Human Workflow Service

Workflow services enable you to interleave human interactions with connectivity to systems and services within an end-to-end process flow. In BPMN workflow services are linked to the BPMN process using a user task. The process assigns a task to a user or role and waits for a response. The users act on the task using Oracle BPM Worklist.

The Human Workflow Service is responsible for handling all interactions with users or groups participating in the business process. It does this by creating and tracking tasks for the appropriate users in the organization. Users typically access tasks through a variety of clients, including Oracle Worklist application, Oracle BPM WorkSpace, e-mail, portals, or custom applications.

For more information about the Human Workflow Service, see the chapter "Introduction to Human Workflow" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

24.1.3.3 Data Tier

Oracle Business Process Management persists Guided Business Processes, BPMN process and task instances to the database at run time. Oracle Metadata Repository (MDS) stores the schemas of available services, including BPMN processes, task and Guided Business Process metadata. The schemas are used when instantiating a Guided Business Process.

24.2 Guided Business Process Use Cases

The following use cases show you different situations where to use Guided Business Processes:

- Section 24.2.1, "Online Public Sector Form Processing"
- Section 24.2.2, "Online Loan Application Procedure"

24.2.1 Online Public Sector Form Processing

Many public sector organizations process forms manually, a labor intensive and environmentally wasteful procedure.

For example, a state agency must provide, collect and process various forms to issue fishing and hunting licenses. The state agency hires additional outside contractors for the summer and fall season to handle the increase of license applications more efficiently while avoiding information loss or negligence.

Rather than manually processing the license applications, the end-to-end form processing procedure can be modeled as a Guided Business Process with two Milestones.

The following outline of an example form processing using Guided Business Process is generic, and can be adapted to enable end-to-end processing of similar forms:

Milestone 1: Filling in and Submitting an Application

Applicants on the state Web site for the Department of Fishing and Hunting can click the Apply button to fill in and submit an application for a fishing or hunting license.

The milestone includes the following tasks:

1. Personal details: Applicants must fill in personal details such as name, address, and so on.
2. License selection: Applicants select a fishing or hunting license.
3. Form submission: Applicants click the Submit button to send the forms to the state fishing and hunting department. The workflow selects an application approver and e-mails the approver a request for review and follow-up.

Milestone 2: Application Processing and Result Notification

Applications for hunting licenses require review and manual approval or rejection.

The milestone includes the following tasks:

1. Approval/Rejection of application: The license application approver navigates the approval/rejection flow.
2. Status notification: The workflow sends the applicant a notification regarding the status of the license application.

With its intuitive guided user experience, the Guided Business Process maximizes the efficiency of the license application process while increasing the productivity of license approvers. In addition, the Guided Business Process enables monitoring the end-to-end application process at both the front- and back-end levels.

24.2.2 Online Loan Application Procedure

In the loan-origination industry, banks must consider several business factors: process consolidation, regulatory compliance and faster product delivery, for example. Loan products change frequently, often depending on state or region where the loan is offered.

The following example focuses on a subset of loan origination. As such, only specific processes are illustrated.

Business Process Flow

In this example, the business process flow for an online loan application procedure is as follows:

1. Application or registration: The customer enters qualification information for a loan product.
2. Processing or locking of a loan: The customer agrees to a specific product and rate. This stage of the procedure spawns several sub-processes that gather additional information on the customer. This information enables the underwriters to decide on the loan.
3. Underwriting the loan: An underwriter uses the information gathered to approve or reject the loan.
4. Closing: The organization selling the loan closes the loan application and grant process.

These processes rely on several interrelated services and procedures, as illustrated in Figure 24–5.

Figure 24–5 Interwoven Loan Processing Services and Procedures

- Origination: The process of acquiring customer data and related data, making decisions based on processing the data and requesting data from third-party services.
- Third-party services: These are used to retrieve data on the customer and the item to be purchased with the loan.
- Secondary processes: These are processes that execute while the main process runs. This example focuses on pricing for various mortgage products.
- Servicing: Following origination, loans are either booked on the bank or sold on the secondary market, for example at other banks or by other loan vendors.

Although these processes appear simple, completing them involves many business challenges.

Increasingly, the interactions between real human actors in software must be coordinated. Humans are key participants in almost every software system, especially in collaborative processes and composite applications. Some common challenges are presented when involving humans interaction with structured workflow systems.

Deciding whether to grant a loan might entail working through a large set of rules based on the customer's credit history, income and other factors. These factors must be coordinated with several business process determined by the bank. Underwriters are alerted to approve or reject an applicant, depending on several factors, including the applicant's personal details and external data requests from third-party services.

A mortgage application Guided Business Process might include several milestones. The following Guided Business Process outline illustrates a mortgage application procedure.

Milestone 1: Loan Application

A potential customer registers on the loan provider Web site and applies for the loan through a series of guided tasks.

1. Registration: The applicant registers on the bank's Web site.

2. Product selection: The applicant selects a loan based on the type of product required and the products available regionally.
3. Application: The applicant enters the personal details required to apply for a loan.

Milestone 2: Application Processing

Once the loan application process has completed, the loan is processed and reviewed for approval.

1. Information retrieval: Various services are used to access information regarding the loan and the applicant's finances and personal details.
2. Review: An underwriter must manually review the loan application.
3. Approval: Based on the data reviewed, the approver must approve or reject the loan.

Milestone 3: Closing

Once the loan has been approved it is ready for closing.

The milestone includes the following tasks:

1. Closing appointment: The title company or closing attorney sets an appointment with the customer.
2. Closing documents: The loan closer gives the loan documents to the closing agent and provides to the customer any required documents, and as the final closing

24.3 Standards and Guidelines for Working with Guided Business Processes

The following standards and guidelines apply to Guided Business Processes:

- Guided Business Processes must be deployed to a standalone Oracle WebLogic Server.

24.4 The Typical Flow of Developing a Guided Business Process

The following describes the main workflow of developing a Guided Business Process:

1. Develop the Guided Business Process:
   a. Develop a BPMN process and configure it as a Guided Business Process.
   b. Configure Milestones and associated tasks.
   c. Develop a task flow.
   d. Deploy, instantiate, and test the Guided Business Process.
2. Develop a Guided Business Process front end for use at run time. To develop a front end, you can:
   - Develop a client application with Oracle ADF UI or Oracle WebCenter UI.
   - Use Oracle Worklist application as a client application.
   - Develop a custom UI with Guided Business Process run-time services.
3. Deploy the Guided Business Process client application.
4. Monitor Guided Business Process instances using the Oracle Enterprise Manager Application Server Control console.

24.5 Introduction to Developing a Guided Business Process

Guided Business Processes allow you to organize your processes into milestones. These milestones are meaningful to the end-user and hide the complexity of the process by showing them only relevant information to their tasks.

You can create a Guided Business Process and organize the tasks in your process into a set of milestones. Using milestones enables you to run your process and track its completion in a more efficient way.

The following list describes some features you can use:

- **Branching**: Branching involves placing a switch in the business process flow which enables splitting the process into two or more branches.

  Conditions are set to determine which branch executes. If the flow branching is determined by a condition, the milestone branching node displays as an ellipsis ("...") in the Activity Guide tree. The milestones to be completed for the selected branch display only when the switch executes at real time.

- **Optional/Required Tasks**: By default, sequential tasks are required, unless configured otherwise. Tasks that are required for the completion of the Guided Business Process display with an asterisk (*). Required tasks cannot be skipped. Optional tasks are not required for the completion of the Guided Business Process. The user has the option to run optional tasks but they can choose to skip them. An example use for an optional task might be a survey that end users can optionally fill out after completing required tasks in an Guided Business Process.

  Configuring tasks as required or optional enables task filtering by required or optional task type at run time.

- **Parallel flow**: Parallel flows enable Guided Business Processes to perform multiple tasks at the same time, which is useful when you must perform several time-consuming and independent tasks.

  If previous milestones have completed, then end users can complete parallel milestones in any order.

- **Future Milestones and Tasks**: When accessing a Guided Business Process, the Milestones display in the Activity Guide tree. The Activity Guide also displays the milestones and tasks to be completed in the future. This provides users with a holistic view of the tasks and milestones to be completed. If the Activity Guide is to be executed sequentially, future tasks and milestones are grayed out. Future tasks and milestones may be viewed, but not executed.

- **Internationalization**: Guided Business Processes support internationalization using resource bundles. Resource bundles separate text, labels, messages and other locale-sensitive objects from the core source code, maintaining a single code base for all localized versions of Activity Guides. To enable internationalization for an Activity Guide, see Section 24.6.33, "How to Localize a BPMN Guided Business Process".

24.6 Developing a BPMN Guided Business Process

To develop a BPMN Guided Business Process you must first create a BPMN process. Then you can develop the Guided Business Process based on the BPMN process.

You can only define one Guided Business Process per project. The Guided Business Process is based on a BPMN process in the project. This process is the root process.

24.6.1 How to Develop a BPMN Guided Business Process

You can develop a Guided Business Process based on a BPMN process.

To develop a BPMN Guided Business Process:

1. Create a BPMN process or use an existing BPMN process.
2. In the BPM Project Navigator, expand the project that contains your BPMN process.
3. Right-click the **Activity Guide** node.
4. Select **Configure**.
5. In the **Title** text-field, enter a title to identify the Guided Business Process.
6. From the Root Process list, select the BPMN process you want to transform into a Guided Business Process.
7. Click **OK**.

24.6.2 What Happens When You Develop a BPMN Guided Business Process

You can add the user tasks in the BPMN process to the milestones in the Guided Business Process. When you finish building the Guided Business Process, you can access the BPMN process using a Guided Business Process client.

24.6.3 How to Add a New Milestone to a Guided Business Process

You can add a new milestone to an existing Guided Business Process.

To add a new milestone to a Guided Business Process:

1. In the BPM Project Navigator, select the **Activity Guide** node.
2. In the Structure window, right-click the **Activity Guide** node.
3. Select **New Milestone**.

   The New Milestone dialog appears.
4. Enter a title for the milestone.
5. Optionally, enter the number of tasks the user has to complete to consider this milestone completed in the Tasks Remaining field.

   At run time, the activity guide tree uses this value to show the percentage of completed tasks over the total tasks, in the progress indicator.
6. Click **OK**.

24.6.4 What Happens When You Add a Milestone to a Guided Business Process

The Guided Business Process displays a new milestone. You can add the user tasks in the root process to the new milestone.

24.6.5 How to Add a User Task to a Milestone

You can add a user task to a milestone in the Guided Business Process.

To add a user task to a milestone:

1. Open the root process.

2. Right-click the user task.
3. Select **Add to Milestone**.

   The Add User Task to Milestone dialog appears.
4. From the Milestone list, select the milestone to which you want to add the user task.

   If you did not create the milestone, you can create it using the Add button next to the Milestone list.
5. If the user task is the last task in the milestone, select the **Last Task in Milestone** check box.
6. Click **OK**.

24.6.6 What Happens When You Add a User Task to a Milestone

You can run the user task from a Guided Business Process client. The user task appears under the milestone in the activity guide tree.

24.6.7 How to Move a User Task to Another Milestone

Ensure that your Guided Business Process contains at least two milestones. If it contains only one milestone, the **Move to Milestone** option is grayed out.

To move a user task to another milestone:

1. In the BPM Project Navigator, select the **Activity Guide** node.
2. In the Structure window, right-click the user task.
3. Select **Move To Milestone**.

   The Move to Milestone dialog appears.
4. From the Milestones list, select the milestone where you want to move the user task.
5. Click **OK**.

24.6.8 What Happens When You Move a User Task to Another Milestone

The previous milestone does not list the user task anymore. The user task appears in the new milestone.

24.6.9 How to Order the Milestones in a BPMN Guided Business Process

Ensure that your Guided Business Process contains at least two milestones. If it contains only one milestone, the **Move to Milestone** option is grayed out.

To order the milestones in a BPMN Guided Business Process:

1. In the BPM Project Navigator, select the **Activity Guide** node.
2. In the Structure window, expand the **Activity Guide** node.
3. Move each milestone to the right position:
   1. Right-click the milestone.
   2. Select **Move-Up** or **Move-Down** according to where you want to move the milestone.

24.6.10 What Happens When You Order the Milestones in a Guided Business Process

The milestones appear in the order you arranged them in the activity guide tree.

24.6.11 How to Delete a Task from a Guided Business Process

You can delete a task from a Guided Business Process.

To delete a task from a Guided Business Process:

1. In the BPM Project Navigator, select the **Activity Guide** node.
2. In the Structure window, expand the **Activity Guide** node.
3. If the milestone that contains the task you want to remove is collapsed, then you must expand it.
4. Right-click the task you want to remove.
5. Select **Delete**.

   A confirmation message appears.
6. Click **OK**.

24.6.12 What Happens When You Delete a Task from a Guided Business Process

You cannot access that task from the Guided Business Process. The milestone that contained it does not list that task anymore.

24.6.13 How to Delete a Milestone

You can delete a milestone that you do not use or need from the Guided Business Process.

To delete a milestone:

1. In the BPM Project Navigator, select the **Activity Guide** node.
2. In the Structure window, expand the **Activity Guide** node.
3. Right-click the milestone you want to remove.
4. Select **Delete**.

   A confirmation message appears.
5. Click **OK**.

24.6.14 What Happens When You Delete Milestone

The milestone does not appear in the Guided Business Process. All the user tasks in the milestone are deleted from the Guided Business Process. You cannot access these tasks from the Guided Business Process anymore.

24.6.15 How to Configure an Optional Task

You can configure a task as optional so that it is not required to complete the Guided Business Process.

To configure an optional task:

1. In the BPM Project Navigator, select the **Activity Guide** node.

2. In the Structure window, expand the **Activity Guide** node.
3. Expand the milestone that contains the task.
4. Right-click the task.
5. Select **Edit**.

   The Edit User Task dialog appears.
6. Select **Show Task as Optional**.
7. Click **OK**.

24.6.16 What Happens When You Configure an Optional Task

By default all task are required unless you configure them to be optional.You must configure a skip button for the tasks you configure as optional.

When a group of users is assigned to a certain task, anybody in the group can claim that task. If after claiming the task the user decides not to complete it, then he can skip the task. When a user skips a task, the tasks is assigned back to the group so that the other users in the group can claim it and complete it.

24.6.17 How to Configure a Parallel Task Flow in a BPMN Guided Business Process

To configure a parallel task flow you must use gateways in the BPMN process. See Chapter 6, "Modeling Business Processes with Oracle BPM" for more information on how to use gateways.

24.6.18 How to Branch the Task Flow in a BPMN Guided Business Process

To branch the task flow you must use gateways and conditional sequence flows in the BPMN process. See Chapter 6, "Modeling Business Processes with Oracle BPM" for more information on how to use gateways and conditional sequence flows.

24.6.19 How to Configure a Task to Display a Blocked Icon

You can configure a task to display a blocked icon and message when it is not available for the end user to run it.

To configure a task to display a blocked icon and message:

1. In the BPM Project Navigator, select the **Activity Guide** node.
2. In the Structure window, expand the **Activity Guide** node.
3. Expand the milestone that contains the task.
4. Select **Edit**.

   The Edit User Task dialog appears.
5. Select the **Display Blocked Icon and Text** check box.
6. If you want the Guided Business Process to display a message explaining it is blocked, enter the message in the field.
7. Click **OK**.

24.6.20 What Happens When You Configure a Task to Display a Blocked Icon and Message

When the current task is completed and the next task is not instantiated, the activity guide tree displays a blocked icon. If you defined an explanation message, it appears as a tooltip when you locate the cursor over the blocked icon.

24.6.21 How to Configure an Icon for a Guided Business Process

You can configure a custom icon for the Activity Guide tree to display next to the Activity Guide node.

To configure an icon for a Guided Business Process:

1. In the BPM Project Navigator, right-click the **Activity Guide** node.
2. Select **Configure**.
3. Click the **Browse** button, next to the Icon Location field.

   The Browse Icons dialog appears.
4. Select an icon from your file system.
5. Click **Open**.

   The icon path appears in the Edit Activity Guide dialog.
6. Click **OK**.

24.6.22 What Happens When You Configure an Icon for a Guided Business Process

The activity guide tree uses this icon to identify the activity guide node. If you do not specify an icon, then the activity guide node does not display an icon.

24.6.23 How to Configure an Icon for a Milestone

You can configure a custom icon for the Activity Guide tree to display next to each milestone.

To configure an icon for a milestone:

1. In the BPM Project Navigator, select the **Activity Guide** node.
2. In the Structure window, expand the **Activity Guide** node.
3. Right-click the milestone.
4. Select **Edit**.
5. Click the **Browse** button, next to the **Icon Location** field.

   The Browse Icons dialog appears.
6. Select an icon from your file system.
7. Click **Open**.

   The icon path appears in the Edit Milestone dialog.
8. Click **OK**.

24.6.24 What Happens When You Configure an Icon for a Milestone

The activity guide tree uses this icon to identify the milestone nodes. If you do not specify an icon, then the milestone nodes do not display an icon.

24.6.25 How to Configure the Display Mode for a Guided Business Process

You can configure the display mode for a Guided Business Process to specify how to display the milestone and task links.

To configure the display mode for a Guided Business Process:

1. In the BPM Project Navigator, right-click the **Activity Guide** node.
2. Select **Edit**.
3. From the Display Mode list, select an option from the following:

Display Mode	Description
Always	Always display the milestone and task links for all the milestones in this Guided Business Process.
When Instantiated	Display 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.

4. Click **OK**.

24.6.26 What Happens When You Configure the Display Mode for a Guided Business Process

The milestones and tasks within the Guided Business Process use this configuration to display the milestone and tasks links. If the milestone and tasks are configured to used another configuration then the Guided Business Process configuration is ignored.

24.6.27 How to Configure the Display Mode for a Milestone

You can configure the display mode for a milestone, to specify how to display the milestone and tasks links.

1. In the BPM Project Navigator, right-click the **Activity Guide** node.
2. Select **Edit**.
3. From the Display Mode list, select an option from the following:

Display Mode	Description
Default	Use the Guided Business Process configuration.
Always	Always display the milestone link.
When Instantiated	Display the milestone link only when one or more of the user tasks in the milestone are instantiated. Use this mode for milestones located after a conditional gateway so that the activity guide tree does not display the milestone until the BPM Service Engine evaluates the condition.

4. Click **OK**.

24.6.28 What Happens When You Configure the Display Mode for a Milestone

The milestone links are displayed according to this configuration, regardless of the Guided Business Process configuration.

24.6.29 How to Configure the Display Mode for a User Task

You can configure the display mode for a user task to specify how to display the task link.

To configure the display mode for a user task:

1. In the BPM Project Navigator, select the **Activity Guide** node.
2. In the Structure window, expand the milestone that contains the user task.
3. Right-click the user task.
4. Select **Edit**.
5. From the Display Mode list, select an option from the following:

Display Mode	Description
Default	Use the milestone configuration.
Always	Always display the task link when the milestone that contains it is visible. If the user task is not instantiated, then the link is grayed out.
When Instantiated	Display tasks only when the user task is instantiated. Use this mode for user tasks located after a conditional gateway so that the activity guide tree does not display the user task until the BPM Service Engine evaluates the condition.

6. Click **OK**.

24.6.30 What Happens When You Configure the Display Mode for a User Task

The task links are displayed according to this configuration, regardless of the Guided Business Process configuration and the milestone configuration. The tasks links appear when the milestone is visible.

24.6.31 How to Configure the Task Access Mode for a Guided Business Process

You can configure the task access mode for a Guided Business Process to specify when to display the task links enabled.

To configure the task access mode for a Guided Business Process:

1. In the BPM Project Navigator, right-click the **Activity Guide** node.
2. Select **Edit**.
3. In the Task Access list select an option from the following:

Task Access Mode	Description
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.

4. Click **OK**.

24.6.32 What Happens When You Configure the Task Access Mode for a Guided Business Process

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 try to run the task.

24.6.33 How to Localize a BPMN Guided Business Process

You can localize a BPMN Guided Business Process so that the client can display it in different locales.

To localize a BPMN Guided Business Process:

1. In the BPM Project Navigator, select the **Activity Guide** node.
2. In the Structure window, right-click the **Activity Guide** node.
3. Select **Edit**.
4. From the Title list, select **Translation**.
5. Click the **Translation** icon, next to the title field.

   The Edit Translatable Strings dialog appears.
6. Click **Create Resource Bundle**.

   The Create Resource Bundle dialog appears.
7. Enter a name to identify the resource bundle.
8. Click **OK**.

   The Edit Translatable Strings dialog shows the resource bundle you created.
9. Click the **Add** icon next to the key list to add a new translation key.

   The Create a New Key dialog appears.
10. In the **Name** field, enter a name to identify the translation key.
11. In the Translatable Text field, enter the title.
12. Click **OK**.
13. From the Description list, select **Translation**.
14. Click the **Translation** icon, next to the description field.

    The Edit Translatable Strings dialog appears.
15. Click the **Add** icon next to the key list to add a new translation key.

The Create a New Key dialog appears.

16. In the **Name** field, enter a name to identify the translation key.
17. In the Translatable Text field, enter the description.
18. Click **OK**.
19. In the Edit Activity Guide dialog, click **OK**.
20. Localize the milestones that compose the Guided Business Process.

24.6.34 How to Localize a Milestone

You can localize a milestone so that the client can display it in different locales.

To localize a milestone:

1. In the BPM Project Navigator, select the **Activity Guide** node.
2. In the Structure window, right-click the **Activity Guide** node.
3. Select **Edit**.
4. From the Title list, select **Translation**.
5. Click the **Translation** icon, next to the title field.

   The Edit Translatable Strings dialog appears.
6. Click the **Add** icon next to the key list to add a new translation key.

   The Create a New Key dialog appears.
7. In the Name field, enter a name to identify the translation key.
8. In the Translatable Text field, enter the title.
9. Click **OK**.
10. From the Description list, select **Translation**.
11. Click the **Translation** icon, next to the description field.

    The Edit Translatable Strings dialog appears.
12. Click the **Add** icon next to the key list to add a new translation key.

    The Create a New Key dialog appears.
13. In the **Name** field, enter a name to identify the translation key.
14. In the Translatable Text field, enter the description.
15. Click **OK**.
16. In the Edit Activity Guide dialog, click **OK**.
17. If the milestone contains user tasks configured to display blocked icon and text, localize the user tasks that compose the milestone.

24.6.35 How to Localize a User Task

In a user task you can localize the following elements:

- Title
- Description
- Blocked Text

This procedure shows you how to localize the blocked text. You can also localize the title and description of the user task following the standard procedure for localizing flow objects.

To localize a user task:

1. In the BPM Project Navigator, select the **Activity Guide** node.
2. In the Structure window, expand the **Activity Guide** node.
3. If the milestone that contains the task you want to remove is collapsed, then expand the milestone.
4. Right-click the task.
5. Select **Edit**.
6. From the Display Block Icon and Text list, select translation.
7. Click the **Translation** icon, next to the title field.

   The Edit Translatable Strings dialog appears.
8. Click the **Add** icon next to the key list to add a new translation key.

   The Create a New Key dialog appears.
9. In the Name field, enter a name to identify the translation key.
10. In the Translatable Text field, enter the title.
11. Click **OK**.

24.6.36 What Happens When You Localize a Guided Business Process

The title and description of the Guided Business Process, milestones and tasks are displayed in the locale specified in the Guided Business Process client.

24.7 Configuring Activity Guide Properties

You can customize Activity Guides behavior by configuring their properties. To configure these properties you must edit the file `activityguide.properties`.

Table 24–1 shows the properties you can specify in this file.

Example 24–1 shows a typical `activityguide.properties` file:

Table 24–1 Activity Guide Properties

Property	Description	Possible Values
ServerAuthenticationMethod	Specifies the authentication mode used to connect to services deployed in the SOA Engine.	IDENTITY_PROPAGATION
ServerConnectionMode	Specifies the mode for the transmission of data.	▪ SOAP ▪ REMOTE
WorklistHttpURL	Only required when using digital signatures. Specifies the URL to access the worklist application.	http://*host:port*/integration/worklisapp
SelectionFilter	Specifies the filter used to filter the processes in an activity guide.	▪ MY ▪ PREVIOUS ▪ REPORTEES ▪ ADMIN

Table 24–1 (Cont.) Activity Guide Properties

Property	Description	Possible Values
AGDefinitionFilter	Specifies the definition ID used to filter the process in an activity guide. The activity guide only displays those processes that match this ID.	*activity guide definition ID*
AGInstanceOrdering	Specifies the order used to display the processes in the activity guide. For example: CREATION_DATE:ASC	■ *column_name*:ASC ■ *column_name*:DESC Default value: ASC
AGInstanceID	Specifies the instance ID used to display the activity guide tree. For example: 10001	*activity guide instance ID*
CustomPredicate1	Specifies an additional predicate to filter the list of processes in an activity guide. For example: CREATOR, EQ, jstein	*column name*, *operator*, *value*
CustomPredicate2	Specifies a different additional predicate to filter the list of processes in an activity guide. This predicate is used with CustomPredicate1	*column name*, *operator*, *value*
ShowAllAGTreeNodesProperties	Specifies if the activity guide shows a section at the top that describes the properties of activity guides, milestones and tasks.	■ true ■ false Default value: true
ShowRefreshButton	Specifies if the regional area displays a refresh button.	■ true ■ false Default value: false
AGTasksPopupTaskFlowID	Specifies the content to display in the task pop-up.	*fully qualified TaskFlow ID*
HideAGTreeRootNode	Hides the Guided Business Process title on the Activity Guide Tree root node.	■ true ■ false Default value: false
realm	Security abstraction that controls access to the resources served by the Java Web Server.	jazn.com
ShowCustomBlockedIcon	Specifies if the Guided Business Process shows the custom task blocked icon.	■ true ■ false Default value: false

Example 24–1 An `activityguide.properties` File

```
#ActivityGuide Properties
ServerAuthenticationMethod=IDENTITY_PROPAGATION
ServerConnectionMode=SOAP
SelectionFilter=MY
Realm=jazn.comShow
RefreshButton=true
#ShowAllAGTreeNodesProperties=true
## Sample value for AGDefinitionFilter:
default/BPMAGPrj2!2.0*31fcd931-6263-4b58-97cf-6fb084addabc
#AGDefinitionFilter=
#AGInstanceID=110003
#AGInstanceOrdering=CIKEY:DESC
#CustomPredicate1=STATE,EQ,OPEN
#CustomPredicate2=STATUS,EQ,In Progress
## Example Value for  AGTasksPopupTaskflowID is
/WEB-INF/ag-popup-task-flow.xml#ag-popup-task-flow
#AGTasksPopupTaskflowID=
#ShowCustomBlockedIcon=true
#HideAGTreeRootNode=false
```

```
##WorklistHttpURL is required only for digital signatures
#WorklistHttpURL=http://host:port/integration/worklistapp
```

24.8 Deploying an Guided Business Process to Oracle Weblogic Server

Guided Business Process are deployed to the application server in the same way as an SOA composite process. However, Guided Business Processes must be deployed to a standalone instance of Oracle WebLogic Server rather than the embedded Oracle WebLogic Server included with JDeveloper.

24.8.1 How to Deploy a Guided Business Process

Deploying a Guided Business Process to Oracle WebLogic Server involves the following main steps:

- Creating a connection to Oracle WebLogic Server
- Using JDeveloper or an Ant script to deploy the Guided Business Process

To deploy an Guided Business Process:

Following are the main steps in deploying an Guided Business Process:

1. Create a connection to Oracle WebLogic Server.
   a. Use connection type **Weblogic 10.3**.
   b. Enter a name for the WLS Domain.
2. Deploy the Guided Business Process through JDeveloper or using an Ant script:

   To deploy a Guided Business Process using JDeveloper:

   - Right-click the SOA composite associated with the Guided Business Process and select **Deploy**, then select the name of the SOA composite and the name of the server connection configured in the previous step.

   To deploy a Guided Business Process using an Ant script:

   - Right-click the SOA composite and select **Deploy**, the name of the SOA composite and **to JAR**.
   - Run the command shown in Example 24–2:

Example 24–2 Deploying a Guided Business Process Using an Ant Script

```
ant -f $ORACLE_HOME/bin/ant-sca-deploy.xml -DsarLocation <location of sca_
composite.jar> -DserverURL <soa server url> - Duser <administrator user name>
-Dpassword <administrator password>
```

For more information about deploying an SOA composite to the application server, see "Deploying SOA Applications with Enterprise Manager" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

24.8.2 What Happens When You Deploy a Guided Business Process to Oracle WebLogic Server

The Guided Business Process runs on WLS. You can view the Guided Business Process using Oracle Enterprise Manager Application Server Control console.

For more information about deploying SOA applications to WLS, see " Deploying SOA Applications with Enterprise Manager" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

24.9 Testing Guided Business Processes

You can create an instance of the deployed Guided Business Process in the Oracle Enterprise Manager Application Server Control console. This is useful for testing purposes.

To create a Guided Business Process instance:

1. In a Web browser, enter the URL of the Oracle Enterprise Manager Fusion Middleware Control Console as follows:

Example 24–3 Oracle Enterprise Manager Fusion Middleware Control Console URL

```
http://<hostname of Weblogic standalone server>:<port>/em
```

2. Browse for the application and click the SOA composite you created.
3. Select **Actions** > **Test service - client**.
4. Test the Guided Business Process by entering sample data and invoking the composite.
5. Refresh Fusion Middleware Control Console and verify that the SOA composite instance has been created. Check that the business process completed.

24.9.1 What Happens When You Create a Guided Business Process Instance

After you create an instance in a Guided Business Process, the Guided Business Process state changes to 'In Progress' and you can view the Activity Guide tree in the client application.

25

Building a Guided Business Process Client Application

This chapter explains how to build a client application to display your process instances using the milestones you defined when creating your Guided Business Process.

This chapter includes the following sections:

25.1 Introduction to Building a Guided Business Process Client Application

Guided Business Processes provide you with predefined ADF tasksflows that you can use to build an ADF application to display and run Guided Business Processes.

If the provided ADF taskflows do not satisfy your requirements, then you can use the set of APIs that Guided Business Processes provide, to obtain the information that your UI client applications displays. These APIs allow you to obtain data about the milestones and tasks using web services and Enterprise Java Beans.

25.2 Developing a Guided Business Process Client Application with Oracle ADF

A Guided Business Process client application provides a user interface for the Guided Business Process task flow. The client application can be developed in a simple ADF JSPX page in any configuration. Typically, a client application includes a region displaying the Activity Guide tree and another region displaying the details of the specific node selected from the tree.

One way to display these two regions is to include a dynamic region on the left side of a JSPX page and a human task flow on the right. However, any configuration is possible.

25.2.1 How to Develop a Guided Business Process Client Application

To develop a Guided Business Process client application:

1. In JDeveloper, create a new application.
2. Right-click the ViewController Project and then select **Project Properties**.
3. Select **Libraries and Classpath**.
4. Click **Add JAR/Directory**.

   A file browser dialog box opens.
5. Select the `oracle.bpm.activityguide-ui.jar` file located under `<JDEV_HOME>/jdeveloper/soa/modules`.
6. Click **Select**.
7. Add the run-time shared library references oracle.soa.bpel. and oracle.soa.workflow.wc to the weblogic-application.xml file by adding the following code:

```
<library-ref>
  <library-name>oracle.soa.bpel</library-name></library-ref><library-ref>
  <library-name>oracle.soa.workflow.wc</library-name>
</library-ref>
```

8. Create a new JSF Page (.jspx) in which to display the Activity Guide.
9. Drag and drop the following task flows onto the JSF Page (.jspx):
   - ag-tasktree-task-flow: for displaying the Activity Guide tree
   - ag-humantask-task-flow: for displaying the individual Activity Guide node

   Note: Dragging and dropping a task flow automatically creates a region for that task flow.
10. Create a file called `activityguide.properties`.

    For more information on Activity Guide properties, see Section 24.7, "Configuring Activity Guide Properties".

    If using identity propagation to secure the Activity Guide, then the properties `WorkflowAdminUser` and `WorkflowAdminPassword` are not required.
11. To enable a task flow popup with summary information, include the following properties in the `activityguide.properties` file:

    `AGTasksPopupTaskFlowID`: Use this parameter to display a task flow summary in ADF dynamic regions. Enter the relevant task flow ID.

    If this parameter is not set then the popup shows the value of `OutputText` as the default task summary.

    If you provide an invalid task flow region ID, then the Guided Business Process does not render the region and logs a message in the server log.

12. Configure the Activity Guide to display a refresh button in the Activity Guide tree., using the following alternative methods:

    - In the file `activityguide.properties`, add the parameter `ShowRefreshButton`. Set its value to true to enable the display of a refresh button, and false or any other value to disable the refresh button.
    - In the Activity Guide tree task flow, add the parameter `ShowRefreshButton` and set its value to true. This task flow parameter overrides the value of the parameter set in the `activityguide.properties` file.

      If the value of the `ShowRefreshButton` parameter is '`empty`' or '`null`', then the property `ShowRefreshButton` in the file `activityguide.properties` defines if the refresh button is shown. If the `activityguide.properties` file does not specify a value for this property then the refresh button is not shown in the client.

      Example 25–1 illustrates adding a `ShowRefreshButton` parameter to the tree task flow.

Example 25–1 Add the `ShowRefreshButton` Parameter to the Tree Task Flow

```
<taskFlow id="dynamicRegion1"
              taskFlowId="${backingBeanScope.dynamicLeft.dynamicTaskFlowId}"
              xmlns="http://xmlns.oracle.com/adf/controller/binding" >
   <parameters>
      <parameter id="ShowRefreshButton" value="true"
                    xmlns="http://xmlns.oracle.com/adfm/uimodel"/>
   </parameters>
</taskFlow>
```

13. Edit the file `adfc-config.xml` to include the location of the `activity.properties` file. This should be the absolute path to the activityguide.properties file.

    An example `adfc-config.xml` is shown in Example 25–2.

Example 25–2 `adfc-config.xml` File with Reference to `activityguide.properties` File

```
<managed-bean id="__10">
   <managed-bean-name id="__12">agProps</managed-bean-name>
   <managed-bean-class id="__
11">oracle.bpel.activityguide.ui.beans.model.AGProperties</managed-bean-class>
   <managed-bean-scope id="__9">session</managed-bean-scope>
   <managed-property id="__15">
    <property-name>agPropsFilePath</property-name>
    <property-class>java.lang.String</property-class>
    <value id="__14"><!-- relative path or absolute path should be given
here-->/activityguide.properties</value>
  </managed-property>
 </managed-bean>
```

14. Create a Workflow Service client configuration file. An example is shown in Example 25–3.

Example 25–3 Workflow Services Client Configuration File

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<workflowServicesClientConfiguration xmlns="http://xmlns.oracle.com/bpel/services/client">
   <server default="true" name="default">
```

```
    <localClient>
       <participateInClientTransaction>false</participateInClientTransaction>
    </localClient>
    <remoteClient>
       <serverURL>t3://host:port</serverURL>
       <initialContextFactory>weblogic.jndi.WLInitialContextFactory</initialContextFactory>
       <participateInClientTransaction>false</participateInClientTransaction>
    </remoteClient>
    <soapClient>
       <rootEndPointURL>http://host:port</rootEndPointURL>
       <identityPropagation mode="dynamic" type="saml">
          <policy-references>
             <policy-reference enabled="true" category="security"
              uri="oracle/wss10_saml_token_client_policy"/>
          </policy-references>
       </identityPropagation>
    </soapClient>
 </server>
```

25.2.2 What Happens When You Develop a Guided Business Process Application with Oracle ADF

A JDeveloper application with an ADF Web project is created. The application includes the following:

- JSF page with two regions, one for the Activity Guide tree and the other for Activity Guide node details.
- An `activityguide.properties` file.

25.2.3 What Happens at Run Time: How a Guided Business Process Application Is Developed with Oracle ADF

At run time, the Oracle ADF application displays the Guided Business Process developed at design time. A contextual event mechanism in the common ADF layer handles communication between the Activity Guide tree and Activity Guide node details, respectively.

When you select a Guided Business Process instance, the Activity Guide tree displays the information for the Activity Guides, milestones, and tasks in that Guided Business Process instance.

Alternatively, you can configure the AGInstanceID property in the activityguide.properties file for the JSF Page to render the following information for a particular Guided Business Process instance:

- Activity Guide
- Milestones
- Tasks

When selecting a milestone node in the Activity Guide tree, it retrieves or refreshes the sub-tree beneath the milestone.

When selecting a task node in the Activity Guide tree, it displays detailed task information for the task.

25.3 Securing the Guided Business Process Client Application

Securing the Guided Business Process client application ensures that only users with proper credentials can complete the tasks outlined in the Guided Business Process. Security features include authentication, authorization, realm verification and policy enforcement.

25.4 Localizing a Guided Business Process Client Application

If you localize a Guided Business Process client application then you can run the client in all the supported languages you defined.

If you want to localize a Guided Business Process application you must localize the following components when you design a Guided Business Process:

- AG Title
- AG Description
- Milestone Title
- Milestone Description
- Task blocked explanation text

The Guided Business Process automatically translates String that are part of the user interface, such as "display title" or Description. Guided Business Processes support the following locales:

- French
- German
- Italian
- Spanish
- Brazilian
- Japanese
- Korean
- Simplified Chinese
- Traditional Chinese
- Arabic
- Czech
- Danish
- Dutch
- Finnish
- Greek
- Hebrew
- Hungarian
- Norwegian
- Polish
- Portuguese

- Romanian
- Russian
- Slovak
- Swedish
- Thai
- Turkish

See Section 25.4.1, "How to Configure the Supported Locales for a Guided Business Process Client Application" for more information on how to localize a Guided Business Process.

25.4.1 How to Configure the Supported Locales for a Guided Business Process Client Application

Before configuring a Guided Business Process application to support additional locales, ensure that you provided the required bundles for those locales when developing the Guided Business Process.

To configure the supported locales for a Guided Business Process Client application:

1. Open the client application in Oracle JDeveloper.
2. Open the jspx client page.
3. Select **Source View** and modify the locale using the following code:

   ```
   <f:view locale= #{view.locale}>
   ```

4. Edit the `faces-config.xml` file located under `Project_Root /public_html/WEB-INF`.
5. Click the **Overview** tab in the editor window.
6. In the editor window, select **Application**.
7. In the **Locale Config** area, click **New** to open the Property Inspector to add the supported locales.
8. Add the supported locales.

   Example 25–4 shows how the `faces-config.xml` file looks after adding a set of supported locales.

Example 25–4 faces-config.xml file

```
<locale-config>
  <default-locale>en</default-locale>
  <supported-locale>ar</supported-locale>
  <supported-locale>ca</supported-locale>
  <supported-locale>cs</supported-locale>
  <supported-locale>da</supported-locale>
  <supported-locale>de</supported-locale>
  <supported-locale>zh_CN</supported-locale>
</locale-config>
```

9. Set the browser locale to a supported locale.
10. Run the client page.

25.5 Guided Business Process Run-time APIs

Guided Business Processes provide you a set of APIs that enable you to get details about the available milestones and the tasks that compose them. If the predefined Activity Guide ADF taskflows do not satisfy your requirements then you can use these APIs to obtain the information that you display in the client application.

25.5.1 Guided Business Process query Service API

This API is designed to support the following user navigation scenarios in an application displaying a Guided Business Process:

Display a list of Guided Business Process instances using a filter. Available filters are:

- `MY`: Guided Business Process instances containing active tasks assigned to the user.
- `REPORTEES`: Guided Business Process instances containing active tasks assigned to reportees to the current user.
- `PREVIOUS`: Guided Business Process instances containing completed tasks assigned to the user, and instances in which a particular task is reassigned to another user.
- `ADMIN`: Guided Business Process instances visible to the Guided Business Process administrator. Active instances can be assigned to any user.

Note: The BPMAGAdmin role maps to a user with an Administrator role assigned. This role enables the user to query for all the Guided Business Process instances available in the server, including completed, active, and instances with errors. The configuration file located in *$DOMAIN_HOME/config/fmwconfig/system-jazn-data.xml* contains the definition of this role.

Note: The Guided Business Process APIs enables retrieving detailed task information by providing the task ID, but do not retrieve the task information. Other APIs, such as the Workflow service APIs, are required for this purpose.

For more information about Workflow services, see "Introduction to Human Workflow Services" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Table 25–1 Guided Business Process Query Service API

Method	Description
`List queryAGDisplayInfos(String bpmProcessType,IWorkflowContext ctx,List agDisplayColumns,AGAssignmentFilter agAssignmentFilter,Predicate predicate,Ordering ordering,` `int startRow,int endRow)`	Returns a list of AGDisplayInfo objects with fields defined in `displayColumns`, meeting the criteria defined by `assignmentFilter` and `predicate`, in the order specified by `order`, and with row numbers between `startRow` and `endRow`. The `AGDisplayInfo` objects contain both metadata and run-time data of the Guided Business Process instances involved, which you can use to render the Guided Business Process instance views in Oracle BPEL Worklist or custom applications. You can assign the String parameter `bpmProcessType` the following values: ■ `IAGQueryService.AG_PROCESS_TYPE_BPM` ■ `IAGQueryService.AG_PROCESS_TYPE_BPEL` ■ `IAGQueryService.AG_PROCESS_TYPE_ANY` As milestones and tasks are not visible from the Activity Guide instance views, it is recommended not to include the `MILESTONE_STATE` column in `displayColumns` to avoid unnecessary performance overhead.
`AGDisplayInfo getAGDisplayInfoDetailsById(String bpmProcessType,IWorkflowContext ctx,long ciKey,List taskDisplayColumns,String agAssignmentFilter)`	Returns the `AGDisplayInfo` object specified by the Activity Guide instance ID. The `AGDisplayInfo` object returned includes both milestone and task information, which you can use to render the Activity Guide tree structure in the custom application. You can assign the String parameter `bpmProcessType` the following values: ■ `IAGQueryService.AG_PROCESS_TYPE_BPM` ■ `IAGQueryService.AG_PROCESS_TYPE_BPEL` ■ `IAGQueryService.AG_PROCESS_TYPE_ANY` The String parameter `agAssignmentFilter` enables specifying a filter for this method. The following `IAGQueryService` parameters can be used as values this parameter: ■ `IAGQueryService.AGASSIGNMENT_FILTER_MY` ■ `IAGQueryService.AGASSIGNMENT_FILTER_REPORTEES` ■ `IAGQueryService.AGASSIGNMENT_FILTER_PREVIOUS` ■ `IAGQueryService.AGASSIGNMENT_FILTER_ADMIN`

Table 25–1 (Cont.) Guided Business Process Query Service API

Method	Description
`MilestoneDisplayInfo getMilestoneDisplayInfo(String bpmProcessType,IWorkflowContext ctx,long cikey,String milestoneName,List taskDisplayColumns,String agAssignmentFilter)`	Returns the display information for a milestone in an Activity Guide instance. The `MilestoneDisplayInfo` object returned contains the metadata and run-time data of the specified milestone and the tasks in the milestone. You can use these to refresh the milestone sub-tree in a custom application when an end-user clicks the milestone. You can assign the String parameter `bpmProcessType` the following values: ■ `IAGQueryService.AG_PROCESS_TYPE_BPM` ■ `IAGQueryService.AG_PROCESS_TYPE_BPEL` ■ `IAGQueryService.AG_PROCESS_TYPE_ANY` The String parameter `agAssignmentFilter` enables specifying a filter for this method. The following `IAGQueryService` parameters can be used as values for this parameter: ■ `IAGQueryService.AGASSIGNMENT_FILTER_MY` ■ `IAGQueryService.AGASSIGNMENT_FILTER_REPORTEES` ■ `IAGQueryService.AGASSIGNMENT_FILTER_PREVIOUS` ■ `IAGQueryService.AGASSIGNMENT_FILTER_ADMIN`

25.5.2 JNDI Names for the Guided Business Process Enterprise Java Beans

The following table describes the JNDI names for the Guided Business Process Enterprise Java Beans.

Service Name	JNDI Names for the Enterprise JavaBeans
Activity Guide MetaData Store	`ejb/bpel/services/workflow/AGMetadataService`
Activity Guide Query Service	`ejb/bpel/services/workflow/AGQueryService`

25.6 Developing an Example of a User Interface for Guided Business Process Tasks Using Guided Business Process Run-Time Services

APIs enable accessing the Guided Business Process query and Metadata Services from within a custom application.

The following example illustrates the use of Java APIs to access Guided Business Process run-time services:

Example 25–5 Accessing the Guided Business Process Run-Time Service Using EJB

```
package client;
import com.oracle.bpel.activityguide.metadata.definition.model.AGDefinition;
import java.util.ArrayList;
import java.util.List;
import oracle.bpel.services.workflow.IWorkflowConstants;
import oracle.bpel.services.workflow.task.model.Task;
import oracle.bpel.services.workflow.verification.IWorkflowContext;
import oracle.bpel.services.workflow.client.IWorkflowServiceClient;
import oracle.bpel.services.workflow.client.IWorkflowServiceClientConstants;
```

```
import oracle.bpel.services.workflow.client.WorkflowServiceClientFactory;
import oracle.bpel.services.workflow.query.ITaskQueryService;
import oracle.bpel.services.workflow.query.impl.TaskQueryService;
import oracle.bpel.services.workflow.client.WorkflowServiceClientContext;
import oracle.bpel.services.workflow.metadata.config.ResourceBundleInfo;
import oracle.bpel.services.workflow.activityguide.query.IAGQueryService;
import oracle.bpel.services.workflow.activityguide.query.impl.AGQueryService;
import oracle.bpel.services.workflow.activityguide.query.model.AGDisplayInfo;
import
oracle.bpel.services.workflow.activityguide.query.model.MilestoneDisplayInfo;
import oracle.bpel.services.workflow.activityguide.metadata.IAGMetadataService;
import
oracle.bpel.services.workflow.activityguide.metadata.impl.AGMetadataService;
import sun.security.util.Password;
public class AGServiceSampleCode {

    private static String USERNAME = "jcooper";
    private static String PASSWORD = "welcome1";
    private static String REALM = "jazn.com";
    private static IWorkflowServiceClient wfSvcClient;
    private static IWorkflowContext sJCooperCtx;

    public static void main(String[] args)
    {
        try {

    testSetUp();

    // GetAGDefinition API requires an AG instance as input, which is not easily
accessible in customer's env.
    // As a result, the sample code for invoking this API is not provided.
    //testGetAGDefinition();

    testGetAGDefinitionById();
    testGetAGResourceBundleInfo();

    testQueryAGDisplayInfos();
    testQueryAGDisplayInfoDetailsById();
    testQueryAGMilestoneDisplayInfo();

        } catch (Exception e) {

            e.printStackTrace();
        }
    }

    private static void testSetUp()
throws Exception
    {
wfSvcClient =

WorkflowServiceClientFactory.getWorkflowServiceClient(WorkflowServiceClientFactory
.REMOTE_CLIENT);

sJCooperCtx =
    wfSvcClient.getTaskQueryService().authenticate(USERNAME, PASSWORD, REALM,
null);
    }

    private static void testGetAGDefinitionById()
```

```
throws Exception
    {
String agDefinitionId =
"HelpDeskRequestSCAApp/HelpDeskRequestComposite!1.0*2007-10-22_13-32-50_
536//HelpDeskRequestProcess//HelpDeskRequestProcess.ag"; // Need to supply a valid
AG definition id here

AGDefinition agDefinition =
    wfSvcClient.getAGMetadataService().getAGDefinitionById (sJCooperCtx,
agDefinitionId);

if (agDefinition != null)
{
    System.out.println("ag def obtained");
    System.out.println("ag def name: " + agDefinition.getName());
    System.out.println("ag def milestone display mode: " +
agDefinition.getMilestoneDisplayMode());
}
    }

    private static void testGetAGResourceBundleInfo()
throws Exception
    {
String agDefinitionId =
"HelpDeskRequestSCAApp/HelpDeskRequestComposite!1.0*2007-10-22_13-32-50_
536//HelpDeskRequestProcess//HelpDeskRequestProcess.ag"; // Need to supply a valid
AG definition id here

ResourceBundleInfo resourceBundleInfo =
wfSvcClient.getAGMetadataService().getResourceBundleInfo(sJCooperCtx,
agDefinitionId, sJCooperCtx.getLocale());

System.out.println("bundle name: " + resourceBundleInfo.getName());
    }

    private static void testQueryAGDisplayInfos()
throws Exception
    {
List agQueryColumns = new ArrayList();
agQueryColumns.add("MILESTONE_STATE");
agQueryColumns.add("DEFINITION_ID");

// Query for all AG instances belonging to user jcooper
List agDisplayInfoList =
      wfSvcClient.getAGQueryService().queryAGDisplayInfos(sJCooperCtx,
  agQueryColumns,
  IAGQueryService.AGAssignmentFilter.MY,
  null, //agPredicate,
  null, //ordering,
  0,
  0);

System.out.println("ag display info list size:" +  agDisplayInfoList.size());
if ( agDisplayInfoList.size() > 0 )
{
    AGDisplayInfo agDisplayInfo =  (AGDisplayInfo) agDisplayInfoList.get(0);
    System.out.println("AG title:" +  agDisplayInfo.getTitle());
    System.out.println("milestone display info list size:" +
agDisplayInfo.getMilestoneDisplayInfo().size());
    for (int i=0; i< agDisplayInfo.getMilestoneDisplayInfo().size(); i++)
```

```
    {
System.out.println("i = " + i + " milestone display info:" +
((MilestoneDisplayInfo)
agDisplayInfo.getMilestoneDisplayInfo().get(i)).getMilestoneInstance().getName());
    }
}
    }

    private static void testQueryAGDisplayInfoDetailsById()
throws Exception
    {
long cikey = 1; // Need to supply a valid AG cikey here

List taskQueryColumns = new ArrayList();
taskQueryColumns.add("TASKID");
taskQueryColumns.add("TITLE");
taskQueryColumns.add("OUTCOME");

AGDisplayInfo agDisplayInfo =
      wfSvcClient.getAGQueryService().getAGDisplayInfoDetailsById (sJCooperCtx,
cikey, taskQueryColumns);
System.out.println("AG display info status:" +
agDisplayInfo.getAGInstanceInfo().getStatus());
System.out.println("AG display info bpel status:" +
agDisplayInfo.getAGInstanceInfo().getBpelStatus());
    }

    private static void testQueryAGMilestoneDisplayInfo()
throws Exception
    {
long cikey = 1; // Need to supply a valid AG cikey here
String milestoneName = "ApprovePricing"; //  Need to supply a valid AG milestone
name here

List taskQueryColumns = new ArrayList();
taskQueryColumns.add("TASKID");
taskQueryColumns.add("TITLE");
taskQueryColumns.add("OUTCOME");

MilestoneDisplayInfo milestoneDisplayInfo = null;

milestoneDisplayInfo =
    wfSvcClient.getAGQueryService().getMilestoneDisplayInfo (sJCooperCtx, cikey,
milestoneName, taskQueryColumns);

System.out.println("milestone display info name:" +
milestoneDisplayInfo.getMilestoneInstance().getName());
System.out.println("milestone display info title:" +
milestoneDisplayInfo.getTitle());
System.out.println("milestone display info task list size:" +
milestoneDisplayInfo.getTask().size());
    }

}
```

For more information regarding the EJB and Web Service APIS, see the Javadoc.

25.7 Using Guided Business Process Logging

Guided Business Processes use a log file to store information about the different operations they perform. This file contains log messages that track the application behavior and possible errors that might occur while running the application.

You can use the information in this log file to find out the cause of an unexpected behavior in your application.

The importance of the log messages varies according to their level. The level of the messages used for debugging purposes is different to the level of the messages that contain warnings or errors.

You can configure Guided Business Process Logging to log only certain level of messages according to your needs.

25.7.1 How to Enable Client Side Logging

You can configure Guided Business Processes to generate a log file on the client side.

To enable client side logging:

1. Locate the logging.xml file in the directory `<DOMAIN_HOME>/config/fmwconfig/servers/`*Server Name*
2. Open the `logging.xml` file for editing.
3. Add the following entry in the `<loggers>` element:

```
<logger name="oracle.bpel.activityguide.ui" level="NOTIFICATION:1"
useParentHandlers='false'>
<handler name="odl-handler"/>
</logger>
```

4. Save the changes.
5. Re-start Web Logic Server.

25.7.2 How to Enable Server-Side Logging

You can configure Guided Business Processes to generate a log on the server side.

To enable server-side logging:

1. Locate the logging.xml file in the directory `<DOMAIN_HOME>/config/fmwconfig/servers/`*Server Name*
2. Open the `logging.xml` file for editing.
3. Add the following entry to the `<loggers>` element:

```
<logger name="oracle.bpm.services.activityguide.query"
level="NOTIFICATION:1" useParentHandlers='true'>
```

4. Save the changes.
5. Re-start Web Logic Server.

25.7.3 Configuring Log Levels

Log messages contain a level that identifies the severity of the problem.

Table 28-3 shows the available log levels. The Severity column describes the common term used to identify a certain severity. The Log Level Value common specifies the value that you must use in the logging.xml file.

Table 25–2 Log Level Values

Severity	Log Level Value	Description
Fatal	INCIDENT_ERROR:1	Indicates a serious problem caused by unknown reasons. Users cannot fix the problem by themselves, they must contact Oracle Support.
Severe	ERROR:1	Indicates a serious problem that requires immediate attention from the System Administrator
Warning	WARNING:1	Indicates a potential problem. The System Administrator should review these log messages.
Information	NOTIFICATION:1	Indicates a major lifecycle event such as the activation or deactivation of a primary sub-component or feature.
Configuration	NOTIFICATION:16	Specifies a normal event occurred at a lower level.
Fine	TRACE:1	Specifies trace or debug information for events that are meaningful to end users of the product, such as public API entry/exit points.
Finer	TRACE:16	Specifies a detailed trace or debug information that can help Oracle Support diagnose problems with a particular subsystem.

You can configure Guided Business Process Logging to specify the level of detail of the information stored in the Guided Business Process logs.

To set the log level must change the value of the attribute level in the logger element in the logging.xml file.

When you set the log level to a certain severity, all the messages that correspond to higher severities are also stored. For example, if you set the log level to severe, then the log messages of severity fatal are also logged.

25.7.4 How to View Guided Business Process Log Messages

Log messages are stored in the following file: `<DOMAIN HOME>/servers/<Server Name>/logs/`*Server Name*`-diagnostic.log`

You can view the file that contains the log messages using a text editor.

25.7.5 Understanding Guided Business Process Log Messages

A log message contains information that helps you identify the problems in your Guided Business Process application.

Table 28-3 describes the items that compose a log message.

Table 25–3 Log Message Items

Log Message Item	Description
Date and Time	Specifies the date and time when this log message was generated.
Message Type	Specifies the severity of the message.

Table 25–3 (Cont.) Log Message Items

Log Message Item	Description
Execution Context ID (ECID)	A global unique identifier and a sequence number that correspond to the thread where the originating component is running. You can use it to correlate messages from multiple components that may be involved in the same thread.
Application Name	Specifies the name of the application that generated the log message.
Class Package Name	Specifies the package of the class that generated the log message.
Message ID	Specifies a short identifier that uniquely identifies the message.
Message Text	Describes the event. This message is localized, thus it displays in the language that corresponds to the locale of your system.

When you read a log file you must look for the message text, this text describes what happened. The message type helps you identify how serious the problem is. For more information about the different message types, see Table 25–2.

You can use the date and time of the log message to identify the action that caused the problem.

> **Note:** before contacting Oracle Support make sure you can provide them the message ID and the execution context ID.

Example 25–6 Log Message Example

```
DefaultServer-diagnostic.log:[2009-07-10T17:39:35.220-07:00] [DefaultServer]
[NOTIFICATION] [AGU-12605] [oracle.bpel.activityguide.ui.beans] [tid:
[ACTIVE].ExecuteThread: '1' for queue: 'weblogic.kernel.Default (self-tuning)']
[userId: jstein] [ecid: 0000I9bG2R3DScQ6ube9UH1ALxd1000007,0] [APP:
AGNonUIshellApp#V2.0] [arg: jstein] Setting user, jstein as the loginUserId in
server interfacing bean.[[
```

Example 28-12 shows a notification log message that contains information about a loginUserId change. In this example the different log message items are:

- **Date and Time:** 2009-07-10T17:39:35.220-07:00
- **Message Type:** NOTIFICATION
- **Execution Context ID:** ecid: 0000I9bG2R3DScQ6ube9UH1ALxd1000007,0
- **Application Name:** APP: AGNonUIshellApp#V2.0
- **Class Package Name:** oracle.bpel.activityguide.ui.beans
- **Message ID:** AGU-12605
- **Message Text:** Setting user, jstein as the loginUserId in server interfacing bean.

26

Using Approval Management

This chapter describes the Approval Management extensions (AMX) to the human workflow services of Oracle SOA Suite. The human workflow service is responsible for handling all interactions with users or groups participating in the business process. It does 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, email, portals, or custom applications.

AMX enables you to define complex task routing slips for human workflow by taking into account business documents and associated rules to determine the approval hierarchy for a work item. Additionally, AMX lets you define multi-stage approvals with associated list builders based on supervisor or position hierarchies. You define the approval task in the Human Task Editor of Oracle JDeveloper, and associate the task with a BPEL process.

For more information about human tasks, see the following chapter and appendix in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*:

- "Designing Human Tasks"
- "Understanding Human Task Services"

This chapter includes the following sections:

26.1 Introduction to Approval Management

Approval Management extensions extend human workflow services with complex approval patterns. It serves as a sophisticated "Assignment Manager" for human workflow. Some of the key workflow features that are provided include:

- Declarative modeling of approval management processes
- The ability to define complex multi-stage approval with static and dynamic approval list

- A Workflow Editor to define task parameters, assignment and routing policies, escalation and expiration settings, and notification settings
- Policy-based task assignment, which allows users to define approval rules based on business documents
- The ability to design a task form to render contents of the approval task and associated task operations
- The ability to define email, voice, and Instant Messaging notifications for various participants in the workflow
- A web-based worklist application for task assignees, process owners, and administrators
- The ability to look up users and roles in various user directories, including Oracle Internet Directory, LDAP, and third-party directories

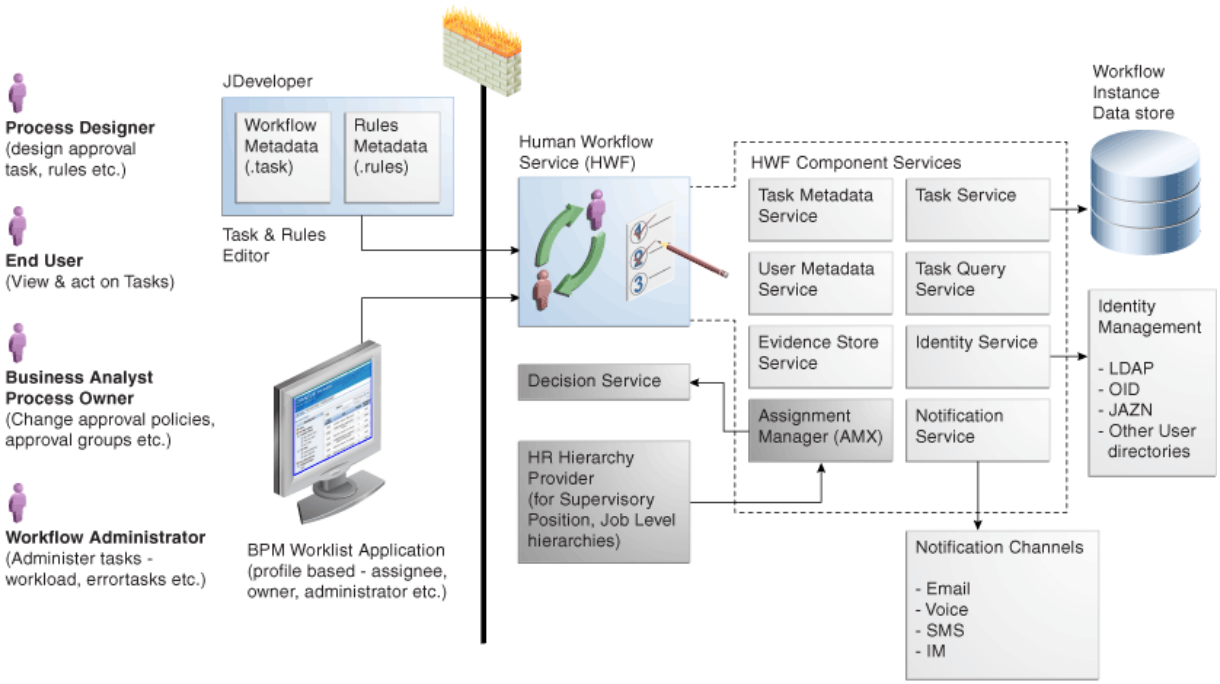
AMX provides the following additional features:

- Attributes derived from ADF view object in transactional applications
- The ability to retrieve various job, position, and supervisory hierarchies from HR systems using hierarchy provider plug-ins.
- The ability to define rules for controlling approval lists and hierarchy configurations

26.1.1 AMX Components

Figure 26–1 shows the key AMX and human task integration components. These components are described in subsequent sections of this chapter.

Figure 26–1 Overall Architecture



The human workflow service enables users to model human interactions as part of a business process. The human workflow service handles requests based on task and rules metadata. It consists of the following set of core services:

- Task service
- Task query service
- User metadata service
- Task metadata service
- Identity service
- Notification service
- Assignment manager

These services are described in detail in the chapter "Designing Task Display Forms for Human Tasks" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*. AMX serves as a sophisticated assignment manager within human workflow allowing you to model complex approval patterns based on business rules.

The core components required for approval management include the following:

- **Human Task Editor in JDeveloper**

  This task editor is used to define the metadata for a human task and the routing slip. The task editor lets you define such things as task parameters, outcomes, expiration and escalation, and notification settings. Some of the components added by AMX include the ability to do the following:

  – Define multi-stage approvals and associated approval list builders in JDeveloper.

  – Determine the approval hierarchy based on business documents (ADF objects) and business rules. This is done through Rules Designer in JDeveloper

- **Human workflow services**

  Some of the key services that are required for handling complex approvals include the following:

  – Task Service - Responsible for creating and managing tasks in the dehydration store

  – Identity Service - Responsible for authentication and authorization of users and groups. The service can look up various user directories for authorization and contact information for users.

  – Task Query Service - Responsible for retrieving tasks for the web-based worklist application

  – Decision Service - Responsible for executing business rules related to approvals

- **Worklist Application**

  The worklist is a web-based application that lets users access tasks assigned to them and perform actions based on their roles in the approval process. The worklist supports the following profiles:

  – Work assignee - An end user who is assigned a task. These users can view tasks assigned to them and perform actions, and also can define custom views and define routing rules for their tasks.

- Process owner - Typically a business analyst responsible for managing certain types of approvals. These users can manage tasks for the processes they own, define approval groups, and change approval policies
- Workflow administrator - Typically a system administrator responsible for managing errored tasks, and administering and monitoring work queues. This user also may use Oracle Enterprise Manager to monitor the health of the workflow services.

26.2 Understanding Approval Management Concepts

As described earlier, AMX extends human workflow services with additional functionality to handle complex approval patterns.

Some human workflow concepts with which you must be familiar are the following:

- Human Task Editor in JDeveloper
- Task metadata (task parameters, allowed operations, and patterns) and routing slip
- ADF task flow based on task forms
- Worklist application

These concepts are described in the following chapters in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*:

- "Designing Human Tasks"
- "Designing Task Display Forms for Human Tasks"
- "Using Oracle BPEL Worklist Application

The sections that follow describe new concepts introduced to handle complex approvals.

26.2.1 Task

A task handles approvals. A different task is created for each approval requirement based on the business served by it. For example, an approve new expense report task or an approve new purchase order task.

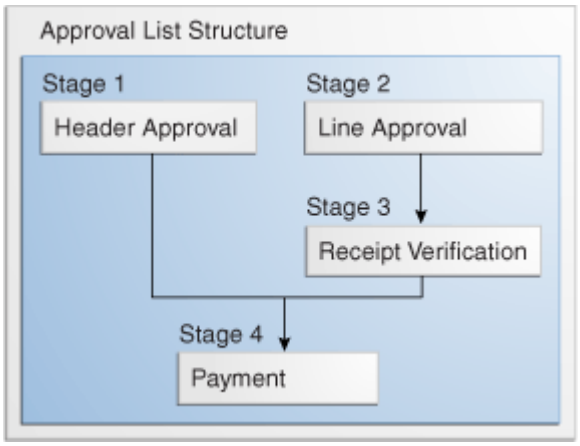
Some of the standard metadata for a task include the following:

- Task attributes such as title, outcomes (approve, reject, and so on) priority, expiration and others
- Task parameters that may be based on simple primitive types, XML elements, or external entities such as ADF view objects
- A complex approval task that may include one or more stages to identify the key milestones within the approval sequence. For more information see Section 26.2.3, "Stages."
- Expiration and escalation policy
- Notification settings for notifying various participants
- *List builders* within stages, which are based on names and expression, management chain, supervisory, position, job-level hierarchy, or approval groups. For more information, see Section 26.2.4, "List Builders."

- *Approval task configurations*, including policies for substitution and modification of approvers, configuration of self-approval, and repeated approvers. For more information, see Section 26.2.1, "Task."

  Figure 26–2 shows the various stages in a sample approval pattern.

Figure 26–2 Approval List Structure



These are described in more detail in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

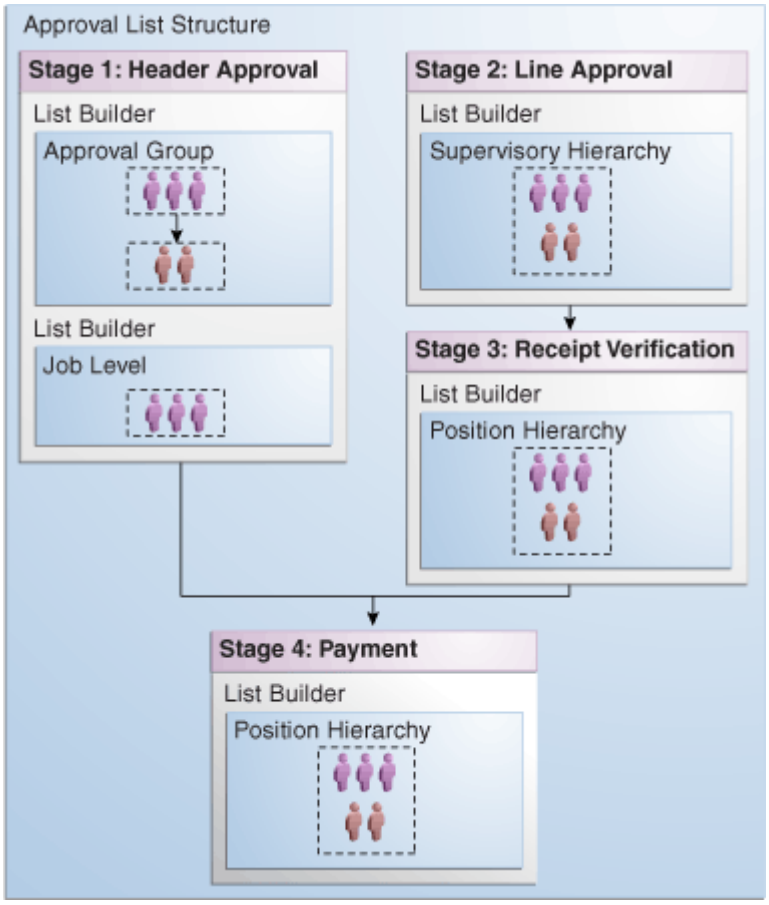
The approval pattern consists of four stages:

- Header approval
- Line approval
- Receipt verification
- Payment

Header approval runs in parallel with line approval and receipt verification. After these stages run, the payment stage runs.

Each of the four stages has list builders. Multiple list builders in a stage can run in serial or parallel to one another. One or more approvers can exist within each list builder. Figure 26–3 illustrates these concepts.

Figure 26–3 Stages and Their List Builders



These concepts are described in the sections that follow.

26.2.2 Service Data Objects

ADF Business Components objects can be exposed easily as Service Data Objects (SDOs) through the service interface. This provides a flexible way to accept business entities. Subsequently, supporting SDOs natively, enables us to accept multiple business entities. This also lays the foundation for future Flexfield SDO support. Since an SDO is a structured XML, you can pass it in as static XML through the task payload.

A collection is defined in an entity parameter for the task. It enables access to a portion of the business entity as an XML fragment retrieved by an XPATH expression. Keys allow us to identify the primary keys in this fragment.

An entity parameter is the definition that tells us how to access an SDO or a static XML. An entity parameter captures the following information for an SDO:

- Identity of a reference in the overall SCA process, including the Web service definition language (WSDL) for the SDO web service
- Method to invoke
- Input message to the web service
- Output message to the web service
- Collections

An entity parameter captures the following information for a static XML:

- XSD for the static XML
- Collections

For example, an expense voucher can have hierarchical groupings of header, lines, and cost centers. For approval policy purposes, you may only define a collection on header and lines if these are the only components required for determining the set approvers. It is not necessary to map as collections those parts of the business document that are not necessary to define rules.

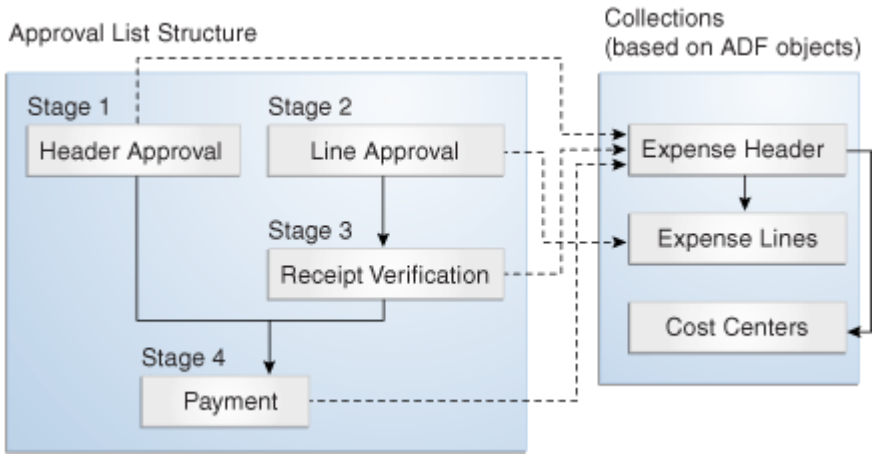
For more information, see "Implementing Business Services with Application Modules" and "Integrating Service-Enabled Application Modules" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

26.2.3 Stages

A stage is a set of approvals related to a collection. The same collection can be associated with multiple approval stages.

Figure 26–4 illustrates the mapping of stages and collections.

Figure 26–4 Mapping of Stages and Collections



Each approval stage is associated with a collection. In Figure 26–4, there are four stages in the approval.

- **Header Approval** is associated with the Expense Header collection.
- **Receipt Verification** is associated with the Expense Header collection.
- **Payment** is associated with the Expense Header collection.
- **Line Approval** is associated with the Expense Lines collection.

A compound approval may consist of multiple stages and then can be modeled in serial or parallel with each other. Each stage consists of list builders to determine the list of approvers.

Optionally, each list builder can be associated with an approval policy, that is, a set of rules. At run time, the appropriate set of approvals are returned based on the list builders used within the stage and on the associated policies.

26.2.4 List Builders

As described in Section 26.2.3, each approval stage consists of list builders to determine the actual list of approvers. The following list builders are supported.

- **Names and Expressions**

  Enables you to construct a list using static names, or names coming from XPath expressions.

- **Approval Groups**

  Includes predefined approver groups in the approver list. Approval groups can be static or dynamic.

- **Job Level**

  Ascends the supervisory hierarchy, starting at a given approver and continuing until an approver with a sufficient job level is found.

- **Position**

  Ascends the position hierarchy, starting at a given approver's position and continuing until a position with a sufficient job level is found.

- **Supervisory**

  Ascends the primary supervisory hierarchy, starting at the requester or at a given approver, and generates a chain that has a fixed number of approvers in it.

- **Management Chain**

  Enables you to construct a list based on management relationships in the corresponding user directory.

  The management chain participant type only supports parallel routing when the first assignee in the management chain is a single user. You cannot specify parallel participants such as a set of users or a group, as the initial assignees in the management chain.

- **Rule-based**

  Enables you to model rules that return different list-builder types based on different conditions. For example, if you model a supervisory list builder with rules, the rule can return only the supervisory list builder. If you model a rule-based list builder, the rule can return different list-builder types.

Note: The Approval Groups, Job Level, Position, and Supervisory list builders are specific to AMX, and are described in detail in Section 26.3.6.3, "How to Model and Configure List Builders."

For information about the Names and Expressions, Management Chain, and Rule-based list builders, see "Creating a Single Task Participant List" in the "How to Assign Task Participants" section in the "Creating the Human Task Definition with the Human Task Editor" chapter in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

26.2.5 Task Operations

Most of the standard human task operations also are available on AMX-based tasks. Some of the common operations include the following:

- **User-defined outcomes** - Business outcomes, such as "Approve" and Reject," that are associated with a task. When a user performs these types of actions, the task is removed from the user's "Inbox" and is marked as completed or moved to the next approver.
- **Delegate** - Allows a user to assign a task to another person or role to act on his or her behalf.
- **Escalate** - Allows a user or an administrator to escalate a task to the user's supervisor.
- **Reassign** - Allows users to transfer a task to another user. From that point on, the new user's hierarchy is used for supervisor or other organization-based approvals.
- **Withdraw** - Allows the task initiator or administrator to cancel or withdraw the task after the approval has started.
- **Request for Information** - Allows a task approver to request information from any prior participant or the task initiator.
- **Pushback** - Allows the task approver to push back the task to the previous approver to review it again.
- **Adhoc Insertions** - Allows any task assignee to insert approvers in the generated approval list.

> **Note:** The position list builder does not allow the approver to delegate, escalate or perform adhoc insertions.

See the section "Acting on Tasks: The Task Details Page," in the chapter "Using Oracle BPEL Worklist Application" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite* for a complete list of actions.

26.2.6 Business Rules for Approval

Approvers of a task can be defined either inline in a task definition or by using business rules to specify the list builders that identify the actual approvers of a task. In addition, you can use business rules to specify approver substitution and list modifications. These rules are defined with the help of Oracle Business Rules and can vary between organizations. Typically, however, they are defined by the customer.

Business rules are a combination of conditions and actions. Optionally, priority and validity periods can be defined for these rules. In Human Workflow rules, rule conditions are defined using fact types that correspond to the task, and to the task message and entity attributes (which are XML representations of SDO objects). Rule actions consist of approver list builders and their parameters. Approver list builders move up a particular hierarchy and construct or modify the approver list according to the parameters defined. Approver list builders are implemented as XML (JAXB) fact types.

For more information about these concepts, see the chapter "Overview of Oracle Business Rules" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

26.2.6.1 List Creation

A list creation policy includes rule conditions and actions that create the list builders.

The following example rules illustrate the configuration of the Supervisory list-builder parameters that create an approver list based on an SDO-based fact type.

Example 26–1 Rule 1

```
IF
ExpenseItems.ReceiptAmount < 200
THEN
call CreateSupervisoryList( levels:1,
startingPoint:HierarchyBuilder.getPrinicipal("jstein",-1,"",""),
uptoApprover:HierarchyBuilder.getPrinicipal("wfaulk",-1,"",""),
autoActionEnabled:false,autoAction:null,
responseType:ResponseType.REQUIRED,ruleName:"Rule_1",lists:Lists)
```

Example 26–2 Rule 2

```
IF
xpenseItems.ReceiptAmount >= 200
THEN
call CreateSupervisoryList( levels:1,
startingPoint:HierarchyBuilder.getPrinicipal("wfaulk",-1,"",""),
uptoApprover:HierarchyBuilder.getPrinicipal("cdickens",-1,"",""),
autoActionEnabled:false,autoAction:null,
responseType:ResponseType.REQUIRED,ruleName:"Rule_2",lists:Lists)
```

For more information, see Section 26.3.6.4.1, "How to Create Lists."

26.2.6.2 Approver Substitution

Users, groups, and application roles appearing in a list can be substituted using list substitution. List substitution is available from Rules Designer and does not require any configuration in JDeveloper.

The following example rule illustrates approver-substitution usage.

Example 26–3 Approver-Substitution Usage

```
IF
ExpenseItems.ReceiptAmount < new BigDecimal(4000)
THEN
call Substitute(fromId:"jcooper", toId:"jstein", ruleName:"Substituted",
substitutionRules: SubstitutionRules)
```

This rule implies that if the expense item amount is less than 4000, then substitute approver "jcooper," if present in the approver list, with approver "jstein."

For more information, see Section 26.3.6.4.2, "How to Make Approver Substitutions."

26.2.6.3 List Modification

Job Level and Position lists can be extended or truncated from rules. List modification is applied after list creation.

The following example rule illustrates list-modification usage.

Example 26–4 List-Modification Usage

```
IF
ExpenseItems.ReceiptAmount > new BigDecimal(3000)
THEN
Call Extend(ifFinalApproverLevel:3, extendBy:2,ruleName:"Modified",lists:Lists)
```

This rule implies that if the expense item amount is greater than 3000, and if the final approver in the approver list is of Job Level 3, then extend the approver list by at least two relative levels.

For more information, see Section 26.3.6.4.3, "How to Make List Modifications."

26.3 Designing Approval Management Tasks in Oracle JDeveloper

You design approval management tasks by defining a human task that provides the ability to model multi-stage approvals and determine the appropriate approvers based on approval policies for a business object and the associated HR hierarchy provider.

This section describes the overall modeling process and the specifics of the process you use to model approval management tasks in JDeveloper.

26.3.1 Introduction to the Modeling Process

The modeling process for designing approval management tasks includes the following:

- Creating a human task definition
- Creating a task display form using the Human Task Editor

Creating a human task definition includes the following tasks:

- Specifying general information, such as task title and task-title globalization, outcomes, priority, owner, and category
- Defining task parameters, including those with service data object (SDO) references
- Specifying mapped attributes
- Modeling task routing by specifying stages and list builders, and modeling any business rules that define the list builders
- Defining escalation and renewal policies
- Specifying notification settings
- Modeling any advanced settings like callbacks, security access rules, and restricted assignment

Some of these procedures are discussed in the sections that follow. For information about those that are not discussed, see "Creating the Human Task Definition with the Human Task Editor" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

You also must create a task display using an ADF task flow to display the details of the approval. ADF task flows are used to model the user interface for the task details page. For more information, see "Designing Task Display Forms for Human Tasks" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

26.3.2 Before You Begin

Before designing approval management tasks, you must satisfy the following prerequisites:

- You must have deployed SDO services.
- You must have created a human task service component in which to design the approval task.

26.3.3 Specifying General Information

Some general information, including task title, outcomes, priority, owner, and category, is not specific to AMX.

For more information about these, see "How to Specify the Task Title, Priority, Outcome, and Owner" in the section "Creating the Human Task Definition with the Human Task Editor" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

26.3.3.1 Task-Title Globalization

The title attribute of the task object contains a user-friendly value that mainly is descriptive in nature. In AMX, the task title can be globalized so that it renders in the user's preferred language.

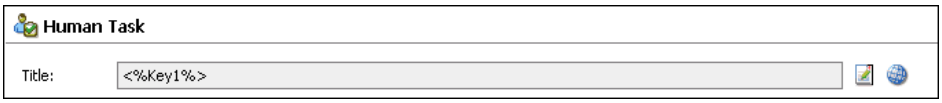
Title is defined in the `*.task` file for design time and in the `WorkflowTask.xsd` file for run time. Currently, the definition of these elements in both of these files are simple xsd:string types. For globalization, the structure and usage of these elements change to accommodate a mechanism that provides translatable, formatted strings.

The design-time metadata for these elements is enhanced to contain a value element and an optional set of parameters. Messages defined as an XPath expression or static have their information stored in the value element and require no parameters. Messages defined that rely on information in a resource bundle have a key stored in the value element with some parameters also defined.

The Human Task Editor provides a mechanism in the Expression Builder to enable the user to specify the resource key and parameters and, at the same time, generate the appropriate design time XML in the taskDefinition.

Figure 26–5 shows the globalization icon in the Human Task Editor.

Figure 26–5 Title Globalization Icon

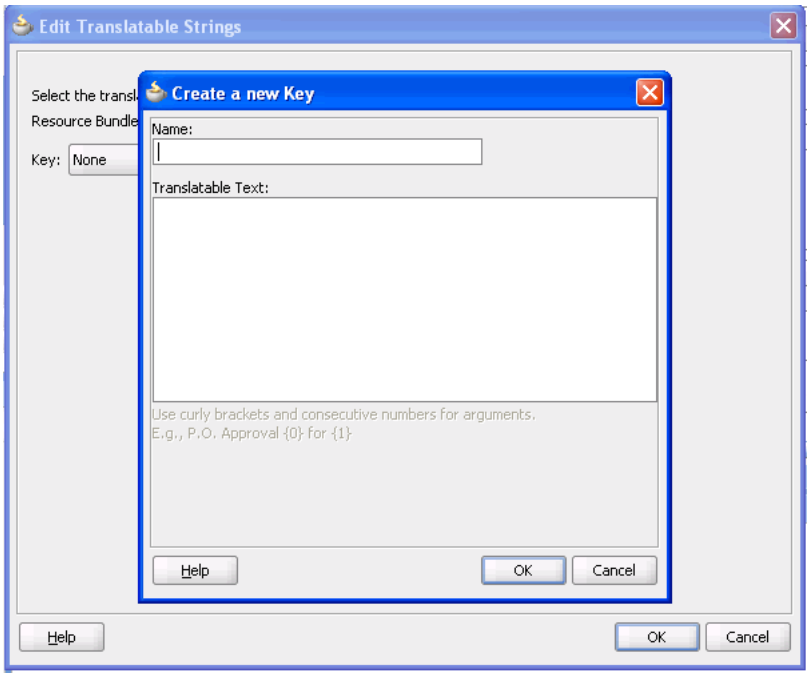


The following procedure explains how to add translatable strings. It assumes that a resource bundle has been specified.

1. Click the icon to display the Edit Translatable Strings dialog.
2. Select a key from the dropdown list or click the plus sign (+) to create one.

   Figure 26–6 shows the Create a New Key dialog, which displays when the plus sign (+) on the Edit Translatable Strings dialog is clicked.

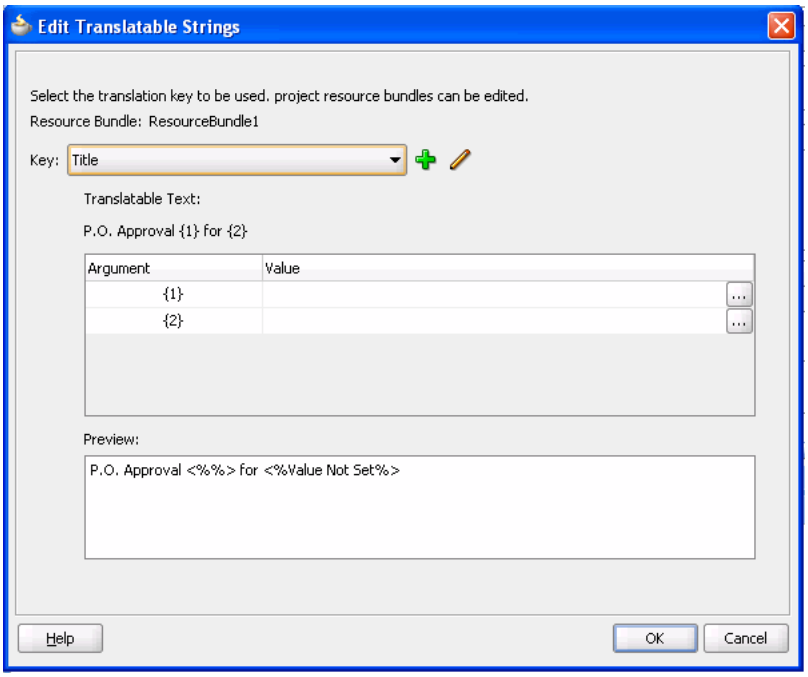
Figure 26–6 Create a New Key Dialog



3. Enter a name, the translatable text, and click **OK**.

   Figure 26–7 shows the Edit Translatable Strings dialog after a new key has been added.

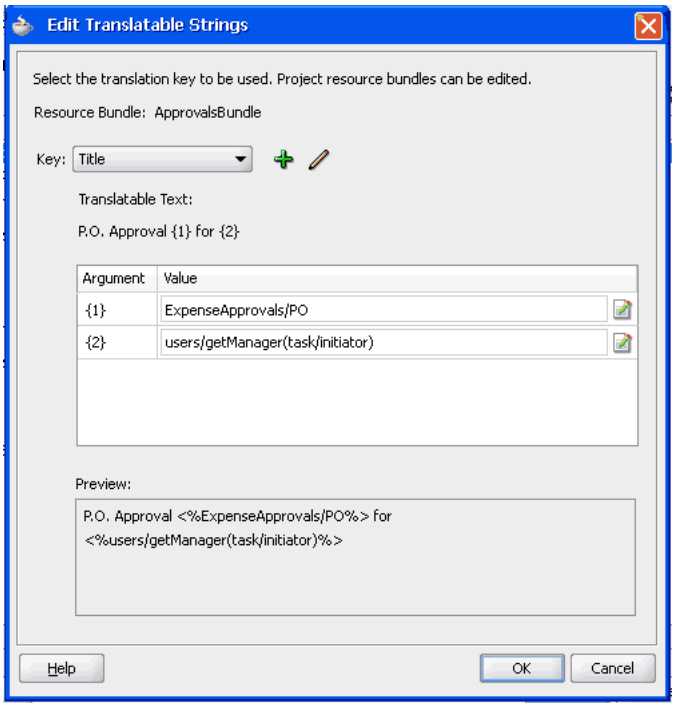
Figure 26–7 New Key Added



4. Use the Expression Builder to add values.

   Figure 26–8 shows the completed Edit Translatable Strings dialog.

Figure 26–8 Translatable Text and Values



Note: The title value, or a definition of the title value can be set in two places: in the TaskDefinition XML (`.task`) file, or in the bpel file. When set in the bpel file, this value takes precedence over the definition in the TaskDefinition. However, the value in the bpel file is not translatable.

5. Click **OK** to close the dialog.

26.3.4 Specifying Task Parameters

Specifying task parameters includes the following tasks:

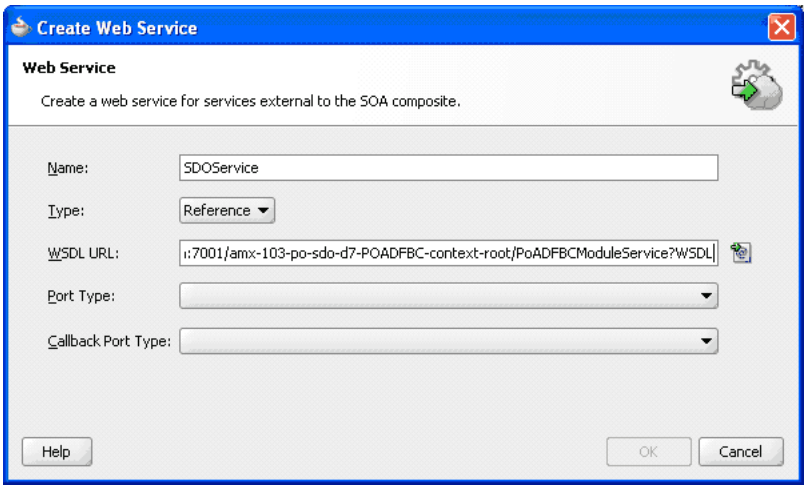
- Creating SDO references
- Defining entity parameters
- Defining collections

26.3.4.1 How to Create Service Data Object (SDO) References

An SDO service can be invoked from workflow services to retrieve the SDO as XML. This invocation is in the form of a SOA web service call. When the SDO service WSDL URL is available, a web service reference should be added using the Create Web Service dialog.

To create a reference, enter the WSDL URL and select the port type from the available port types, as shown in Figure 26–9.

Figure 26–9 Web Service Reference



For information about creating SDOs, see "Introduction to References" in the section "Introduction to the SOA Composite Editor" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

26.3.4.2 How to Define Entity Parameters

The following procedure enables you to accept a service data object (SDO).

1. Create a Service reference in the composite.

   This allows Fabric to create all the necessary wiring to a specific URL that points to a WSDL.

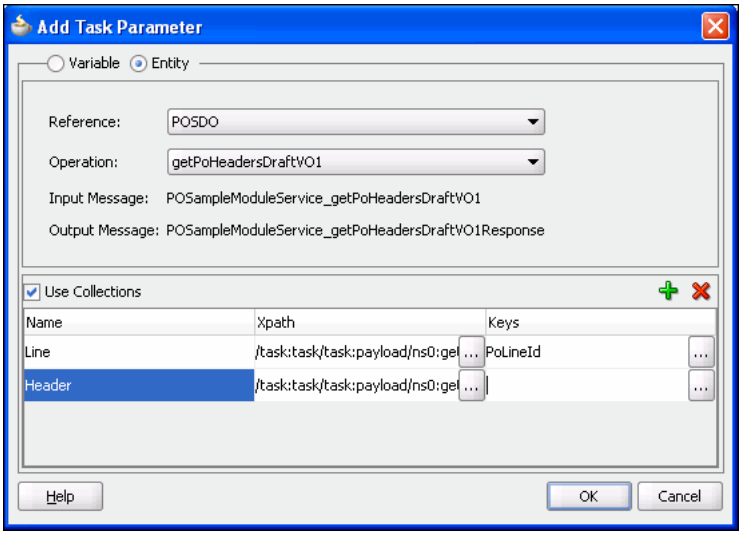
2. Define the task payload as external and specify which workflow retrieves the SDO object.

   This creates task parameters representing the input and output to the SDO web service.

3. Choose **Entity**.

4. Define the collection, its XPATH expression, and its keys, as shown in Figure 26–10.

Figure 26–10 Add Task Parameter: Define Collection



5. Set the collection for the stage.
6. Click **OK**.

The following procedure enables you to accept static XML.

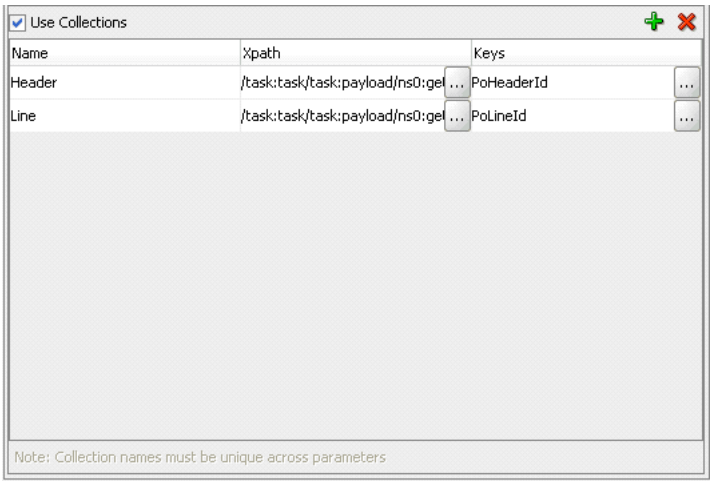
1. Provide the XSD where the schema is defined.
2. Define the task payload parameter as static XML.
3. Define the collection, its XPATH expression, and its keys.
4. Set the collection for the stage.
5. Click **OK**.

26.3.4.3 How to Define Collections

Collections are references to specific parts of a task message attribute, both static-XML based and entity attributes. After defined, collections can then be associated with stages to identify a stage as acting on a collection.

Defining a collection involves defining the name of the collection and the XPath to the collection element. If the collection is defined for an entity attribute, the keys for the collection element have to be specified as well. Each key has to be a direct child of the collection element. Figure 26–11 shows how collections are defined.

Figure 26–11 Defining Collections



When you define a collection, JDeveloper automatically determines if it should be repeating element or not. This information is used when collections are associated with a stage. A non-repeating collection can be associated with a singular stage. A repeating collection, when associated with a stage, repeats the stage in parallel for each element in the collection at run time. For information about how the collection information is used in a stage, see Section 26.3.6.1, "How to Model and Configure Stages."

26.3.5 Specifying Mapped Attributes

Human workflow provides task-message attributes that you can use for storing use-case-specific data, such as data extracted from a task's payload. These attributes are also known as *flexfield attributes* or *mapped flexfield attributes*.

Mapped flexfield attributes allow payload values to be displayed as columns in the task listing, rather than being hidden in the task details. These values are stored in the human workflow database schema, and you can use them in queries, view definitions, and assignment rule definitions.

There are two types of message attributes:

- *public* - Attributes mapped to specific task components at run time. These mappings can be changed at any time, and must be re-created when a task component is redeployed. For more information see the following sections in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*:
  - "How to Map Flex Fields" in the chapter "Using Oracle BPM Worklist"
  - "Introduction to Flex Fields" in the chapter "Introduction to Human Workflow"
- *protected* - AMX-specific mappings between a task component and protected flexfield attributes defined at design time. They cannot be changed at run time, and are deployed along with the task component.

  Table 26–1 summarizes the 60 available protected flexfield attributes.

Table 26–1 Protected Flexfield Attributes

Name	Description
ProtectedTextAttribute1 - ProtectedTextAttribute20	Stores text data, up to 2000 characters. The content in these fields is checked during keyword searches in the Worklist Application and through the task-query service.
ProtectedFormAttribute1 - ProtectedFormAttribute10	Stores text data, up to 2000 characters. The content in these fields is **not** checked during keyword searches in the Worklist Application.
ProtectedURLAttribute1 - ProtectedURLAttribute10	Stores text data, up to 200 characters. The content in these fields is **not** checked during keyword searches in the Worklist Application.
ProtectedDateAttribute1 - ProtectedDateAttribute10	Stores date information.
ProtectedNumberAttribute1 - ProtectedNumberAttribute10	Stores number information.

26.3.5.1 About Attribute Labels and Attribute-Label Mappings

Attribute labels are user-defined properties that allow a meaningful string to be applied to a particular flexfield attribute. The label should reflect the data to store in the attribute. For example, “CustomerName” for “ProtectedTextAttribute1,” “OrderNumber” for “ProtectedNumberAttribute2,” or “OrderDate” for “ProtectedDateAttribute1.”

A flexfield attribute can have multiple attribute labels defined for it. For example, the attribute “ProtectedTextAttribute1” could have the labels “CusomerName,” “PartId” and “EmployeeDepartment”.

Attribute-label mappings for protected attributes are defined at design time in the Human Task Editor. They define a mapping between a particular task component and an attribute label, and also specify how the value of the attribute should be populated. The same attribute label can be re-used in multiple mappings. This allows task components to map data having the same semantic meaning into a common attribute identified by a common label.

For example, PurchaseOrder, LoanRequest and ServiceRequest tasks all could define mappings to the “CustomerName” label. By sharing the same attribute labels across multiple task components, it is possible to construct worklist queries that query multiple task types and display or filter values from the common attribute labels. For example, it would be possible to construct a query that selected PurchaseOrder, LoanRequest, and ServiceRequest tasks, and then displayed the “CustomerName” as a column in the worklist task listing.

For more information, see Section 26.5.2, "How to Create Mapped Attribute Labels."

26.3.5.2 How to Define Attribute-Label Mappings

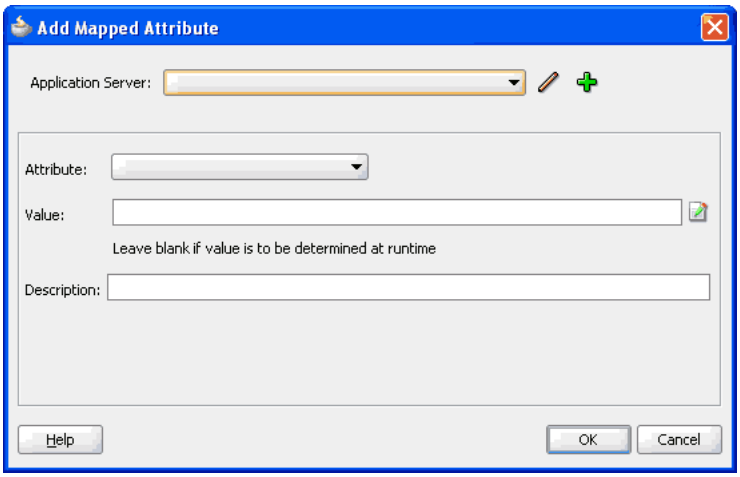
You define attribute-label mappings in the **Mapped Attributes** section of the Human Task Editor, as shown in Figure 26–12.

Figure 26–12 Mapped Attributes Section

Use the following procedure to define attribute-label mappings:

1. Click the **Add** icon to display the Add Mapped Attribute dialog, which is shown in Figure 26–13.

Figure 26–13 Add Mapped Attribute Dialog



2. Perform one of these options:
   - From the dropdown list, select the application server that contains the protected-attribute labels.
   - Click the **Add** icon to create a connection.
   - Click the **Edit** icon to edit an existing connection.

   The **Attribute** dropdown list populates with the available attribute labels from the specified server.
3. From the dropdown list, select an attribute.

   Note: The list does not include any labels for flexfield attributes to which this task component is being mapped.

4. At the **Value** field, specify a value using one of these options:
   - Enter an XPath expression that determines the value to be stored in the attribute.
   - Click the icon to create a value in the Expression Builder.
   - Leave the field blank to allow the value to be determined at run time.

   Usually, this XPath expression selects a value from the tasks's payload, but you can specify any valid expression that evaluates to a simple type, such as a string, a date, or a number.

   Be aware that specifying an XPath expression is not mandatory. You may prefer to set the value of the underlying flexfield-attribute value yourself. For example, you can add a custom assign activity to the BPEL process that initiates the task, or manipulate the Task object through the workflow service APIs.
5. Enter a description. This is optional.
6. Click **OK**.

26.3.6 Specifying Routing and Approval Policies

Specifying routing and approval policies includes the following tasks:

- Modeling and configuring stages
- Modeling task participants
- Modeling and configuring list builders
- Defining business rules
- Using business rules to specify list builders
- Using assignment-context information
- Aggregating task approvals

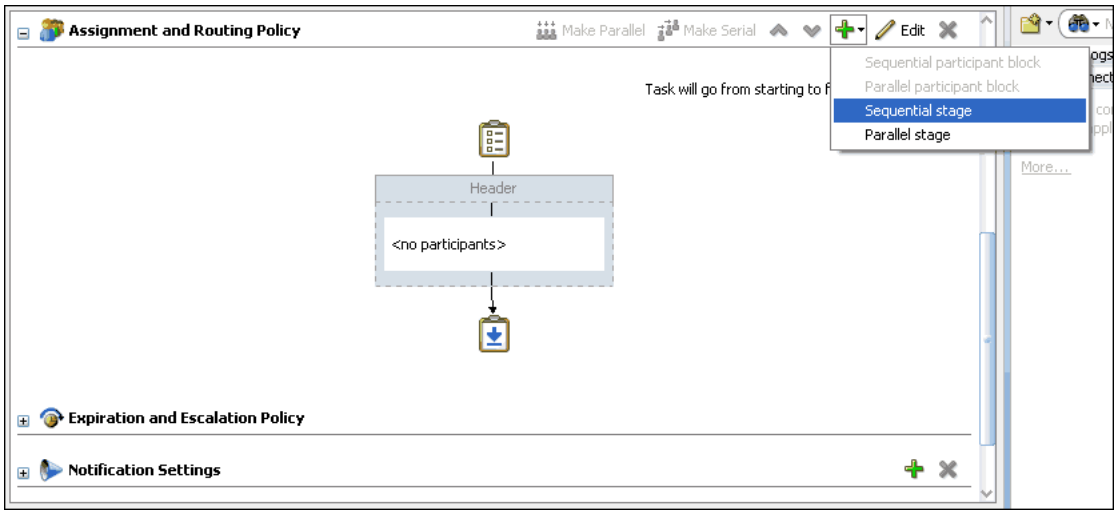
26.3.6.1 How to Model and Configure Stages

Based on functional needs, you can add and arrange multiple stages in a structure that can be a combination of sequential and parallel stages. This section describes how to create sequential and parallel stages.

Use the following procedure to create a stage:

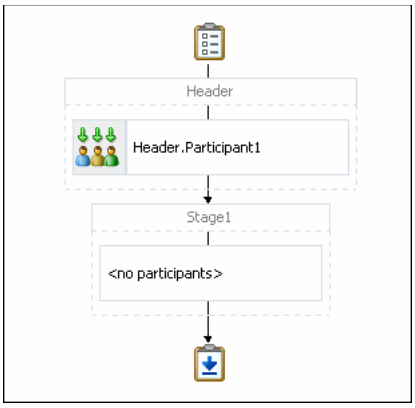
1. In the **Assignment and Routing** section of the Human Task Editor, select a stage.
2. Click the **Add** icon, and then select either **Sequential stage** or **Parallel stage** from the dropdown list, as shown in Figure 26–14.

Figure 26–14 Create Stage



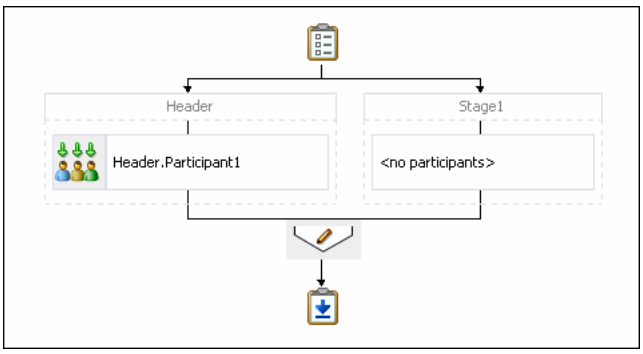
If you chose to create a sequential stage, the **Assignment and Routing** section looks like Figure 26–15.

Figure 26–15 Add Sequential Stage



If you chose to create a parallel stage, the **Assignment and Routing** section looks like Figure 26–16.

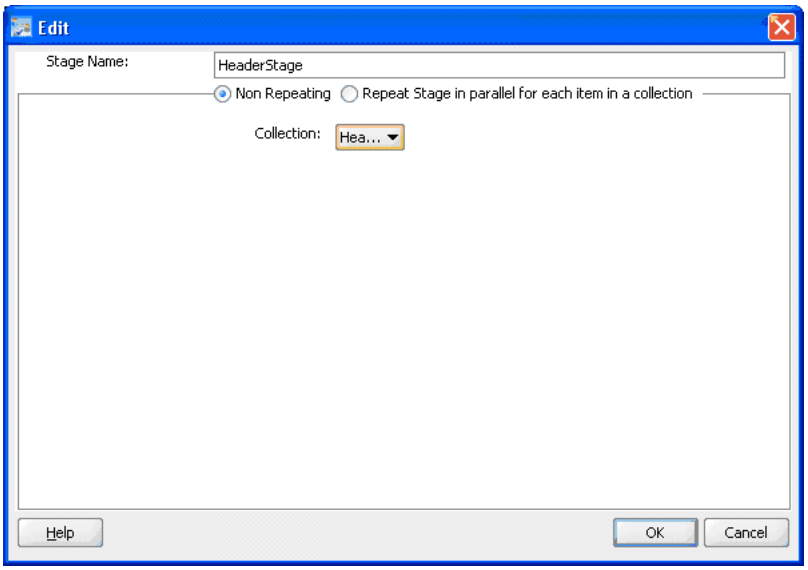
Figure 26–16 Add Parallel Stage



3. Double-click the stage you just created, or select the stage and click the **Edit** icon.

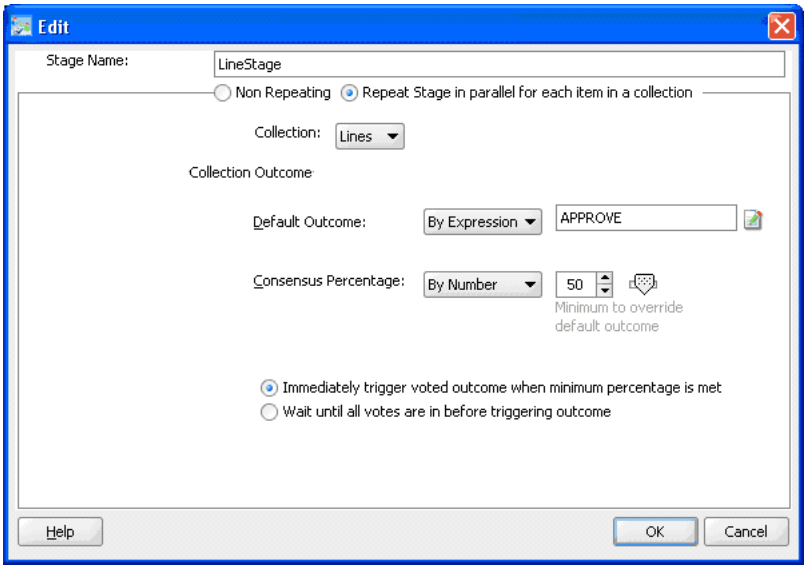
   The Edit dialog displays, as shown in Figure 26–17.

Figure 26–17 Edit Stage Dialog



4. Enter a name for the stage.
5. Choose one of these options:
   - **Non Repeating** - specifies that there is only one stage in parallel for each element in a collection
   - **Repeat Stage in parallel for each item in a collection** - specifies that the stage to repeat in parallel for each element in a collection. For example, if a purchase order contain 10 lines, the stage is repeated 10 times in parallel.
6. From the dropdown list, select a collection.
7. According to your selection, use one of these options:
   - If you selected **Non Repeating**, click **OK** to close the Edit dialog.
   - If you selected **Repeat Stage in parallel for each item in a collection**, additional options display, as shown in Figure 26–18.

Figure 26–18 Edit Stage Dialog: Repeat Stage



Do the following:

- Select a default outcome.
- Select a consensus percentage.
- Choose either to trigger the outcome immediately or wait until all the votes are in before triggering the outcome.
- Click **OK** to close the Edit dialog.

26.3.6.2 How to Model Task Participants

Inside each stage you either can edit the default task participant or add new task participants. Task participants are assigned based on routing patterns, which can be any of the following:

- Single
- Parallel
- Serial

- FYI

For information on participant types and assigning task participants to a stage, see "How to Assign Task Participants" in the "Creating Human Task Definitions with the Human Task Editor" section in the "Designing Human Tasks" chapter in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

After selecting a routing pattern, you also must select and model a list builder. This process is discussed in more detail in Section 26.3.6.3, "How to Model and Configure List Builders."

26.3.6.3 How to Model and Configure List Builders

Stage uses a combination of list builders to generate the approver list. For more information, see Section 26.2.3, "Stages" and Section 26.2.4, "List Builders." You can only use each type of list builder only one time per stage. You can arrange these approver list builders in either sequential or parallel order. The order you select governs the order in which those approvers included in approver lists that are generated by list builders are assigned an approval task.

The following list builders are specific to AMX:

- Approval Groups
- Job Level
- Position
- Supervisory

> **Note:** For information about modeling other list builders, see "Creating a Single Task Participant List" in the "How to Assign Task Participants" section in the "Creating the Human Task Definition with the Human Task Editor" chapter in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Table 26–2 describes the AMX-specific list builders and the options available to them.

Table 26–2 List-Builder Options

Option Name	Description	List Builder
Value-based	Specifies constraints to build the list of participants based on provided values.	All Except Position
Rule-based	Specifies constraints to build the list of participants based on rules that are defined in the Rule Editor.	All
Name	The name of the approval group to use.	Approval Groups
Allow Empty Groups	Allows the use of approval groups with no members.	Approval Groups
List Ruleset	Name of the ruleset specifying constraints for building participant list.	All
Starting Participant	The first participant in a list, usually a manager.	Job Level Position Supervisory

Table 26–2 (Cont.) List-Builder Options

Option Name	Description	List Builder
Top Participant	The last participant in the approval. Approval does not go beyond this participant in a hierarchy.	Job Level Position Supervisory
Number of Levels	A positive number specifying the number of levels to traverse for Supervisory, or the number of job level for Job Level and Position. This number can be an absolute value, or a value relative to starting point or creator.	Job Level Position Supervisory
Relative to	A positive number specifying the number of levels to traverse for Supervisory, or the number of job level for Job Level and Position. Possible values are: starting point, creator and absolute.	Job Level Position
Include all managers at last level	If the job level equals that of the previously calculated last participant in the list then it includes the next manager in the list.	Job Level
Utilized Participants	Utilizes only the participants specified in this option from the calculated list of participants. Available options are: Everyone, First and Last manager, Last manager.	Job Level Position
Auto Action Enabled	Specifies if the list builder automatically acts on task based on the next option.	Supervisory Job Level Position
Auto Action	Specifies the outcome to be set. It can be null if auto action is not enabled.	Supervisory Job Level Position

If you do not configure the hierarchy provider plug-in, then the position list builder does not work.

When you define a hierarchy extension, if you do not define the property `mustUseSpecifiedProvider` then its default value is `true`..

You can configure the Supervisory and Job Level list builders not to throw an exception when there is a problem with the hierarchy plug in. To configure the list builders, you must add the `mustUseSpecifiedProvider` property to the `workflow-identity-config.xml` configuration file, and set the value attribute to `false`.

By default, the `workflow-identity-config.xml` file does not include the `mustUseSpecifiedProvider` property. If this property is present and its value is `false`, then, then the Supervisory and Job Level list builders use the LDAP management chain when there is a problem with the hierarchy plugin.

Example 26–5 shows a `workflow-identity-config.xml` file that specifies the mustUseSpecifiedProvider property. The value of this property is set to true so that the Supervisory and Job Level builders fail when the hierarchy plug in is not available.

Example 26–5 workflow-identity-config.xml Configuration File

```
<ISConfiguration xmlns="http://www.oracle.com/pcbpel/identityservice/isconfig">
```

```
  <configurations>
    <configuration realmName="jazn.com">
      <provider providerType="JPS" name="JpsProvider" service="Identity">
        <property name="jpsContextName" value="default"/>
        <property name="IdentityServiceExtension"
                  value="HCMIdentityServiceExtension"/>
      </provider>
    </configuration>
  </configurations>
  <property name="caseSensitive" value="false"/>
  <property name="mustUseSpecifiedProvider" value="true"/> <!-- Fail when the
hierarchy plug ins are not available-->
  <serviceExtensions>
...
</ISConfiguration>
```

26.3.6.3.1 How to Model an Approval Groups List Builder Approval groups are a statically defined or a dynamically generated list of approvers. Approval groups usually are configured by the process owner using the worklist application. Typically, they are used to model subject matter experts outside the transaction's managerial chain of authority, such as human resources or legal counsel, that must act on a task before or after management approval.

Static approval groups are predetermined lists of approvers, while dynamic approval groups generate approver lists at run time. Dynamic approval groups require:

- delivery of an implementation according to the dynamic approver list interface by the developer
- registration of the above implementation as a dynamic approval group using the Worklist Application's UI by the IT department
- availability of the class file in a globally well-known directory that is part of the SOA class path

> **Note:** In Drop 8, an approval group is a flat list. Serial and parallel patterns no longer are defined.

For more information, see Section 26.5, "Using Approval Management Features of the Oracle BPM Worklist and Workspace."

Two views of the Approval Groups list builder are shown in Figure 26–19 and Figure 26–20.

Figure 26–19 Value-Based Approval Groups List Builder Dialog

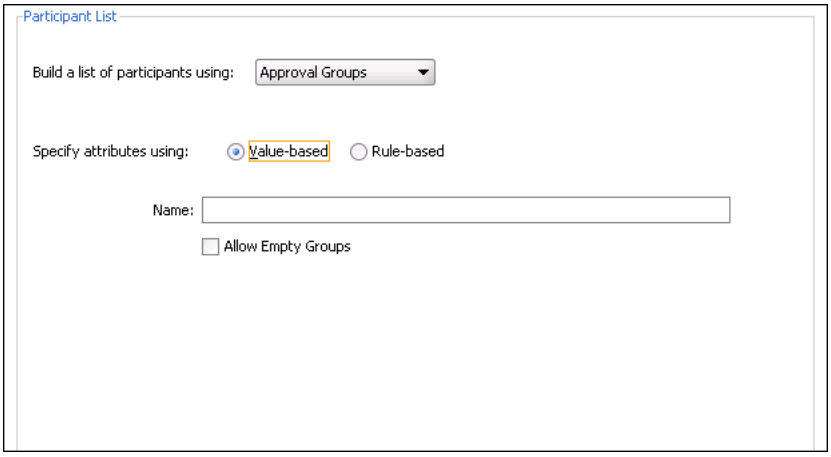
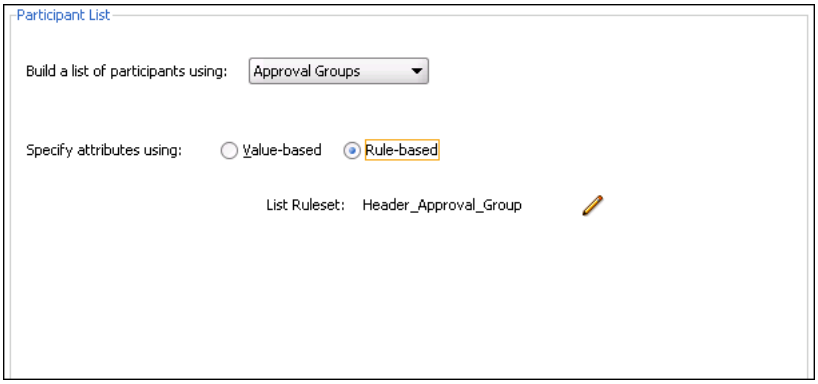


Figure 26–20 Rule-Based Approval Groups List Builder Dialog



To model an Approval Groups list builder, first specify if the list builder's attributes are to be value-based or rule-based, and then select the options on the corresponding dialog. For information about the options, see Table 26–2.

26.3.6.3.2 How to Model a Job Level List Builder The Job Level list builder ascends the supervisory hierarchy, starting at a given approver and continuing until an approver with a sufficient job level is found.

Two views of the Job Level list builder are shown in Figure 26–21 and Figure 26–22.

Figure 26–21 Value-Based Job Level List Builder Dialog

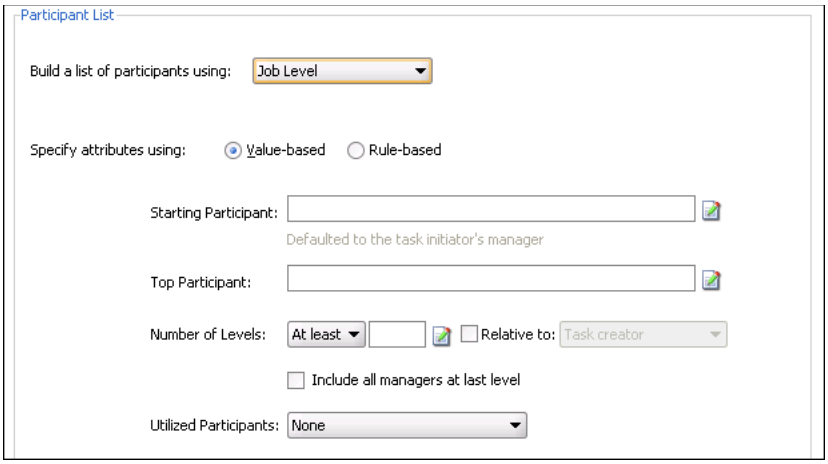
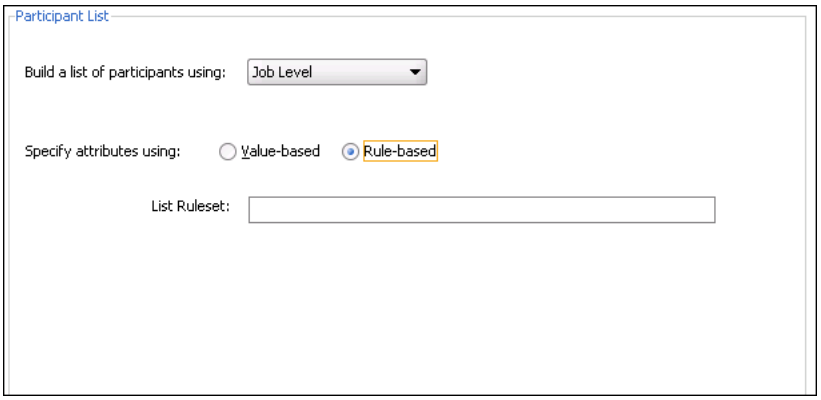


Figure 26–22 Rule-Based Job Level List Builder Dialog

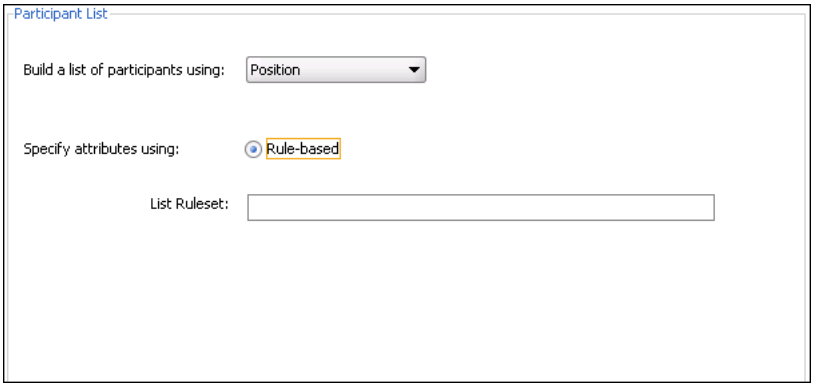


To model a Job Level list builder, first specify if the list builder's attributes are to be value-based or rule-based, and then select the options on the corresponding dialog. For information about the options, see Table 26–2.

26.3.6.3.3 How to Model a Position List Builder The Position list builder ascends the position hierarchy, starting at the requester's or at a given approver's position, and goes up a specified number of levels or to a specific position.

Figure 26–23 shows a view of the Position list builder.

Figure 26–23 Rule-Based Position List Builder Dialog



To model a Position list builder, first specify if the list builder's attributes are to be value-based or rule-based, and then select the options on the corresponding dialog. For information about the options, see Table 26–2.

26.3.6.3.4 How to Model a Supervisory List Builder The Supervisory list builder ascends the primary supervisory hierarchy, starting at the requester or at a given approver, and generates a chain that has a fixed number of approvers in it.

Two views of the Position list builder are shown in Figure 26–24 and Figure 26–25.

Figure 26–24 Value-Based Supervisory List Builder Dialog

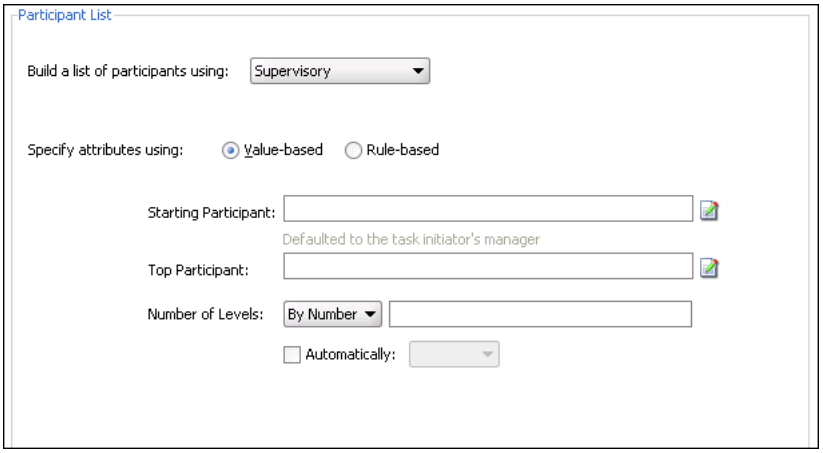
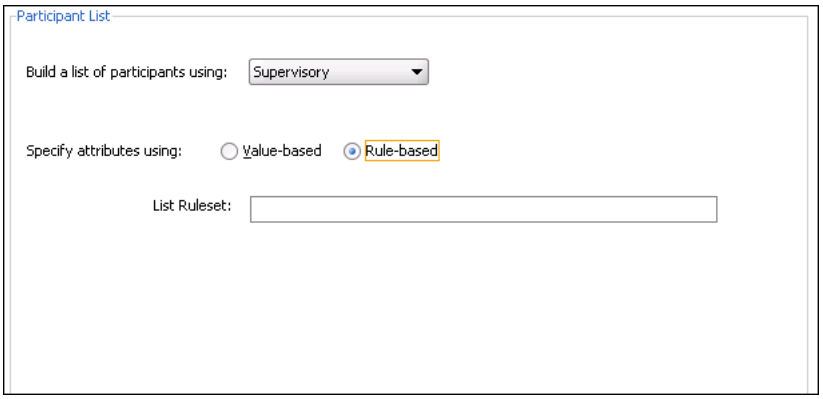


Figure 26–25 Rule-Based Supervisory List Builder Dialog



To model a Supervisory list builder, first specify if the list builder's attributes are to be value-based or rule-based, and then select the options on the corresponding dialog. For information about the options, see Table 26–2.

26.3.6.4 How to Use Business Rules to Specify List Builders

Approvers of a task can be defined either inline in a task definition or by using business rules to specify the list builders that identify the actual approvers of a task. In addition, you can use business rules to specify approver substitution and list modifications. These rules are defined with the help of Oracle Business Rules and can vary between organizations. Typically, however, they are defined by the customer.

Business rules are a combination of conditions and actions. Optionally, priority and validity periods can be defined for these rules. In Human Workflow rules, rule conditions are defined using fact types that correspond to the task, and to the task message and entity attributes (which are XML representation of SDO objects). Rule actions consist of approver list builders and their parameters. Approver list builders move up a particular hierarchy and construct or modify the approver list according to the parameters defined. Approver list builders are implemented as XML (JAXB) fact types.

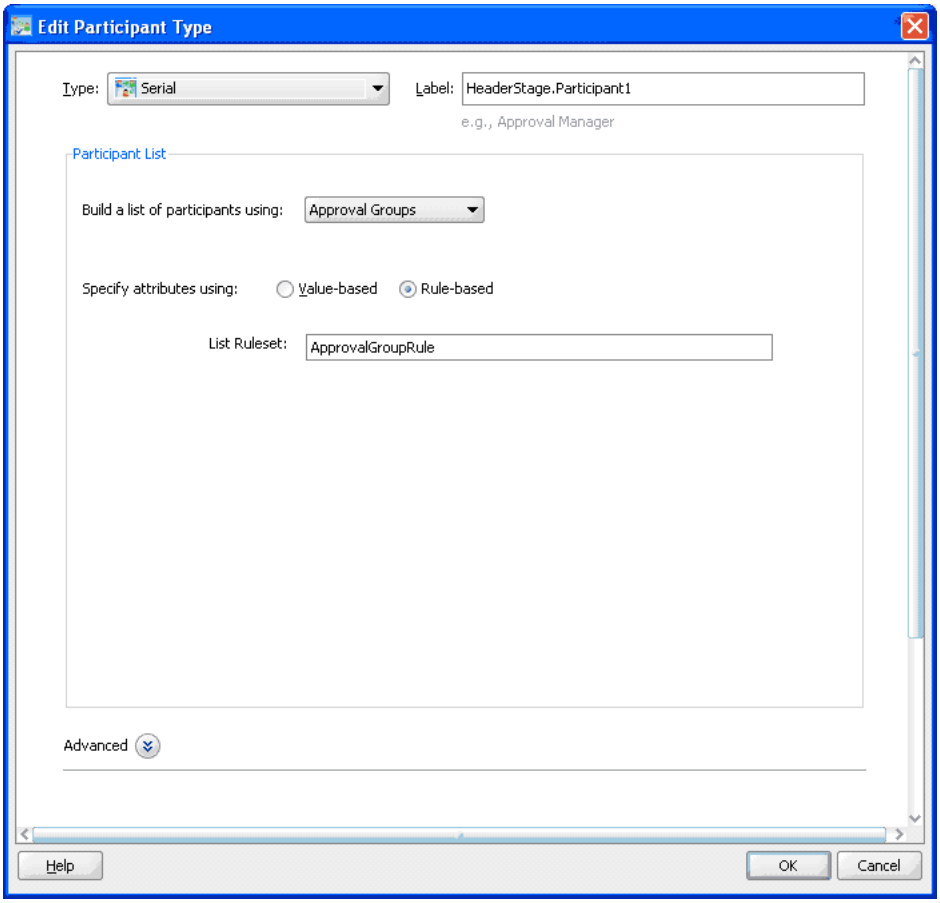
For more information about these concepts, see the chapter "Overview of Oracle Business Rules" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

The sections that follow explain list creation, approver substitution, list modification, and repeating node attributes using Oracle Business Rules.

26.3.6.4.1 How to Create Lists You can use business rules to define the list builders you want to use. There are two types of business rules:

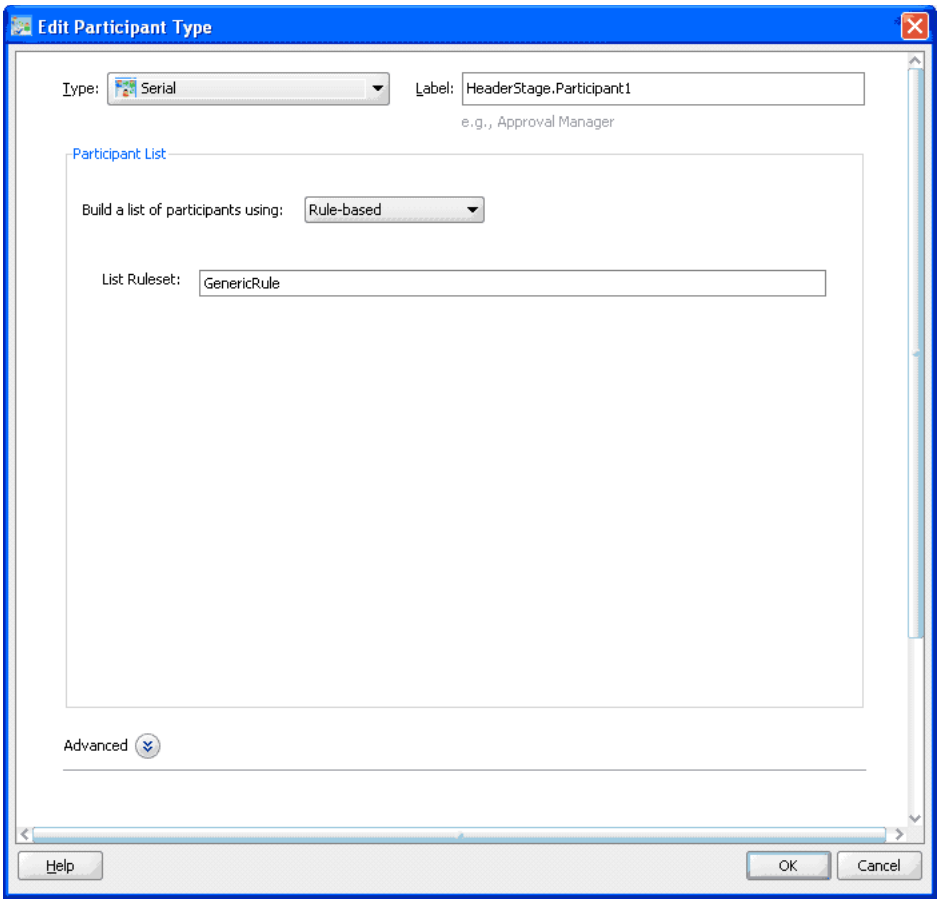
- Rules that define the parameters of a specific list builder. In this case, the task routing pattern dialog is modeled to use a specific list builder. The parameters in the list builder come from rules. With this option, rules should return a list builder of the same type as the one modeled in JDeveloper. Figure 26–26 shows a sample configuration.

Figure 26–26 Specific List-Builder Configuration



- Rules that define the list builder and the list-builder parameters. In this case, the list itself is built using rules. Figure 26–27 shows a sample configuration.

Figure 26–27 List Builder and Parameters Configuration

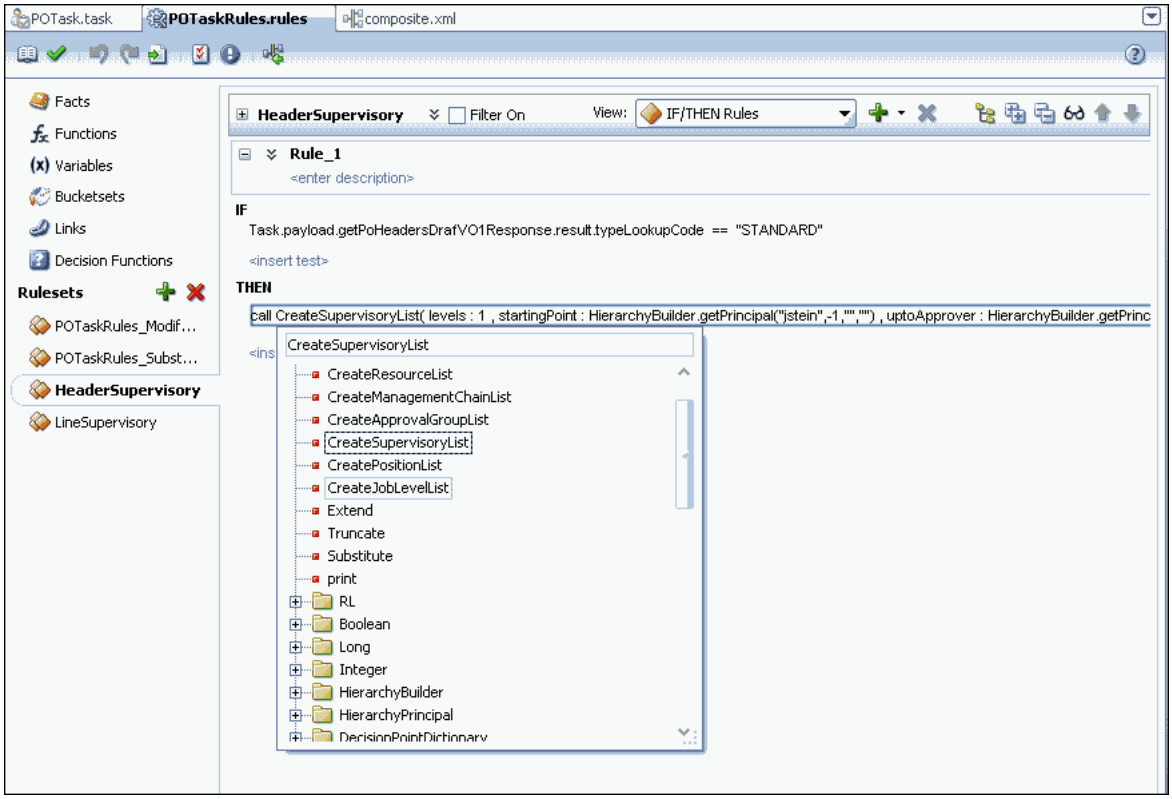


In the rule dictionary, rule functions are seeded to facilitate the creation of list builders. These functions are the following:

- CreateResourceList
- CreateSupervisoryList
- CreateManagementChainList
- CreateApprovalGroupList
- CreateJobLevelList
- CreatePositionList

In Rules Designer, model your conditions and, in the action part, "call" one of the functions above to complete building your lists, as shown in Figure 26–28.

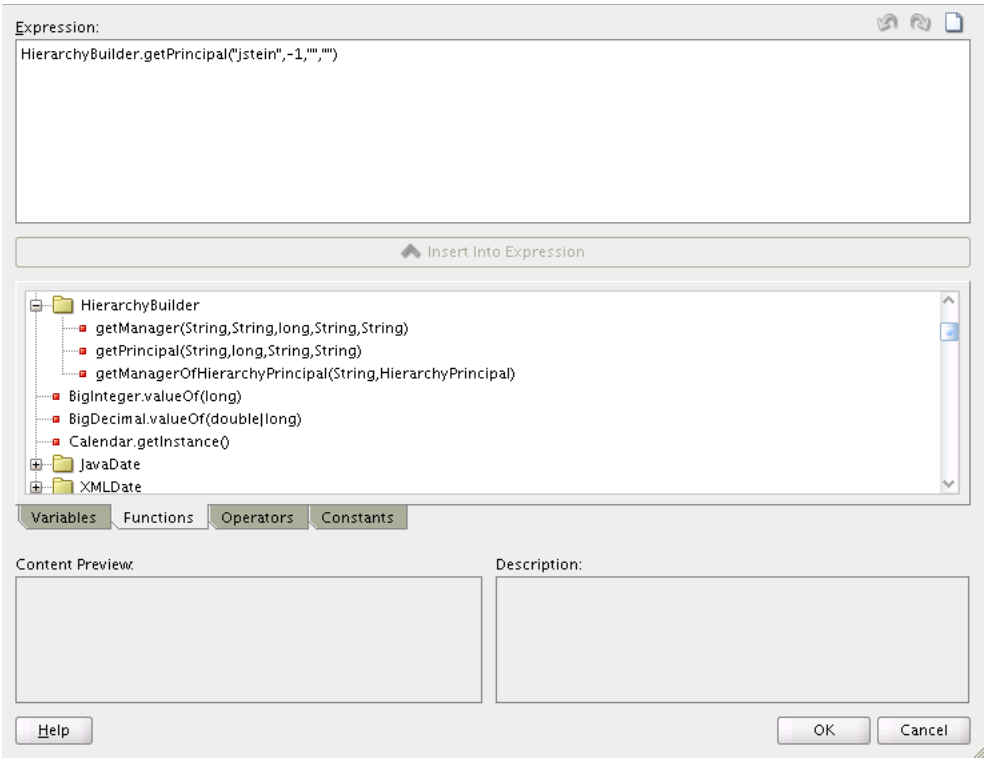
Figure 26–28 Modeling Conditions in Rules Designer



The parameters for the rule functions are similar to the ones in JDeveloper modeling. In addition to the configurations in JDeveloper, some additional options are available in Rules Designer for the following attributes:

- startingPoint and topApprover - In JDeveloper, starting point and top approver are specified as users. In Rules Designer, you can build a HierarchyPrincipal as the starting point and top approver. To build a Hierarchy Principal, use the HierarchyBuilder function, as shown in Figure 26–29.

Figure 26–29 HierarchyBuilder Function



- autoActionEnabled and autoAction - From Rules Designer, you can configure that the users resulting from a particular list builder can act automatically on the task.
- responseType - If the response type is REQUIRED, the assignee has to act on the task; otherwise, the assignment would be converted to an FYI assignment.
- ruleName - Rule name is used to create an assignment reason. Rule set name + "_" + rule name is used as a key to look up the resource bundle for a translatable reason for assignment. This resource is looked up first in the project resource bundle, then in the custom resource bundle, and last in the system resource bundle.
- lists - This is an object that is a holder for all the lists that are built. Clicking this option shows a pre-asserted fact 'Lists' object to be used as the parameter.

Figure 26–30 and Figure 26–31 show examples of rules.

Figure 26–30 Example Rules (1)

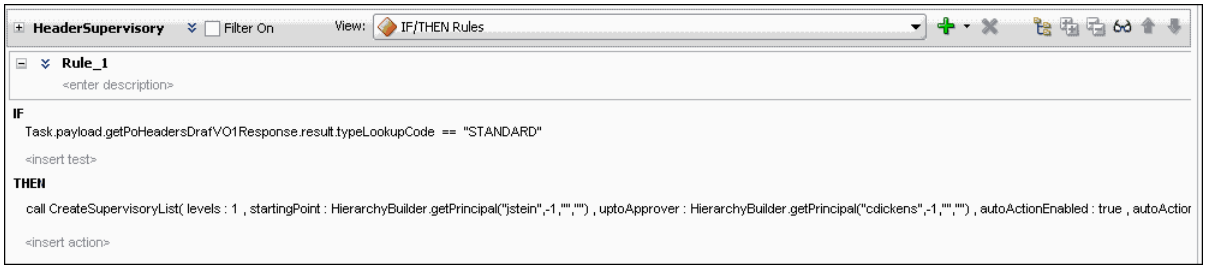
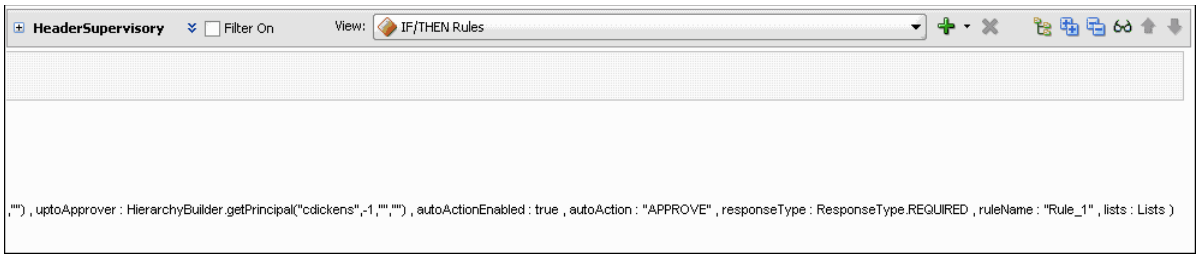


Figure 26–31 Example Rules (2)



Note: If multiple rules fire, the list builder created by the rule with the highest priority is selected.

If the rules have the same priority, they are fired in random order, the first one fired is selected.

WARNING: An improper or incomplete rules definition in a list-creation rule set can cause run-time errors. Errors can be caused by the following:

- **No rule was defined in the rule set.**
- **None of the conditions defined in the rule was met.**

Ensure that rules are properly defined to handle all conditions.

26.3.6.4.2 How to Make Approver Substitutions List substitution enables you to substitute users, groups, and application roles that appear in a list. List substitution is available from Rules Designer and does not require any configuration in JDeveloper. In each rule dictionary there is a pre-seeded rule set named "SubstitutionRules." Also in the rule dictionary, a "Substitute" rule function is seeded to configure list substitutions. Table 26–3 lists the "Substitute" functions and their parameters.

Table 26–3 "Substitute" Function Parameters

Parameter	Description
fromId	The ID of the user/group/application role from which to substitute.
toId	The ID of the user/group/application role which to substitute to.
ruleName	Used to create an assignment reason. Rule set name + "_" + rule name is used as a key to look up the resource bundle for a translatable reason for assignment. This resource is looked up first in the project resource bundle, then in the custom resource bundle, and last in the system resource bundle.
substitutionRules	An object that is a holder for all the substitutions. Clicking this option shows a pre-asserted fact 'SubstitutionRules' object to be used as the parameter.

Figure 26–32 shows a sample approver-substitution action.

Figure 26–32 Sample Approver-Substitution Action

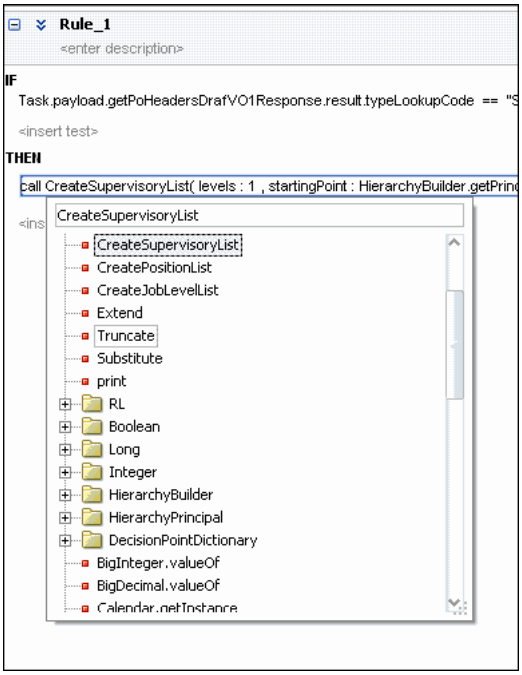
26.3.6.4.3 How to Make List Modifications List modification enables you to extend or truncate the Job Level and Position list builders from rules. List modification is applied after the list is created. This feature does not require any configuration from JDeveloper. In each rule dictionary there is a pre-seeded rule set named "ModificationRules." This rule set is called only when the Job Level and Position list builders are asserted in the list that created the rule sets. Only the highest priority applicable rule is applied.

In Rules Designer, rule functions are seeded to facilitate list modifications. These functions are the following:

- Extend
- Truncate

These rule functions are shown in Figure 26–33.

Figure 26–33 Rule Functions



Extend and truncate parameters are listed in Table 26–4 and Table 26–5.

Table 26–4 "Extend" Function Parameters

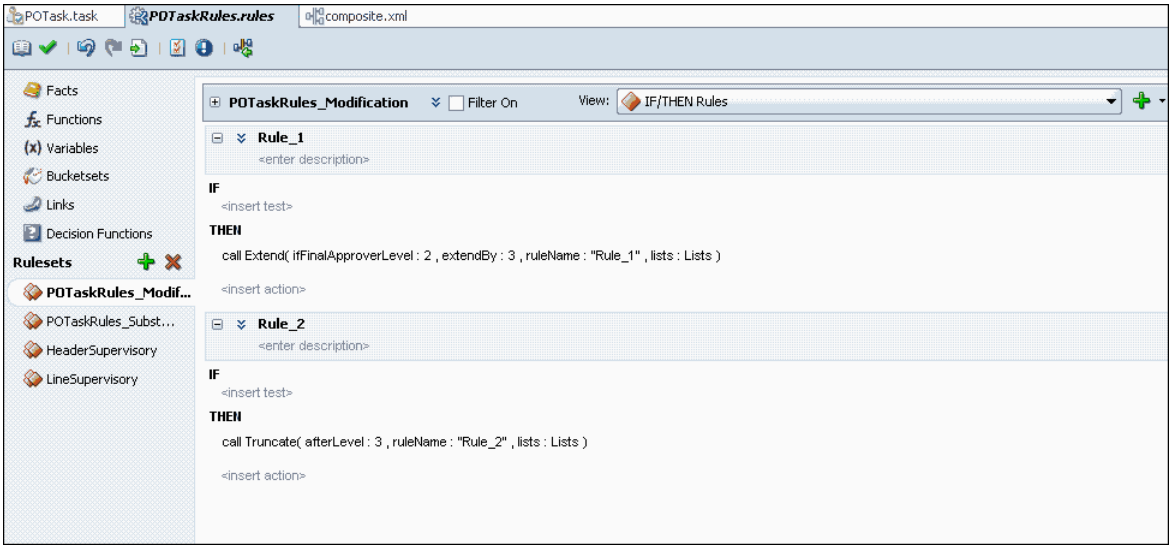
Parameter	Description
ifFinalApproverLevel	The level at which final approver is at or below.
extendBy	The number of levels to add to the final job level.
ruleName	Used to create an assignment reason. Rule set name + "_" + rule name is used as a key to look up the resource bundle for a translatable reason for assignment. This resource is looked up first in the project resource bundle, then in the custom resource bundle, and last in the system resource bundle.
lists	An object that is a holder for all the lists that are built. Clicking this option shows a pre-asserted fact 'Lists' object to be used as the parameter.

Table 26–5 "Truncate" Function Parameters

Parameter	Description
afterLevel	The level after which to truncate.
ruleName	Used to create an assignment reason. Rule set name + "_" + rule name is used as a key to look up the resource bundle for a translatable reason for assignment. This resource is looked up first in the project resource bundle, then in the custom resource bundle, and last in the system resource bundle.
lists	An object that is a holder for all the lists that are built. Clicking this option shows a pre-asserted fact 'Lists' object to be used as the parameter.

Figure 26–34 shows a sample list-modification action.

Figure 26–34 Sample List-Modification Action



26.3.6.4.4 How to Define Repeating-Node Attributes of a Business Rule Condition When defining a business rule, you can base a rule condition on an attribute that comes from a repeating node. For example, there can be multiple line items for each purchase-order header in a purchase-order scenario. In this case,

PurchaseOrderHeader is a non-repeating node, and PurchaseOrderLines is a repeating node.

When defining a rule like the following:

IF line item's amount is <50000, THEN create supervisory list containing jcooper up to two levels

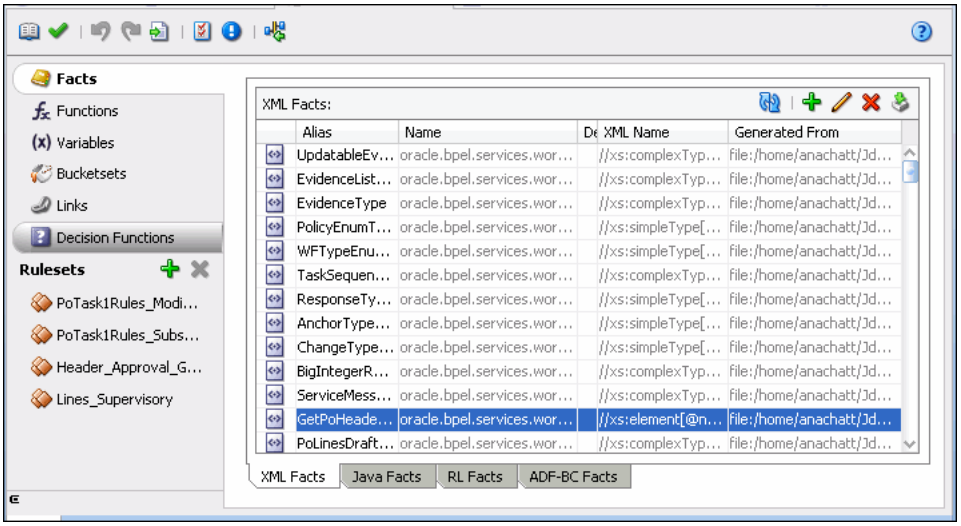
the amount is an attribute of *line*, that is, it is an attribute of a repeating node.

Use the following procedure to define repeating-node attributes:

1. In Rules Designer, select **Facts**.

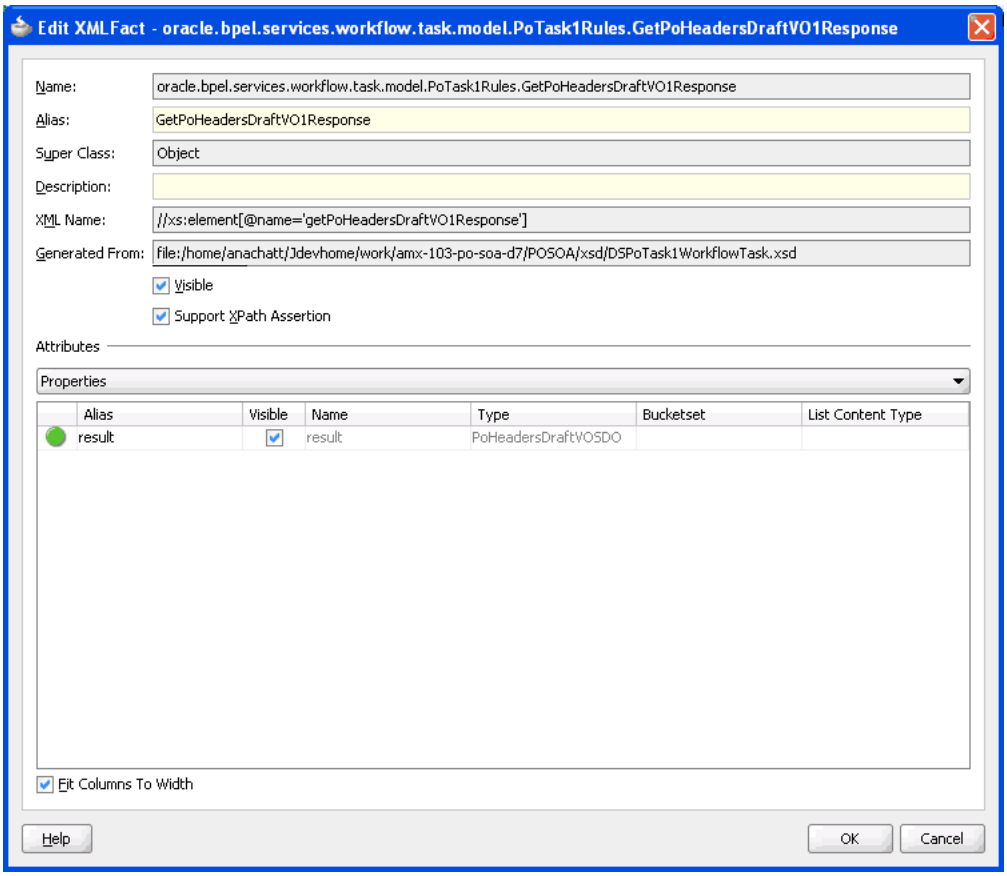
   A list of facts displays, as shown in Figure 26–35.

Figure 26–35 Facts List



2. Edit each appropriate fact to ensure that it is visible, as shown in Figure 26–36.

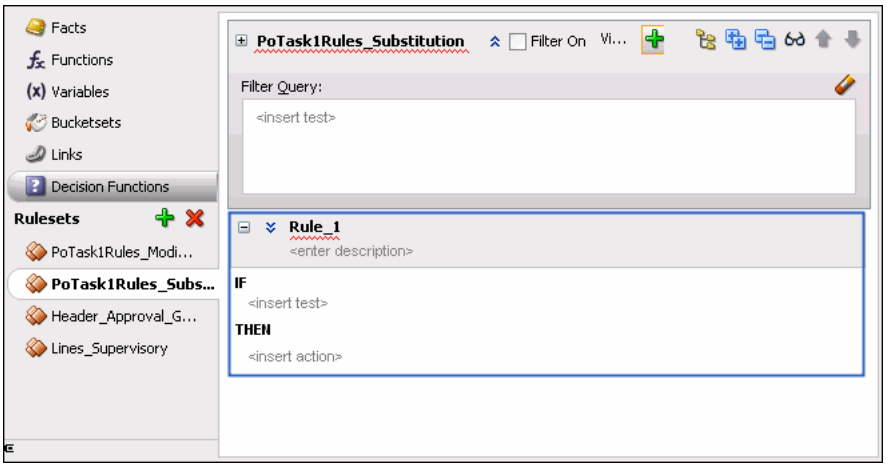
Figure 26–36 Edit XML Fact Dialog



3. In Rules Designer, select a rule and then click **Add** icon (+).

   A rule-definition section displays, as shown in Figure 26–37.

Figure 26–37 Rule-Definition Section

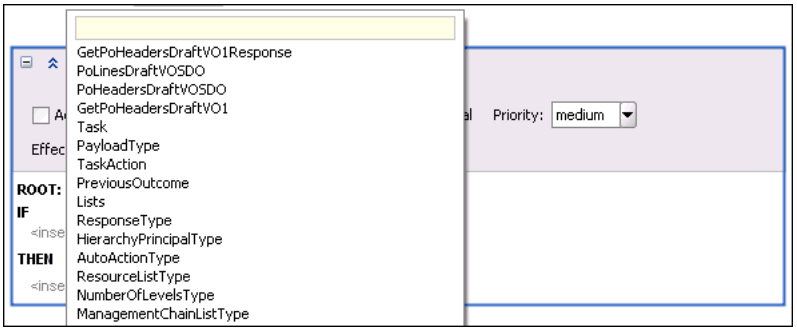


4. Click the double down arrows to the left of the rule name to show advanced settings, as shown in Figure 26–38.

Figure 26–38 Advanced Settings

5. Select **Tree Mode**, then click **<fact type>** to display a list of options from which to choose a ROOT, as shown in Figure 26–39.

Figure 26–39 ROOT Options



6. Define the rule conditions.

26.3.6.5 How to Use Assignment Context

Assignment context is information that is present in the task. During a task's life cycle, it progresses through various assignees. As the context of the task assignees changes, the assignment-context value also changes.

When browsing through the history of a task, you can see the various assignment contexts that the task contained during its life cycle. The Worklist Application uses assignment context when it displays task history.

26.3.6.5.1 Configuring Assignment Context You configure assignment context in the Add (or Edit) Participant Type dialog in JDeveloper in the following ways:

- Select the **Rule-based** option in the **Participant Type** section.

  In this case, the assignment context is configured implicitly, behind the scenes. The Rules layer resolves the list of assignees based on the rule. As the task progresses through the various assignees, the assignment context value is computed based on the rule.

- Expand the **Advanced** section to configure any number of assignment contexts.

  In this case, you can customize assignment contexts by entering your own information into the Assignment Conext fields. Figure 26–40 shows the fields.

Figure 26–40 Assignment Context Section

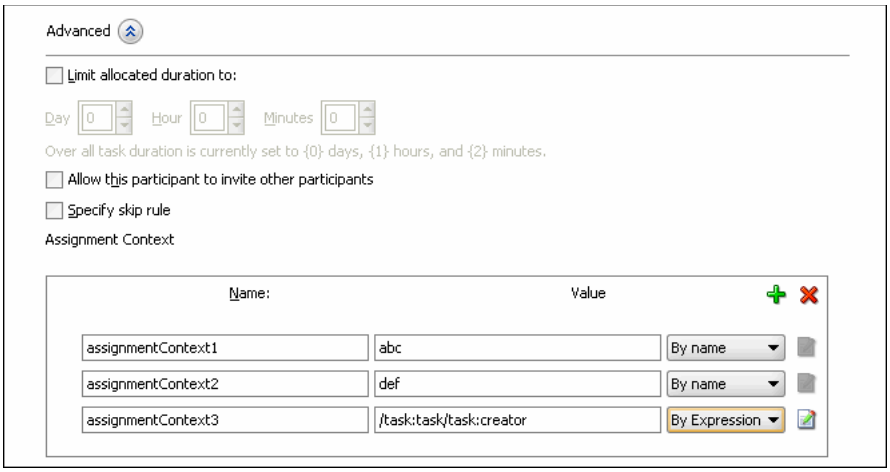


Table 26–6 contains field descriptions.

Table 26–6 Assignment-Context Field Descriptions

Field	Description
Name	Assignment-context name, which can be whatever you choose. This is a string field.
Value	Assignment-context value, which can be whatever you choose. This is a string field.
Type	Associated with the Value field. Possible values are: ■ **By name** - A user-provided Value parameter. ■ **By Expression** - A Value parameter created by the Expression Builder.

26.3.6.6 How to Aggregate Task Approvals

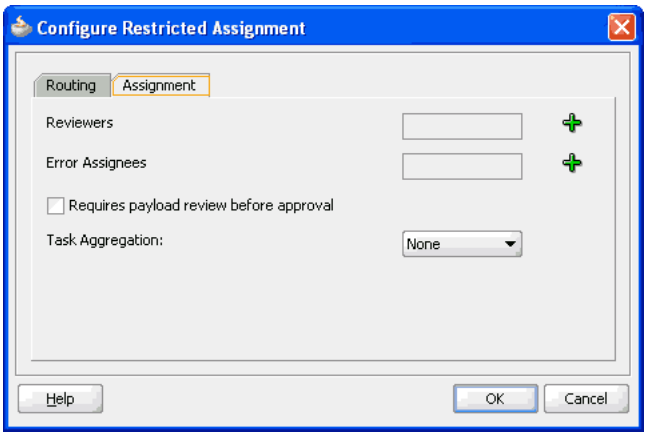
A task can be assigned multiple times to one user during the task life cycle. The Human Task Editor enables you to configure how often a user sees the task.

The following procedure explains how to configure task-approval aggregation.

1. In the **Assigmnent and Routing Policy** section of the Human Task Editor, select a stage and click the pencil icon to the right of "Task goes from starting to final participant."
2. In the Configure Restricted Assignment dialog, select the Assignment tab.

   Figure 26–41 shows the tab selected in the dialog.

Figure 26–41 Configure Restricted Assignment Dialog



3. Select a task-aggregation option from the dropdown list:
   - None - Indicates there is no approval aggregation, which means the user sees the task as many times as it is assigned to him or her.
   - Stage - A user sees the task only one time in a stage.
   - Task - A user sees the task only one time in the task life cycle.
4. Click **OK**.

When the task is aggregated and assigned to a user, the task has a collection table in the Worklist Application that diplays all the collections in the task the user is approving. After the user performs an action, the action is recorded and then replayed to all the user's assignments, either in the stage or task.

An aggregated task is a proxy task for all the regular assignments. Only the following actions are permitted on an aggregated task:

- Custom actions like approve/reject
- Reassign
- Acquire

The user can perform additional actions on the individual tasks, such as escalate, but those actions are not be recorded and played back for other tasks. Those actions are treated as actions on an individual task and not on an aggregated task.

26.3.7 Defining Escalation and Renewal Policies

This feature is not specific to AMX. For more information, see "Designing Human Tasks" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Note: Escalation is only applicable to management chain.

26.3.8 Specifying Notification Settings

This feature is not specific to AMX. For more information, see "Designing Human Tasks" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

26.3.9 Using Advanced Settings

Using advanced settings includes the following tasks:

- Specifying callbacks for notes, attachments, and validation
- Defining security access rules

26.3.9.1 How to Add Callbacks for Notes, Attachments, and Validation

Callbacks are mechanisms that allow you to do the following:

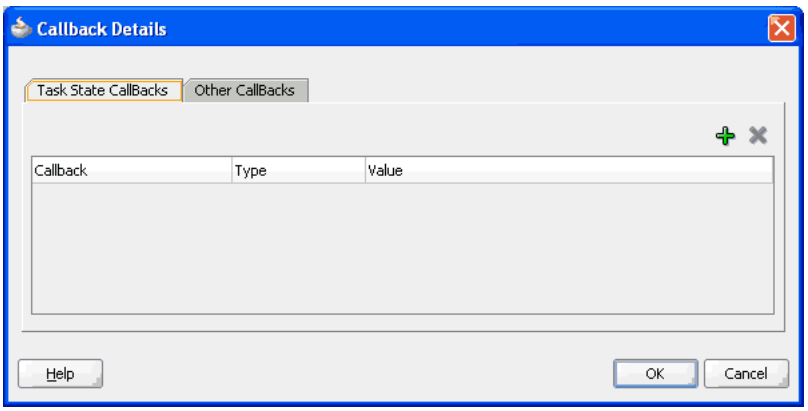
- Access notes and attachments associated with business objects from external content-management systems or custom schemas
- Perform custom validation of workflow tasks at various points in a task life cycle by defining validation logic for each task action

Use the following procedure to add callbacks:

1. From the Task Editor, expand the **Advanced Settings** section.
2. Click **Configure Callbacks...**.

   The Callback Details dialog opens, as shown in Figure 26–42.

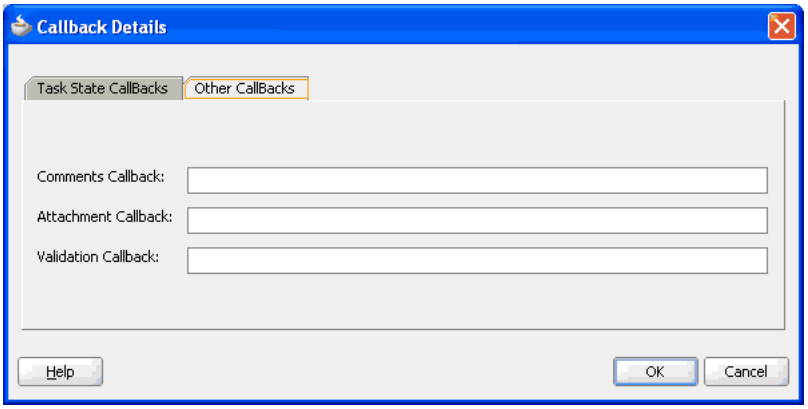
Figure 26–42 Callback Details Dialog



3. Click the Other Callbacks tab.

   Additional entry fields display, as shown in Figure 26–43.

Figure 26–43 Other Callbacks Tab



4. Use one of these options:
   - In the Comments Callback field, enter the appropriate Java class for the notes callback.
   - In the Attachments Callback field, enter the appropriate Java class for the attachments callback.
   - In the Validation Callback field, enter the appropriate Java classes, separated by commas, for the validation callback.
5. Click **OK**.

26.3.9.2 How to Define Security Access Rules

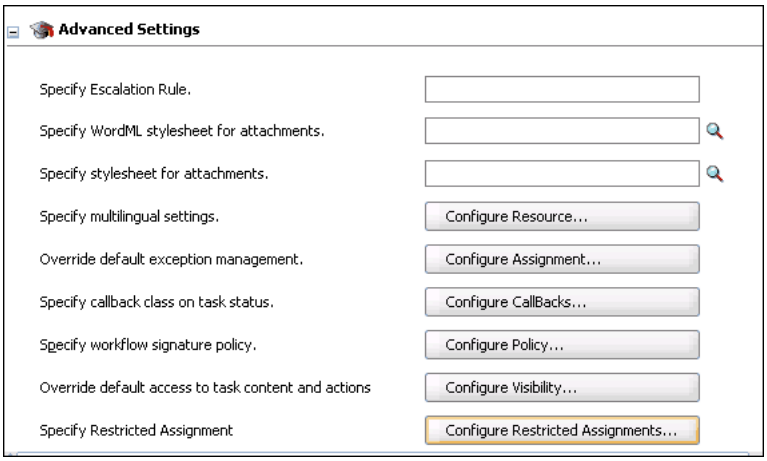
Access rules restrict the actions that a user can perform by overriding default actions and permissions. At run time, the system checks every operation in a task against any defined access rules to see if a user is permitted to make changes, such as approve, add, delete, and so on If the user is not permitted to make changes, the operation errors out with an appropriate error message.

In AMX, access rules can be defined for Groups and Application Roles. For example, if an access rule is defined to restrict the "Withdraw" action for a group called Operators, then any user belonging to that group is not allowed to withdraw the task. Similarly, if an access rule is defined to restrict the "Withdraw" action for an application role called SOAAuditViewer, then any user who has been granted the SOAAuditViewer application role is not allowed to withdraw the task.

To define a security access rule:

1. Expand the **Advanced Settings** section of the Human Task Editor, and locate the **Override default access to task content and actions** option, as shown in Figure 26–44.

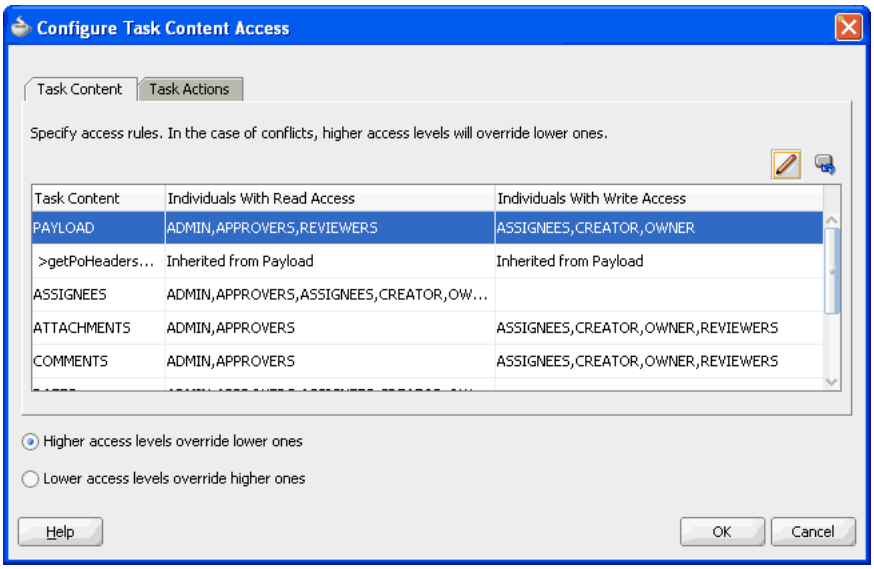
Figure 26–44 Access Rules Option



2. Click **Configure Visibility...**.

   The Configure Task Content Access dialog displays, as shown in Figure 26–45.

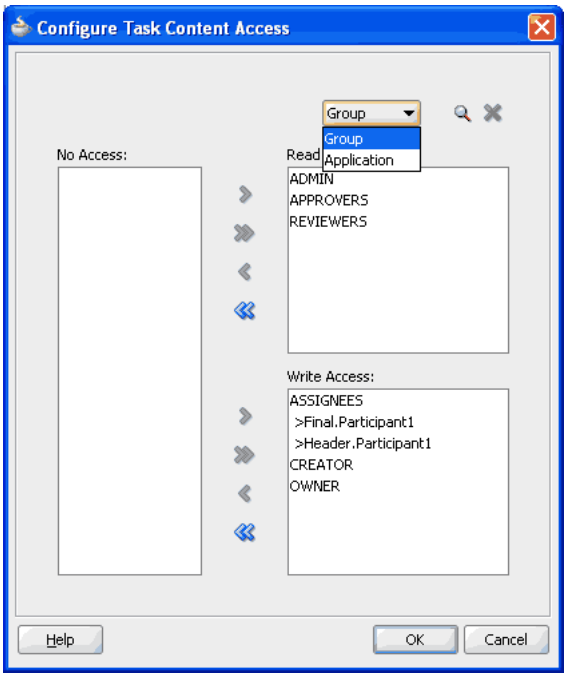
Figure 26–45 Configure Task Content Access Dialog (1)



3. Click the **Task Content** or **Task Actions** tab to select it. (This procedure assumes the **Task Content** tab has been selected.)
4. Select a task to edit and click the **Edit** icon.

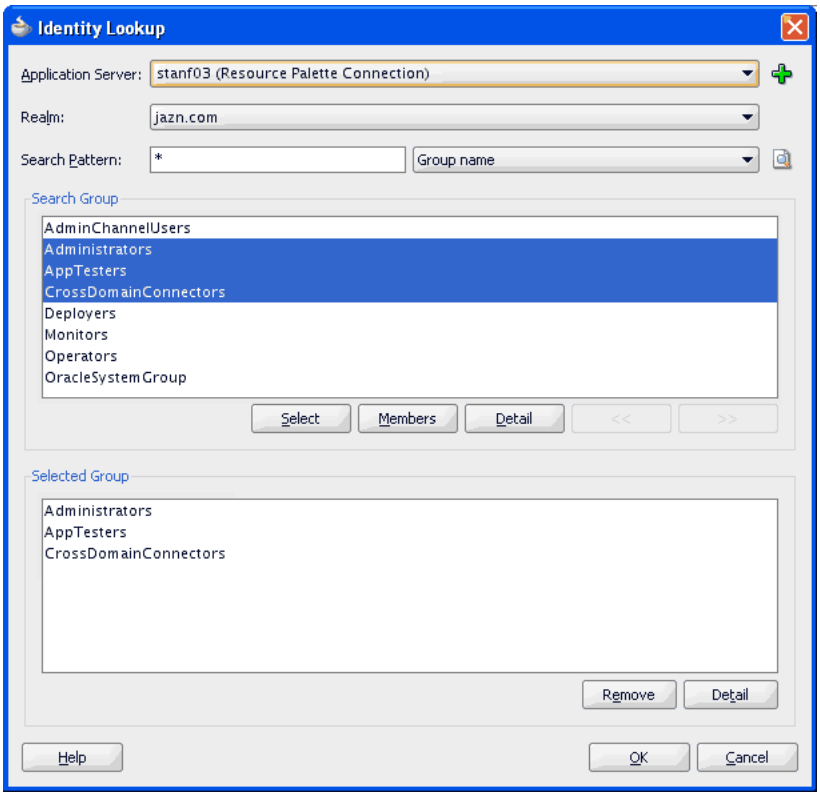
   A second Configure Task Content Access dialog displays.
5. From the dropdown list, select **Group**, as shown in Figure 26–46.

Figure 26–46 Configure Task Content Access Dialog (2)



   The Identity Lookup dialog displays.
6. Select an application server, realm, and appropriate groups, as shown in Figure 26–47.

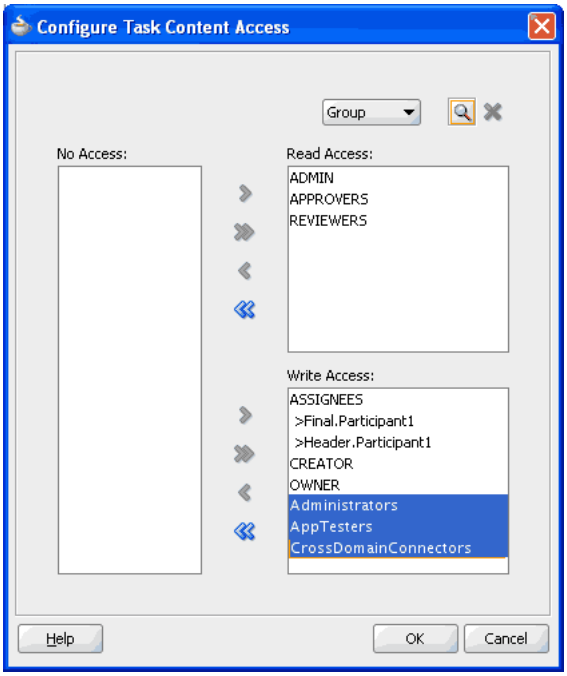
Figure 26–47 Identity Lookup Dialog



7. Click **OK**.

   The selected groups are added to the access rule, as shown in Figure 26–48.

Figure 26–48 Groups Added to Access Rule

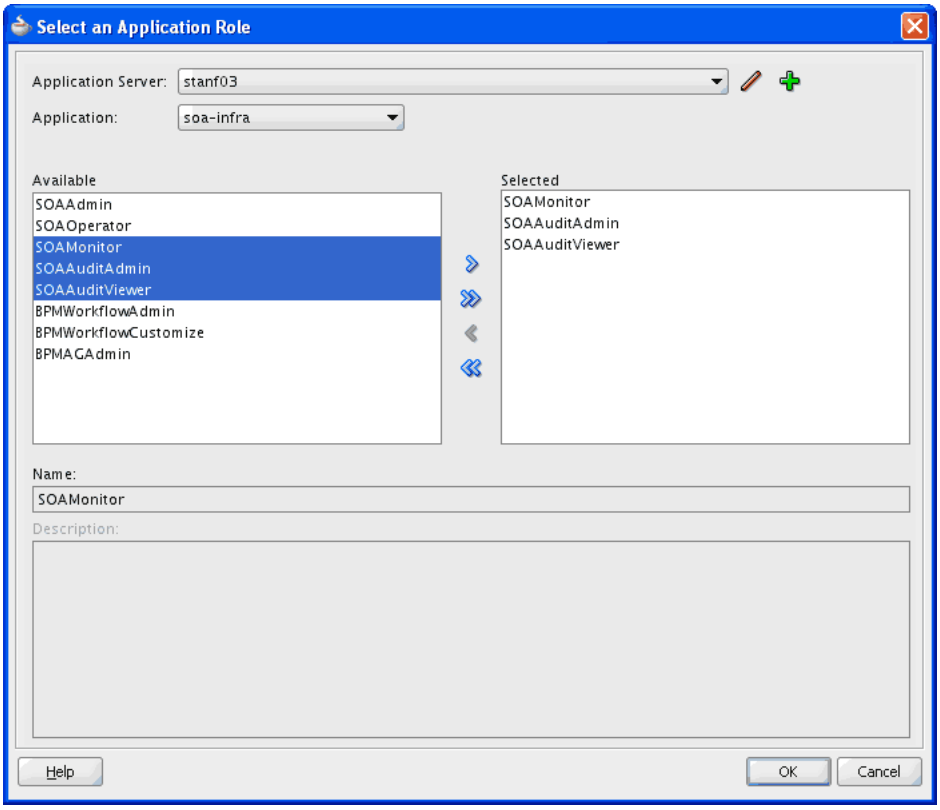


8. Click **OK** to close the dialog.

Use the same procedure to define access rules for Application Groups, with the following exceptions:

- Click the **Task Actions** tab to select it.
- Select **Application** from the dropdown list.
- Select application roles to include in the access rule from the Select an Application Role dialog, as shown in Figure 26–49.

Figure 26–49 Select an Application Role Dialog



For more information, see the section "Specifying Access Rules on Task Content" in the chapter "Designing Human Tasks" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

26.4 Using the End-to-End Approval Management Samples

Table 26–7 shows the end-to-end workflow examples included in the `ORACLE_HOME\samples\soa-infra\workflow` directory.

In addition to the demonstration features listed in the table, all samples show the use of worklist applications and workflow notifications.

Table 26–7 End-to-End Samples

Sample	Description	Location
Expense Line Approval	Illustrates line-level approval with approval policy defined.	`ORACLE_HOME\samples\soa-infra\workflow\amx\amx-101-expense-line`
Employee Hiring	Illustrates ad-hoc insertion capabilities for an approval having two stages - Approval Group List Builder in "Order" voting regime and a Supervisory list builder.	`ORACLE_HOME\samples\soa-infra\workflow\amx\amx-102-hiring-approval-group`
Purchase Order Approval	Illustrates the Purchase Order approval scenario with header and line-level approvals.	`ORACLE_HOME\samples\soa-infra\workflow\amx\amx-103-purchaseOrder-2dimensions`
Employee Transfer	Illustrates the Employee Transfer scenario from one team to another through parallel job level participants.	`ORACLE_HOME\samples\soa-infra\workflow\amx\amx-104-employee-transfer`
Self Approval	Illustrates how to implement self-approval through auto-action rules.	`ORACLE_HOME\samples\soa-infra\workflow\amx\amx-105-self-approval`
Position List Builder	Illustrates the use of the Position list builder.	`ORACLE_HOME\samples\soa-infra\workflow\amx\amx-108-position-list`

26.5 Using Approval Management Features of the Oracle BPM Worklist and Workspace

The Oracle Business Process Management (BPM) Worklist enables users to perform various approval-management tasks, including the following:

- Use task forms to manage tasks
- Define mapped-attribute labels
- Administer approval groups
- Use task configuration to configure approval rules
- Use task listing regions and portlets
- Use the task history region to preview approvers

26.5.1 How to Use Task Forms

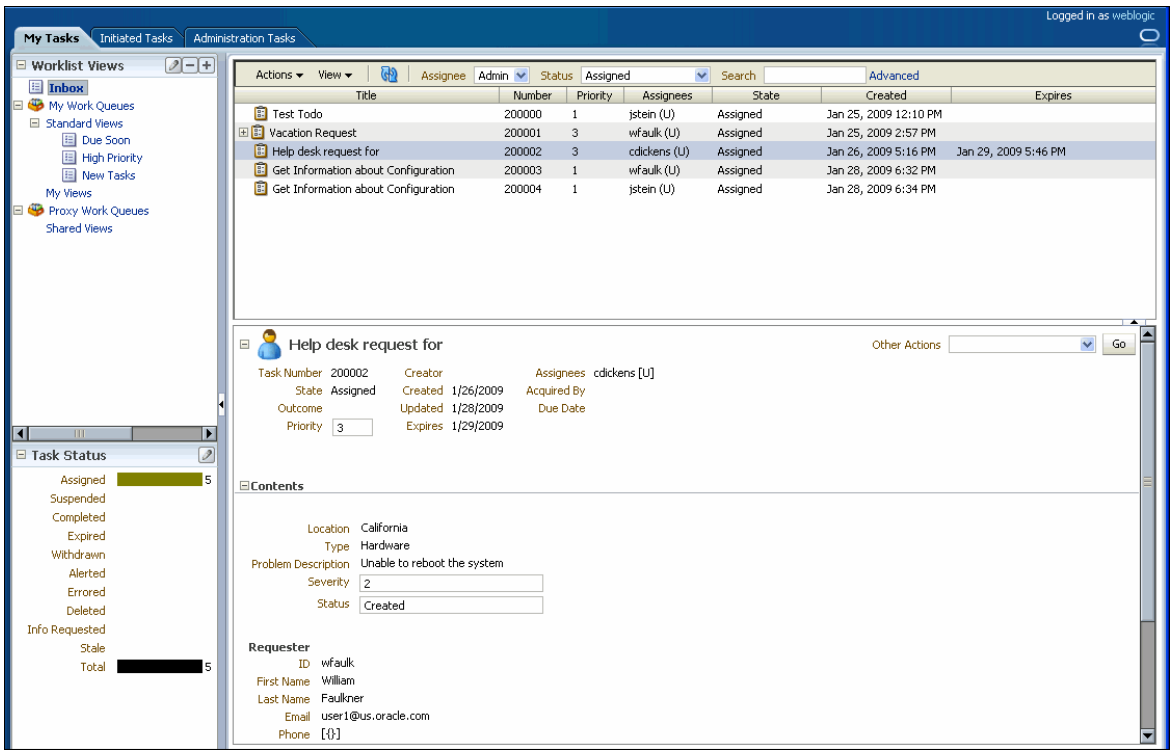
The human workflow service creates tasks for users to interact with the business process. Each task has two parts—the task metadata and the task form. The task form is used to display the contents of the task to the user's worklist.

The task form is designed in JDeveloper with Oracle Application Development Framework (Oracle ADF). For more information, see "Designing Task Display Forms for Human Tasks," in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Oracle BPEL Worklist Application displays all worklist tasks that are assigned to a user or a group. When a worklist user drills down into a specific task, the task display form renders the details of that task. For example, an expense approval task may show a form with line items for various expenses, and a help desk task form may show details such as severity, problem location, and so on.

The sections that follow describe the task form in the Worklist Application that users with administrator rights use to manage tasks. Figure 26–50 shows what a task form looks like.

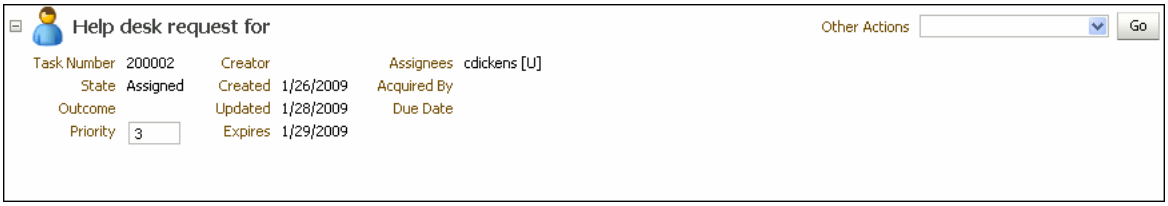
Figure 26–50 Task Form in the Worklist Application



26.5.1.1 Header View

The **Header** view, shown in Figure 26–51, is created in JDeveloper using the header drop handler.

Figure 26–51 Header



By default, the drop handler includes the **Header** fields listed in Table 26–8. However, you can include or remove any of the fields based on the use case.

Table 26–8 Header Fields

Field	Description
Task Number	The number automatically assigned to the task.
State	The current status of the task.
Outcome	The user outcome. For examaple, if the user clicks **Approve**, the Outcome field displays "Approve."

Table 26–8 (Cont.) Header Fields

Field	Description
Priority	The priority level assigned to the task.
Creator	The user who originated the task.
Created	The task's creation date.
Updated	The date the task was last updated.
Expires	The date the task expires.
Assignees	The name(s) of the administrator(s) to whom the task has been assigned.
Acquired By	The name of the user who claims the task before any action is taken on it.
Due Date	The date by which the task is due.

The **Header** also contains custom and system actions. *Custom actions* are those that depend on task metadata outcomes. For example, if the metadata contains Approve and Reject outcomes, then **Approve** and**Reject** appear in the **Header** as custom actions. If the metadata contains more than two outcomes, then the custom actions appear in the **Header** as a dropdown list instead of separate buttons. In Figure 26–51, **Approve** and **Reject** appear as separate buttons. This indicates that the task metadata includes Approve and Reject outcomes.

System actions, such as Escalate, Suspend, and Resume, always appear in a dropdown list. The actions that appear depend on what the user is doing. For example, after a task has been initiated it can be withdrawn. Subsequently, if a user logs into the Worklist Application to view the details of an initiated task "Withdraw" appears in the list containing the available actions.

Table 26–9 lists all the actions the administrator can perform from the **Header** and their descriptions.

Table 26–9 Header Actions

Action	Description
Reject	Enables the administrator to reject the task.
Approve	Enables the user to approve the task.
Delegate	Enables the administrator to assign a task to another person to act on his or her behalf.
Reassign	Enables the administrator to transfer a task to another administrator.
Suspend	Enables the administrator to suspend the task to work on it at a later time.
Resume	Enables the administrator to resume a suspended task.
Withdraw	Enables the administrator to withdraw an initiated task.
Escalate	Enables the administrator to escalate a task to a supervisor.
Delete	Enables the administrator to delete a task. Appears for all "To Do" tasks.
Purge	Enables the administrator to purge a task. Appears for all "To Do" tasks.

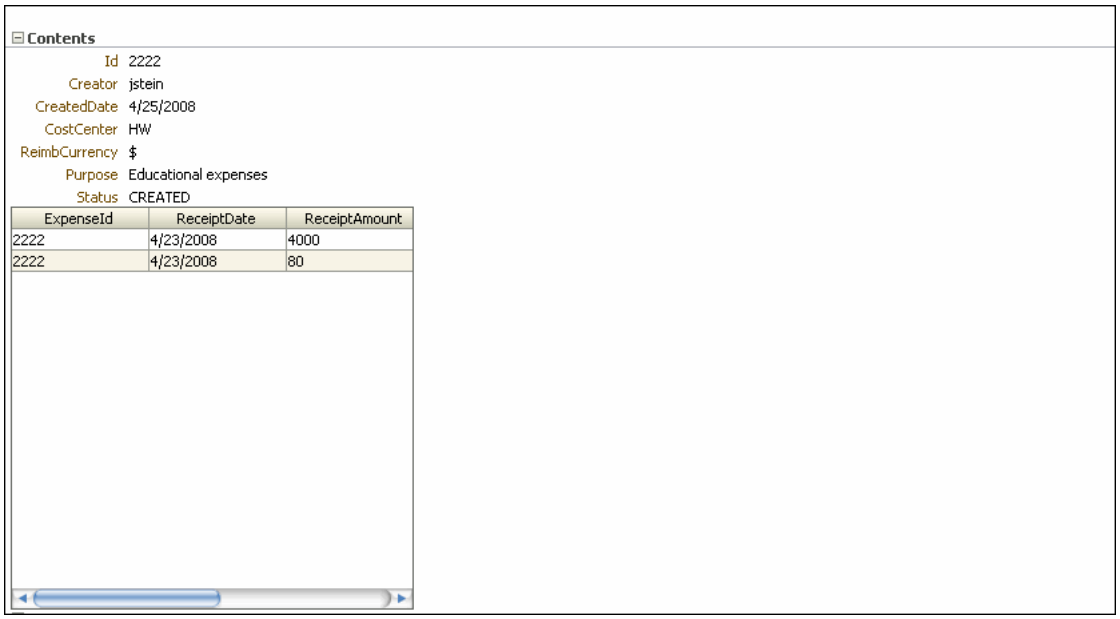
Table 26–9 (Cont.) Header Actions

Action	Description
Go	Enables the administrator to "go" to the action selected from the dropdown list.
Save	Enables the administrator to save the task.

26.5.1.2 Task Payload View

The **Task Payload** view, shown in Figure 26–52, displays the details of the task parameters and provides the ability to update them. Any parameter that has been designed based on SDOs also can be viewed here. While the values of the general parameters are based on what was passed to the task or update by any user, the values of the SDO-based parameters reflect the current value of the business object as it was updated in the underlying application.

Figure 26–52 Task Payload

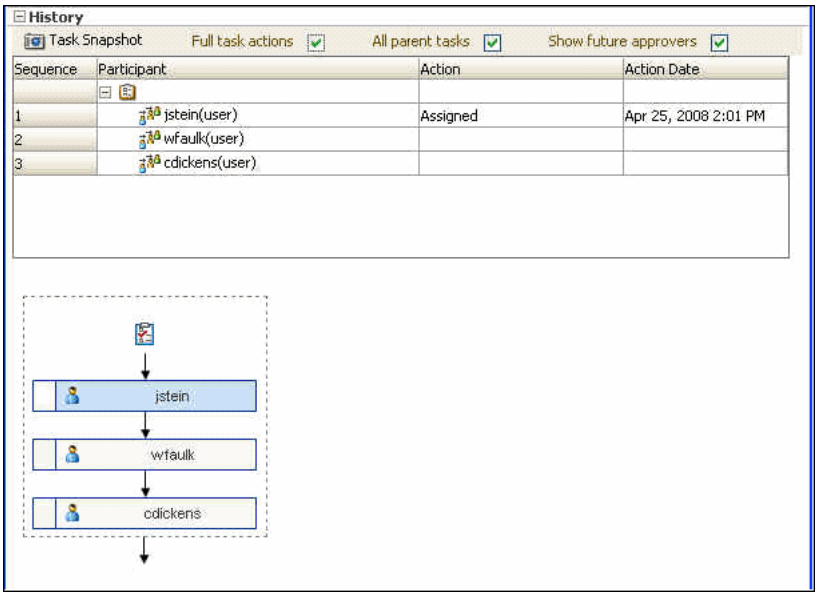


For more information, see "Designing Task Display Forms for Human Tasks" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

26.5.1.3 Task History View

The **Task History** view, shown in Figure 26–53, provides a graphical and tabular view of events in the task life cycle. In addition, if the Edit Approver Configuration option was selected in the designer, special controls in the tabular view that allow futire approvers to be edited are available. Any approver added manually can be deleted and new approvers can be added. Approver changes made in the tabular view are immediately reflected in the graphical view. When the task is saved or a custom action, such as approve or reject, is performed on the task, all approver-related changes also are saved.

Figure 26–53 Task History



If the **Allow all participants to edit future participants** option is selected while configuring the approval task, the history region displays additional actions that allow a participant to edit the future participants list. For more information, see Section 26.3.3, "Specifying General Information."

Figure 26–54 shows the addition of the **Apply** and **Reset** buttons.

Figure 26–54 Task History - Additional Actions

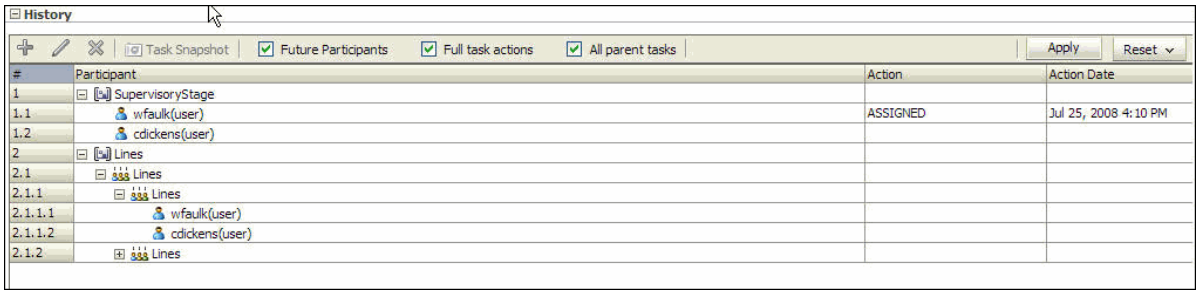


Table 26–10 describes all additional approval-task actions.

Table 26–10 Edit Future Participants List Actions

Action	Description
Add	Enabled when the user selects a future participant. When this option is selected, the Add Participant dialog opens and the user can insert ad- hoc participants.
Edit	Enabled when the user selects a participant that has been inserted ad hoc. When this option is selected, the Edit Participant dialog opens and user can move the position of an inserted participant.
Delete	Enabled when the user selects a participant that has been inserted ad hoc. When this option is selected, the corresponding participant is deleted.
Apply	Persists the edits to the future participant list.

Table 26–10 (Cont.) Edit Future Participants List Actions

Action	Description
Reset	Resets the edits from the future participant to a system-generated list.

Table 26–11 lists the actions the administrator can perform from the **Task History** view and their descriptions.

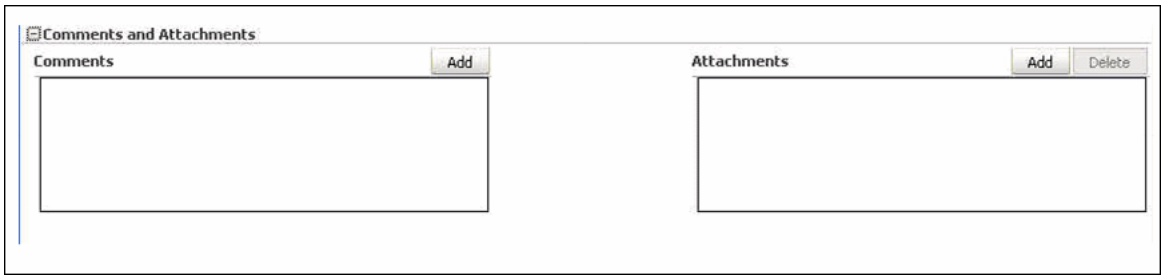
Table 26–11 Task History Actions

Action	Description
Task Snapshot	Displays the task details for the selected version.
Full task actions	If checked, displays the full task-action history. If unchecked, displays only action history. By default, the box is unchecked.
All parent tasks	If checked, displays the parent task in a sub-task view. If unchecked, displays only the sub-task history. By default, the box is checked.
Show future approvers	If checked, displays the future approver with history. By default, the box is unchecked.

26.5.1.4 Comments and Attachments View

The **Comments and Attachments** view, shown in Figure 26–55, is created in JDeveloper using the task data control drop handler. It includes a text-entry field in which to enter comments about the task, and the functionality to attach supporting documents.

Figure 26–55 Comments and Attachments



26.5.2 How to Create Mapped Attribute Labels

Mapped attributes are those that you can use for storing use-case-specific data, such as data extracted from a task's payload. You can view and create mapped attribute labels on the server using the Worklist Application.

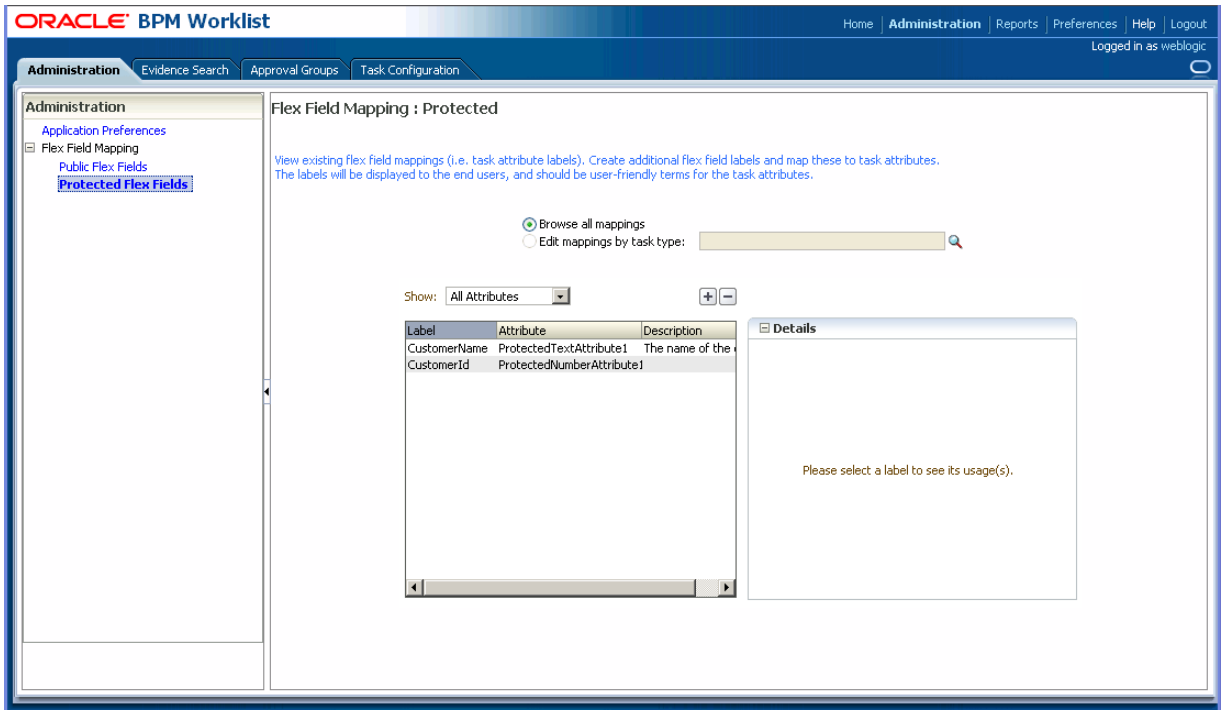
> **Note:** You must have the *workflow.mapping.protectedFlexfield* privilege to create protected flexfield attributes. The default administrative user, *weblogic*, has this privilege.

For more information, see Section 26.3.5, "Specifying Mapped Attributes."

To view attribute labels:

1. From the BPM Worklist Application main page, click the "Administration" link in the upper-right corner of the page.
2. In the **Administration** pane, click the "Protected Flex Fields" link. A page similar to the one shown in Figure 26–56 displays.

Figure 26–56 Flexfield Mapping: Protected

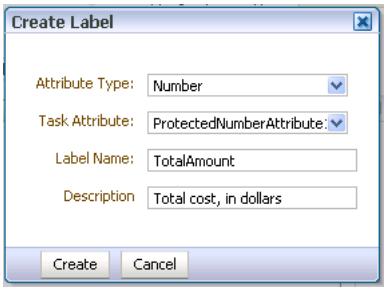


The page displays a list of existing attribute labels. You can filter the list by selecting an attribute type from the dropdown list. Clicking a specific label displays the list of mappings the attribute uses in the **Details** panel.

To create an attribute label:

1. Click the **Add** (+) button. The Create Label dialog displays, as shown in Figure 26–57.

Figure 26–57 Create Label Dialog



2. Select an attribute type and task attribute from the dropdown lists.
3. Enter a unique label name and a description.

4. Click **Create**. The label is created and is made available for mapping in task components.

To delete an attribute label:

To delete an attribute label, first select it from the list of attribute labels. Then click the **Delete** (-) button.

Note: Attribute labels can be deleted only if they are not used in any mappings.

26.5.2.1 Importing and Exporting Attribute-Label Definitions

If attribute labels have been defined on one server and must be re-created on another, you can use the user metadata migration utility to export a list of protected attribute labels from the server on which they were defined to an XML file. The utility then can deploy the attribute labels from this file to a new server. This eliminates the necessity to manually re-create the attribute labels manually in the Worklist Application. For more information, see Section 26.6, "Using the User Metadata Migration Utility."

26.5.2.2 Internationalizing Attribute Labels

When attribute labels are displayed to end users, for example in the task listing page of the Worklist Application, the label name specified when the label was created is used. In cases where users of different nationalities may see the label, a translation of the label name appropriate to the Worklist Application user's locale can be displayed instead. Translations of attribute labels can be customized using the **WorkflowLabels.properties** resource bundle.

For more information, see "Internationalization of Attribute Labels" in the chapter "Introduction to Human Workflow Services" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

26.5.3 Administering Approval Groups

An approval group consists of a name and a predefined set of users configured to act on a task in a certain pattern. This pattern is similar to a human workflow routing slip pattern where users can act on tasks in serial or parallel. An approval group also can contain a nested approval group in the pattern.

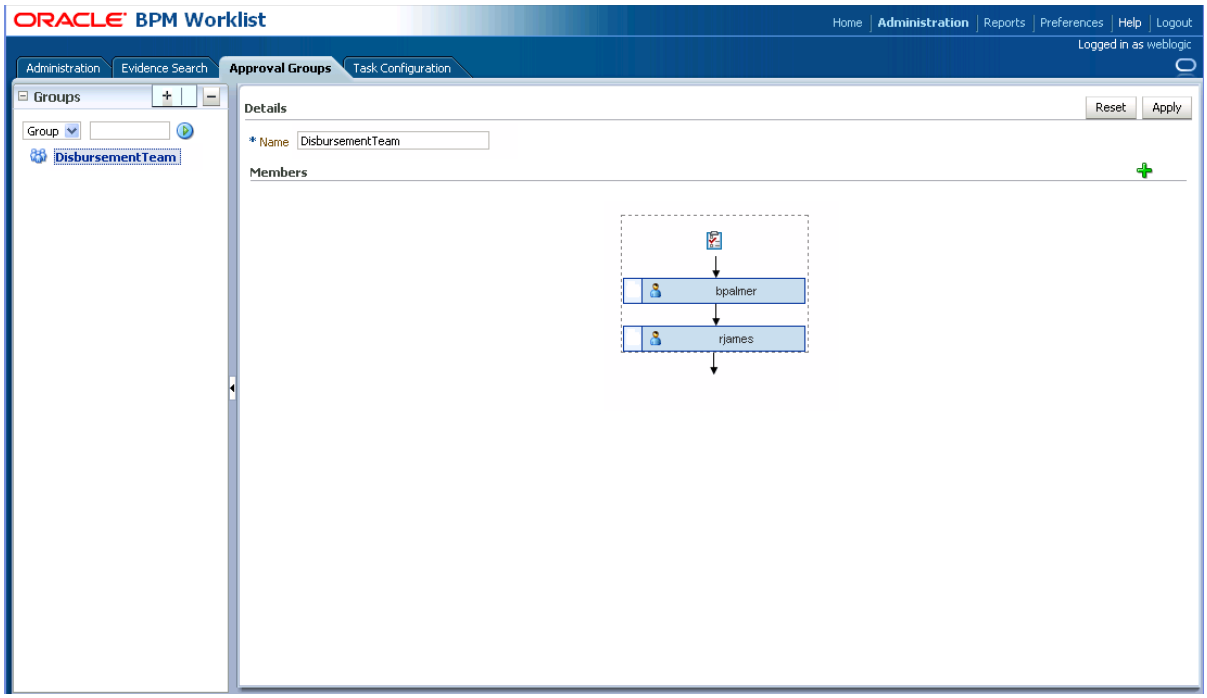
The name of an approval group is needed when specifying the approval group list builder as discussed in Section 26.3.6.3.1, "How to Model an Approval Groups List Builder." The pattern configured in the approval group is used by default to order the users who must act on the task. However, when creating the list builder, the default pattern can be overridden by specifying the voting regime.

The sections that follow describe the Worklist Application user interface that enables users with administrator rights to manage approval groups.

26.5.3.1 How to View Approval Groups

From the Oracle BPM Worklist home page, click the **Approval Groups** tab. A page similar to the one shown in Figure 26–58 appears.

Figure 26–58 Worklist Application: Approval Groups Tab



The graphic shows that the "DisbursementTeam" approval group has two users, "bpalmer" and "rjames." The users act on a task in a specific sequence configuration.

26.5.3.2 How to Search for an Approval Group

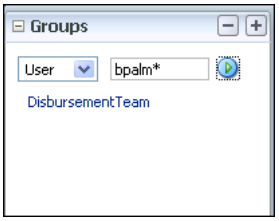
You can search for an approval group either by user name or group name.

To search by user name:

1. In the left navigation pane, select **User** from the dropdown list.
2. Enter the full username for the user in the text-entry field. (You also can perform a wildcard search (*) with a partial username.)
3. Click the action (>) button.

   A list of all approval groups to which the user belongs displays in the left navigation pane, as shown in Figure 26–59.

Figure 26–59 User Name Search Results



Clicking on the approval group name refreshes the details pane on the right with the structure of that group.

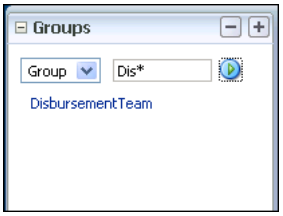
To search by group name:

1. In the left navigation pane, select **Group** from the dropdown list.

2. Enter the full group name in the text-entry field. (You also can perform a wildcard search (*) with a partial group name.)
3. Click the action (>) button.

   A list of all matching approval groups displays in the left navigation pane, as shown in Figure 26–59.

Figure 26–60 Group Name Search Results



Clicking on the approval group name refreshes the **Details** pane on the right with the structure of that group.

26.5.3.3 How to Add a Static Approval Group

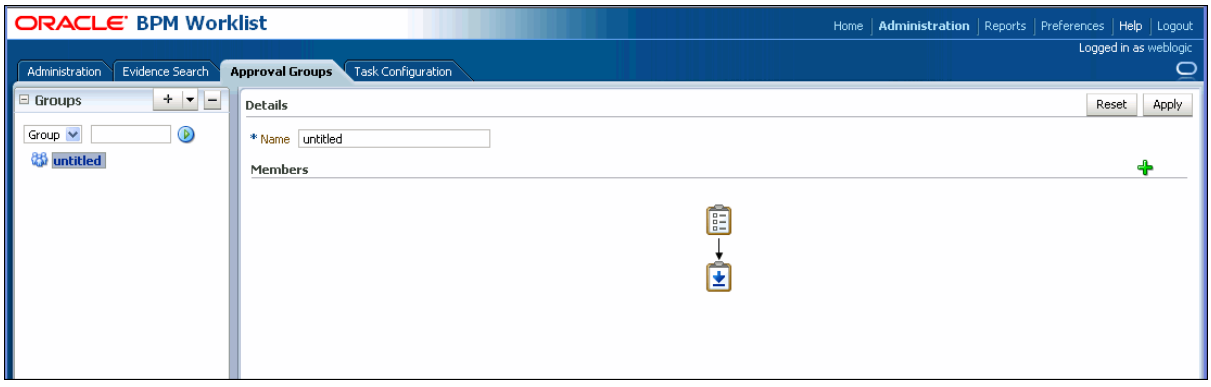
The following procedure explains how to add a static approval group.

1. In the left navigation pane, click the **Add** (+) button and select **Create Static** from the dropdown list, as shown in Figure 26–61.

Figure 26–61 Create Approval Group: Select Static Group

A page similar to the one shown in Figure 26–62 appears.

Figure 26–62 Create Static Approval Group: Details



2. Enter a new name for the group.
3. Click **Apply**.

You now can add members to the new approval group.

26.5.3.4 How to Add a New Member to a Static Approval Group

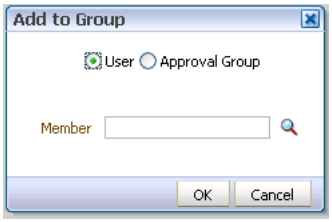
Members of a static approval group either can be users or other approval groups. The following procedure explains how to add a new user member to an approval group.

1. From the page shown in Figure 26–62, click the **Add** (+) icon.

   The other icons enable you to edit, delete, and reorder members in the approval sequence.

   The Add to Group pop-up dialog appears, as shown in Figure 26–63.

Figure 26–63 Add to Group Dialog

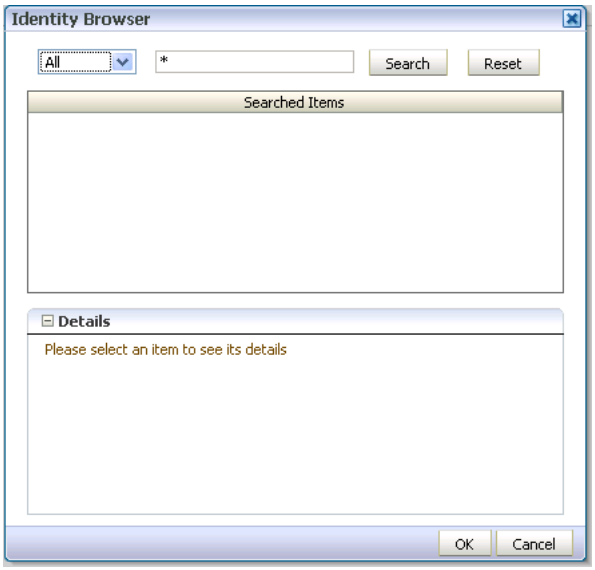


2. Select **User**.
3. Select a user using one of these options:
   - Enter a full user name and click **OK**.

     The dialog closes and the new member appears in the **Members** section of the **Details** pane.
   - Click the magnifying glass to search for a user.

     If you click the magnifying glass, an Identity Browser pop-up dialog appears, as shown in Figure 26–64.

Figure 26–64 Identity Browser: Add User

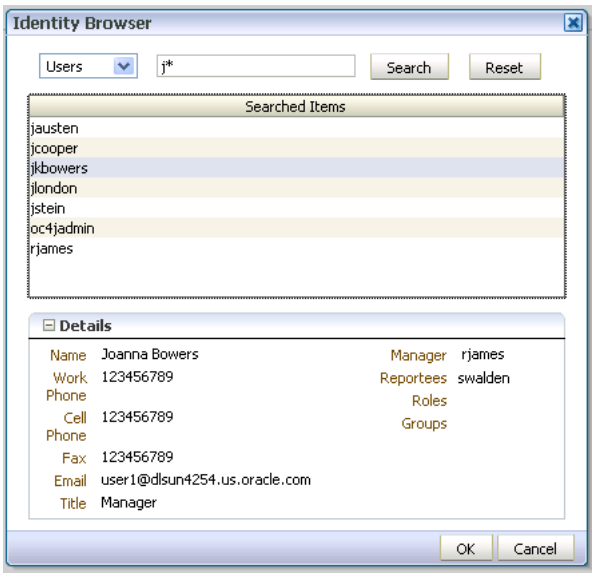


4. Select **Users** from the dropdown list.

5. Enter a full name in the text-entry field and click **Search**. (You also can perform a wildcard search (*) with a partial user name.)

   The Identity Browser dialog refreshes and the search results appear, as shown in Figure 26–65.

Figure 26–65 Identity Browser: Search Results



6. Choose a user from the list.

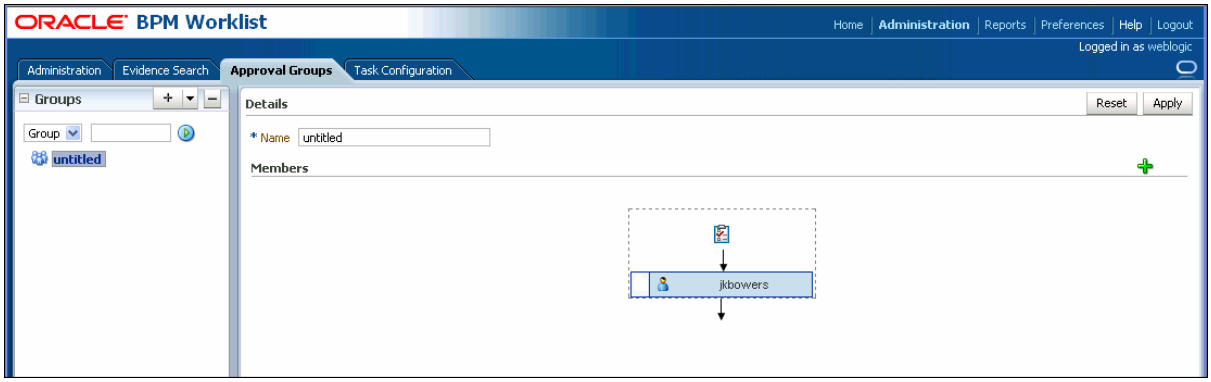
   The details for that user appear in the **Details** section of the dialog.

7. Click **OK**.

8. Click **OK** again to close the Add to Group dialog.

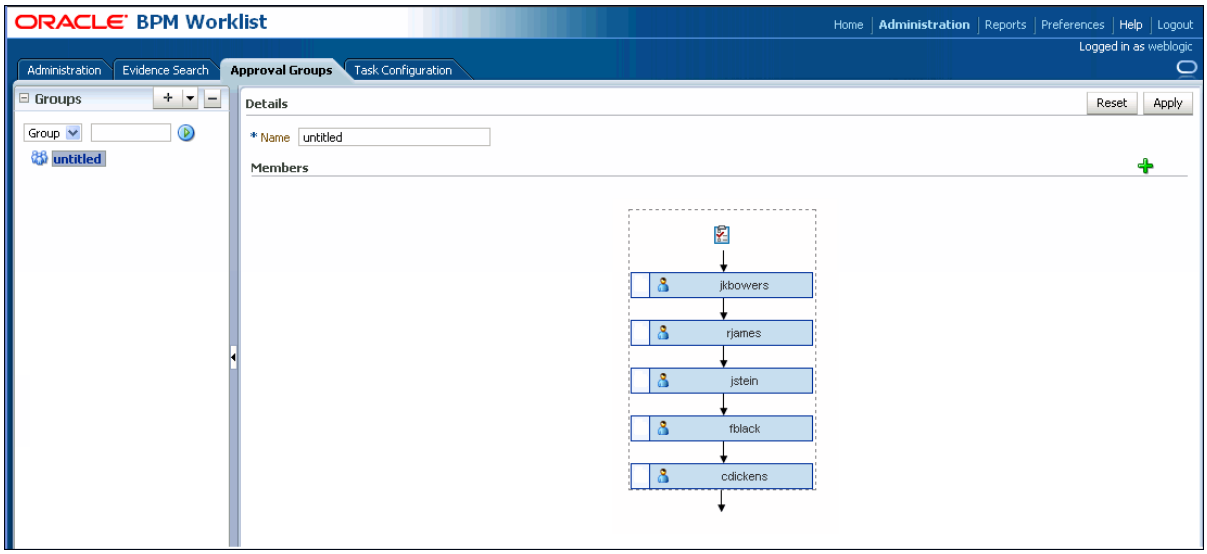
   A node representing the selected user appears in the approval group structure in the **Members** section of the **Details** pane, as shown in Figure 26–66.

Figure 26–66 Approval Group Structure



You can add more members to the approval group by repeating the steps above. The resulting approval group structure looks similar to the one shown in Figure 26–67.

Figure 26–67 Approval Group Structure: Multiple Members



26.5.3.5 How to Delete a Member from an Approval Group

The following procedure explains how to delete a member from an approval group.

1. Choose the appropriate member node from the approval group structure.
2. Click the **Delete** icon.

   The approval group structure refreshes and the member node has been deleted.

26.5.3.6 How to Move an Approval Group's Members

The following procedure explains to to change the sequence order of an approval group.

1. Choose a member node to move.
2. Use the Push Member Up (^) and Push Member Down (v) icons to move the member to the desired location.

Figure 26–68 and Figure 26–69 show the positions of rjames and jstein before and after a move.

Figure 26–68 Before Move

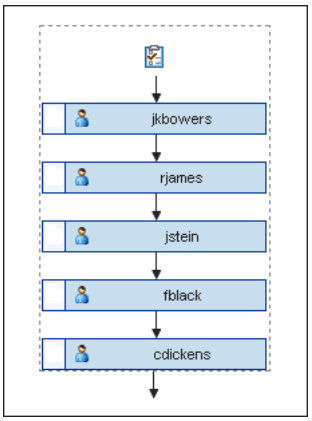
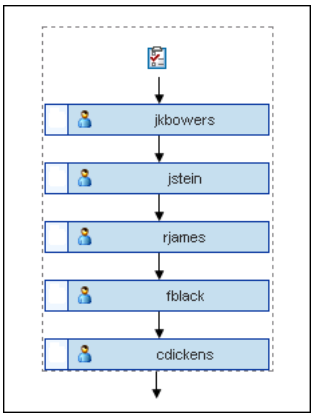


Figure 26–69 After Move



26.5.3.7 How to Nest Approval Groups

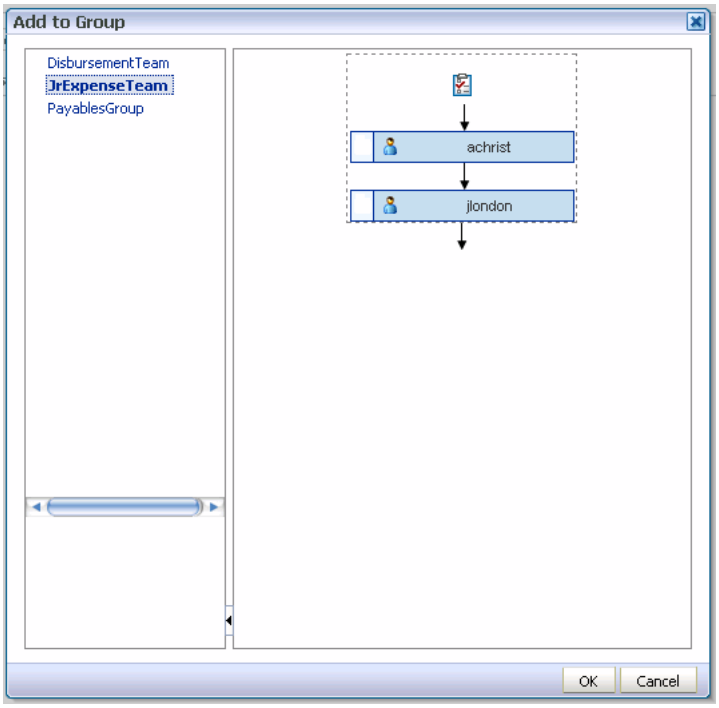
The following procedure explains how to nest approval groups, that is, add an approval group to another approval group.

1. Click the **Add** icon.
2. Select **Approval Group**.
3. Click the magnifying glass.

   Another Add to Group dialog appears.
4. From the left pane, choose the approval group you want to add.

   Its structure appears in the right pane, as shown in Figure 26–70.

Figure 26–70 Choose Approval Group to Nest

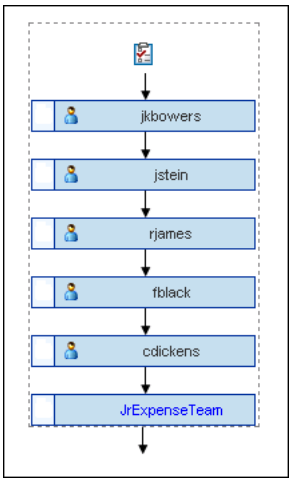


5. Click **OK**.

6. Click **OK** again to close the Add to Group dialog.

   The new approval group appears in the approval group's structure, as shown in Figure 26–71.

Figure 26–71 Nested Approval Groups



26.5.3.8 How to Rename an Approval Group

The following procedure explains how to rename an approval group.

1. Enter the new name of the approval group in the **Name** field.
2. Click **Apply**.

The name change also is reflected in other approvals groups in which this approval group is nested.

26.5.3.9 Using Dynamic Approval Groups

Dynamic Approval Groups provide a way to create approval groups through a custom Java class at run time. This requires the following:

- Writing a custom dynamic approval group class for the custom implementation by the developer
- Registering the custom dynamic approval group using the worklist apps UI by the IT department
- Making the class file available in a globally well known directory that is part of the SOA class path

26.5.3.9.1 How to Write a custom Dynamic Approval Group Class to define a dynamic approval group, the customer must define an implementation class using the interface file `IDynamicApprovalGroup.java`, defined by AMX for dynamic approval groups in the package *oracle.bpel.services.workflow.task*. This contains only one public method that gets the approval group members. The Task object is the only input parameter. The primary key list can be obtained from the task *task/systemAttributes/collectionTarget*.

Example 26–6 Implementation Class

```
************** IDynamicApprovalGroup.java ******************
```

```
public interface IDynamicApprovalGroup {

    /**
        *  Get members of this dynamic approval group
        *  @param task Property bag containing information required to generate
             the approver list
        *  @return list of IApprovalListMember including sequence, member, member_
             type; null for empty group
        *  The primary key list can be obtained from task:
             task/systemAttributes/collectionTarget
       */

   public List getMembers(Task task )
       throws WorkflowException;
}
*******************************************************
```

Figure 26–72 shows a code snippet for a sample dynamic approval group class.

Figure 26–72 Code for Dynamic Approval Group Class

```
package oracle.bpel.services.workflow.repos.test;

import java.util.ArrayList;
import java.util.List;
import oracle.bpel.services.workflow.WorkflowException;
import oracle.bpel.services.workflow.task.IDynamicApprovalGroup;
import oracle.bpel.services.workflow.runtimeconfig.impl.RuntimeConfigUtil;
import oracle.bpel.services.workflow.runtimeconfig.model.ApprovalGroupMember;
import oracle.bpel.services.workflow.task.model.Task;
import oracle.bpel.services.workflow.IWorkflowConstants;

public class SampleDynAprGrp1  implements IDynamicApprovalGroup {
    public List getMembers(Task task)
        throws WorkflowException
     {
        // Add logic later to do conditional return of different group mebers b
ased on some task attribute
        List approversList=new ArrayList();
        ApprovalGroupMember taskAssignee1 = RuntimeConfigUtil.getFactory().crea
teApprovalGroupMember();
        taskAssignee1.setMember("jlondon");
        taskAssignee1.setType(IWorkflowConstants.IDENTITY_TYPE_USER);
        taskAssignee1.setSequence(1);
        approversList.add(taskAssignee1);
        ApprovalGroupMember taskAssignee2 = RuntimeConfigUtil.getFactory().crea
teApprovalGroupMember();
        taskAssignee2.setMember("wfaulk");
        taskAssignee2.setType(IWorkflowConstants.IDENTITY_TYPE_USER);
        taskAssignee2.setSequence(2);
        approversList.add(taskAssignee2);
        ApprovalGroupMember taskAssignee3 = RuntimeConfigUtil.getFactory().crea
teApprovalGroupMember();
        taskAssignee3.setMember("jstein");
        taskAssignee3.setType(IWorkflowConstants.IDENTITY_TYPE_USER);
        taskAssignee3.setSequence(3);
        approversList.add(taskAssignee3);
        ApprovalGroupMember taskAssignee4 = RuntimeConfigUtil.getFactory().crea
teApprovalGroupMember();
        taskAssignee4.setMember("achrist");
        taskAssignee4.setType(IWorkflowConstants.IDENTITY_TYPE_USER);
        taskAssignee4.setSequence(4);
        approversList.add(taskAssignee4);
        return approversList;
     }
```

26.5.3.9.2 How to Register the custom Dynamic Approval Group Class For more information, see Section 26.5.3.9.4, "How to Add a Dynamic Approval Group."

26.5.3.9.3 How to Make the custom Dynamic Approval Group Class Available To make the class file available in a globally well-known directory that is part of the SOA class path, you must place your class files in the following WebLogic Server directory:

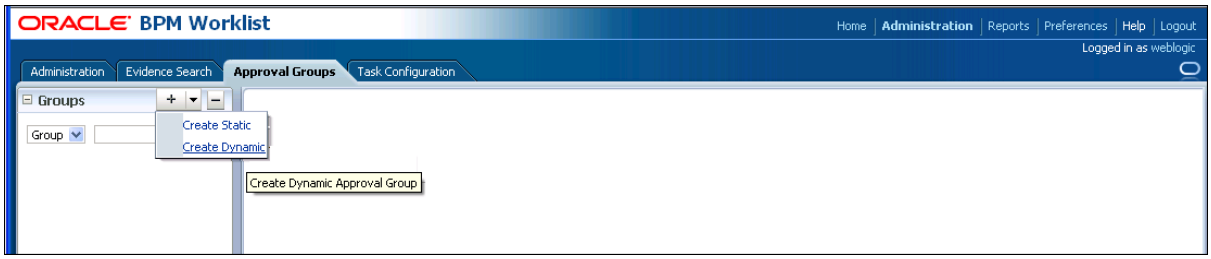
`$BEAHOME/AS11gR1SOA/soa/modules/oracle.soa.ext_11.1.1/classes`

For example, for the Java class *oracle.apps.DynamicAG* , the path would be `$BEAHOME/AS11gR1SOA/soa/modules/oracle.soa.ext_11.1.1/classes/oracle/apps/DynamicAG.class`. You **must** restart WebLogic Server after you place your class files there.

26.5.3.9.4 How to Add a Dynamic Approval Group The following procedure explains how to add a dynamic approval group approval group.

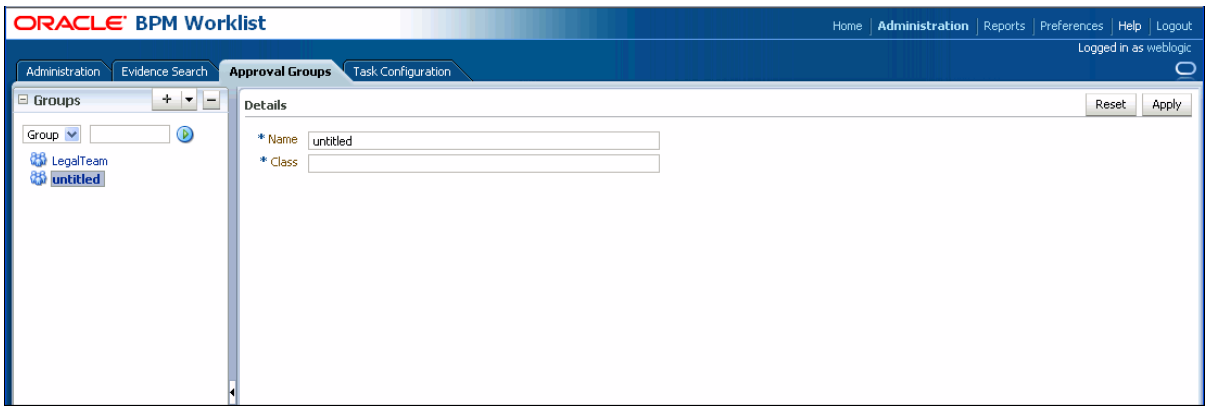
1. In the left navigation pane, click the **Add** (+) button and select **Create Dynamic** from the dropdown list, as shown in Figure 26–73

Figure 26–73 Create Approval Group: Select Dynamic Group



A page similar to the one shown in Figure 26–74 appears.

Figure 26–74 Create Dynamic Approval Group: Details



2. Enter a name and a class for the group.
3. Click **Apply** to save the group.

26.5.3.10 How to Delete an Approval Group

The following procedure explains how to delete an approval group.

1. In the left navigation pane, choose the approval group you want to delete.
2. Click the **Delete** (-) button, as shown in Figure 26–75.

Figure 26–75 Delete Approval Group

A confirmation dialog appears.

3. Click **OK**.

The approval group is deleted.

> **Note:** If the approval group you deleted is nested in other approval groups, it also is deleted from those parent groups.

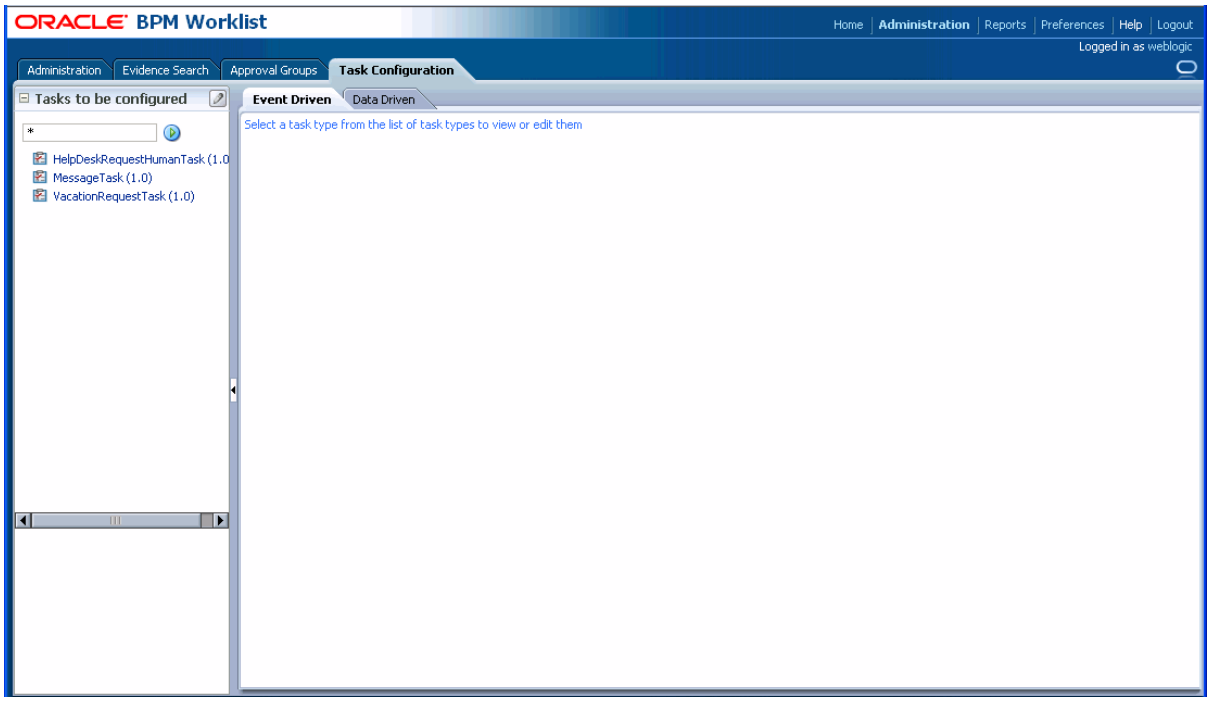
26.5.4 Using Task Configuration

Task Configuration in the Worklist Application lets business users and administrators review the rules that have been configured out of the box by the workflow designer. These pre-defined rules can be modified to customize the approval flow for a specific customer at any point in time based on the customer's applicable corporate policies.

For example, if a corporate policy requiring two levels of approvals for expense amounts greater than 1000 is changed to a policy requiring three levels, the customer can use this web-based application to change the rule rather than having its IT department first modify the rule in the underlying process and then deploy it again. Any change made to the rule is applied starting with the next instance; instances that are in progress use the current rule definitions.

Task Configuration enables you to edit the event driven and data-driven rules associated with an approval flow at run time; that is, when the workflow is deployed. The **Event Driven** and **Data Driven** tabs are accessible by clicking the main **Task Configuration** tab of the Administration section of the application. A screen similar to the one shown in Figure 26–76 displays.

Figure 26–76 Task Configuration: Main Tab



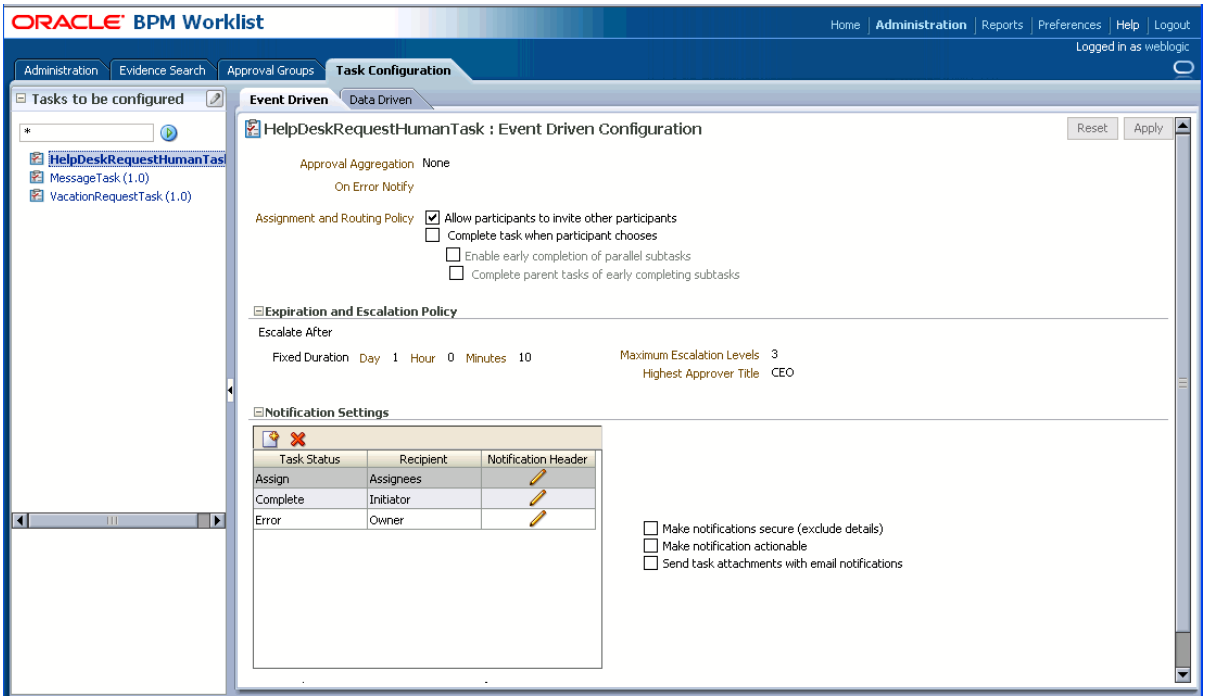
The **Tasks to be configured** panel on the left lists all workflow tasks that have been configured to use approval-flow rules. It also provides a search capability. In the view mode, the right panel displays the default configuration and rules for overriding the approval-flow list builder configuration. The rule configurations are displayed based on the stages defined in the approval flow.

26.5.4.1 How to Edit Event-Driven Settings

This section contains information about event-driven settings, that is, task metadata.

Figure 26–77 shows an event-driven task configuration page.

Figure 26–77 Task Configuration: Event Driven

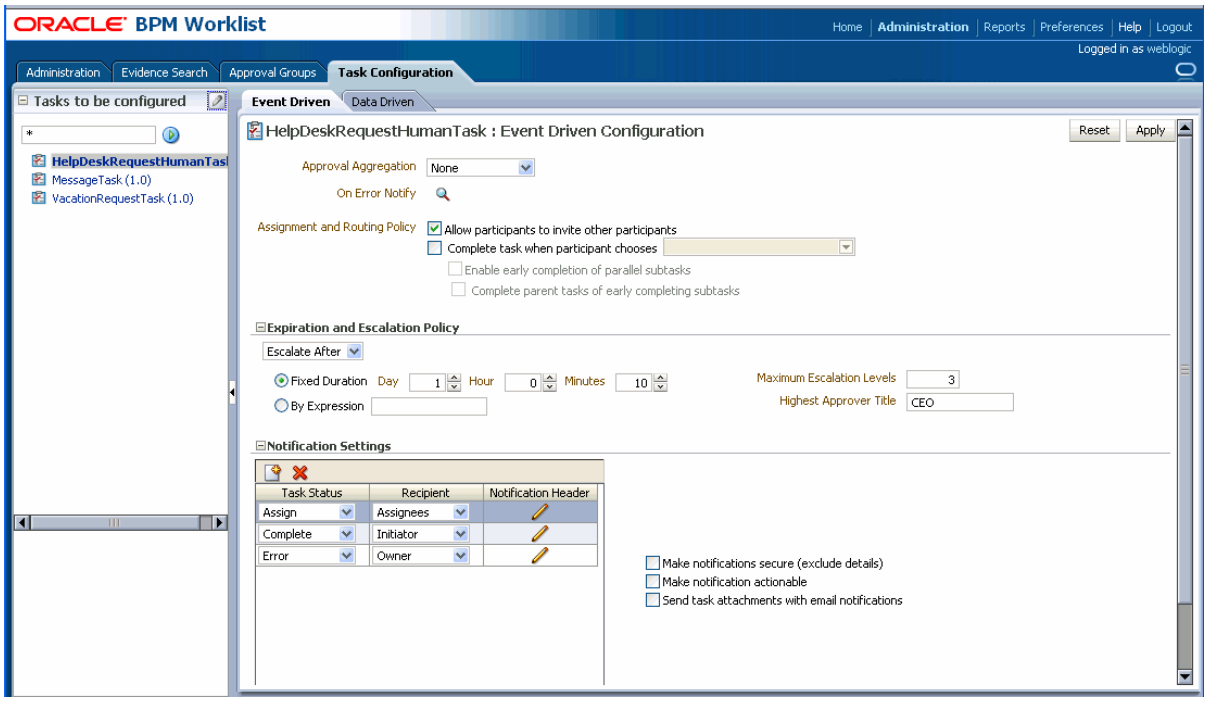


Use the following procedure to edit an event-driven setting.

1. Under the **Task Configuration** tab, click the **Event Driven** tab.
2. In the **Tasks to be configured** pane, select a task. Then click the **Edit** (pencil) icon.

   The main page refreshes in edit mode, as shown in Figure 26–78.

Figure 26–78 Task Configuration: Event Driven, Edit Mode



3. Make your changes and click **Apply** to save them
4. Click **Deploy** to deploy your changes.

WARNING: An improper or incomplete rules definition in a list-creation rule set can cause run-time errors. Errors can be caused by the following:

- **No rule was defined in the rule set.**
- **None of the conditions defined in the rule was met.**

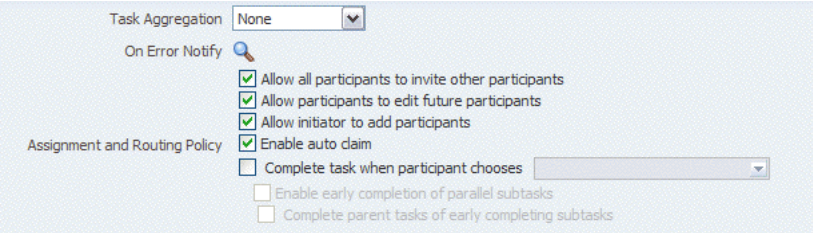
Ensure that rules are properly defined to handle all conditions.

26.5.4.1.1 How to Specify Routing Settings The **Event Driven** tab contains a limited set of the routing options available in the Human Task Editor.

Approval aggregation requirements can be any of the following, as shown in Figure 26–79:

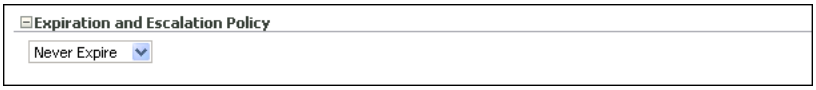
- None
- Once per task
- Once per stage

Figure 26–79 Routing Settings



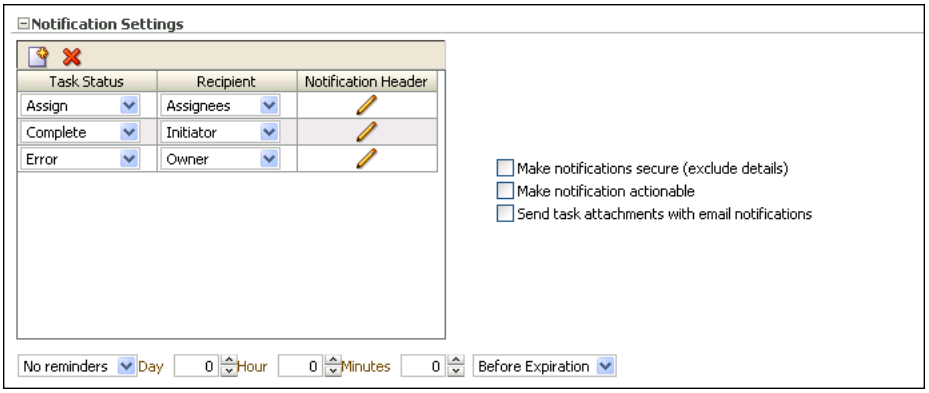
26.5.4.1.2 How to Define Expiration and Escalation Policies Defining expiration and escalation policies in the Worklist Application is very similar to how it is done in the Human Task Editor. For more information, see the section "How to Escalate, Renew, or End the Task" in the chapter "Designing Human Tasks" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Figure 26–80 Expiration and Escalation Policies



26.5.4.1.3 How to Specify Notification Settings Creating or updating notification settings for a task in the Worklist Application s very similar to how it is done in the Human Task Editor. For more information, see the section "How to Specify Participant Notification Preferences" in the chapter "Designing Human Tasks" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

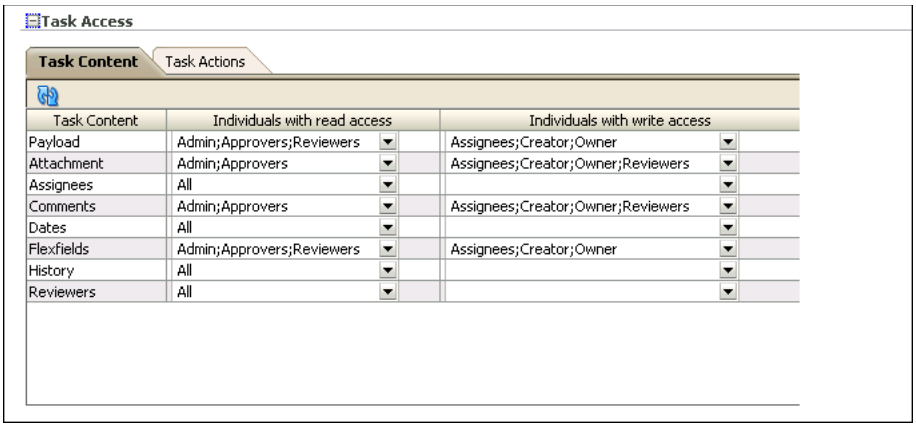
Figure 26–81 Notification Settings



26.5.4.1.4 How to Allow Task Access Access-rule settings can be set to control the actions a user can perform, and is very similar to how it can be done in the Human Task Editor. Content and action permissions can be specified based on the logical role of a user, such as creator (inititator), owner, assignee, and reviewers.

For more information, see Section 26.3.9.2, "How to Define Security Access Rules."

Figure 26–82 Task Access



26.5.4.2 How to Edit Data-Driven Settings

Use the following procedure to edit a data-driven setting, that is, a rule or condition.

1. Under the **Task Configuration** tab, click the **Data Driven** tab.
2. In the **Tasks to be configured** pane, select a task. Then click the **Edit** (pencil) icon.

   The right panel refreshes in edit mode, as shown in Figure 26–83 and Figure 26–84.

Figure 26–83 Task Configuration: Data Driven, Edit Mode (1)

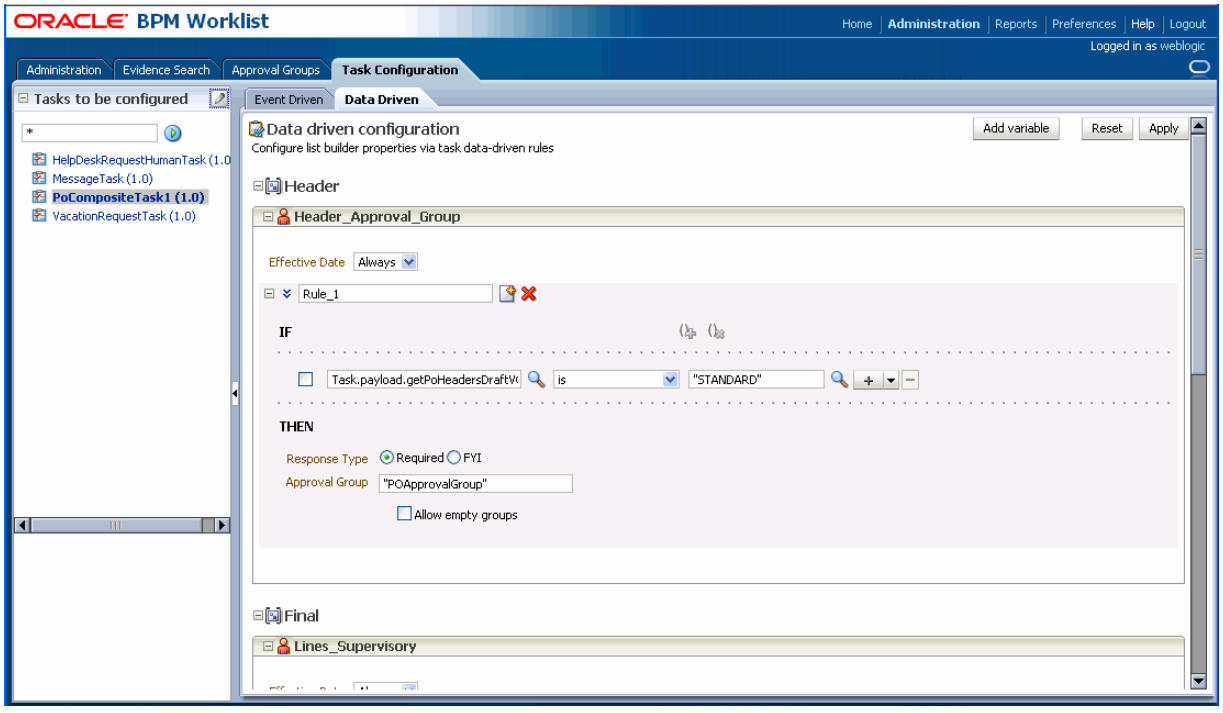
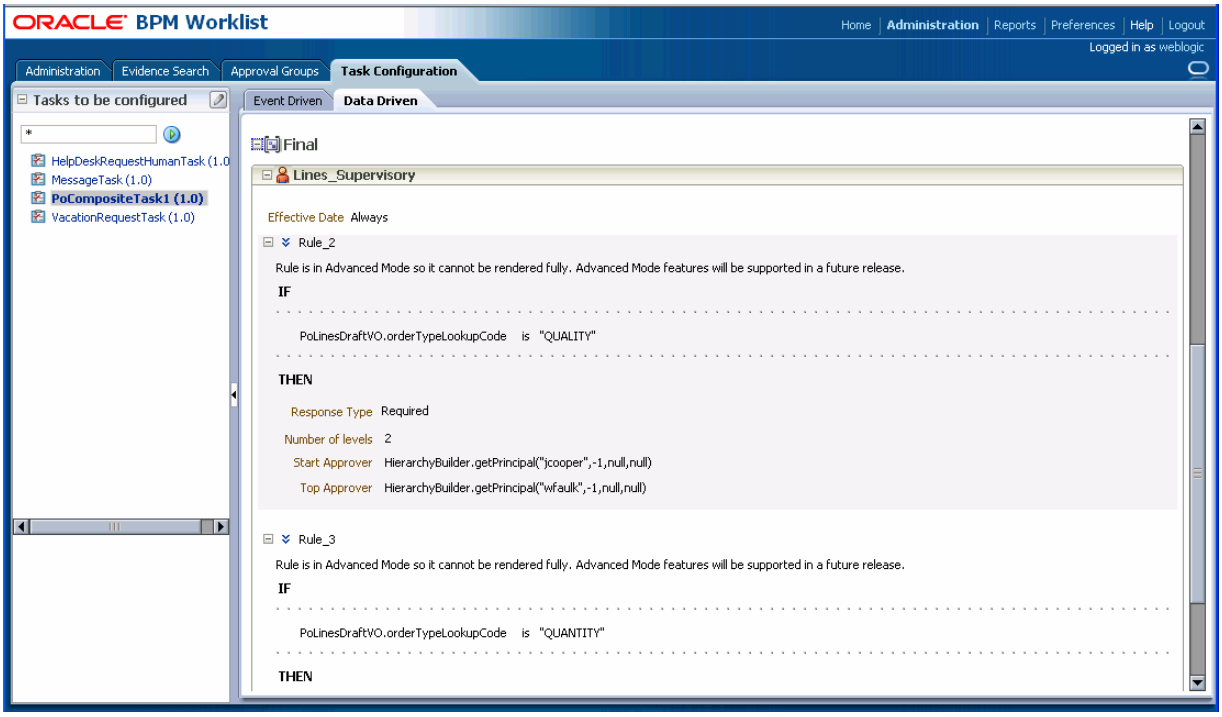


Figure 26–84 Task Configuration: Data Driven, Edit Mode (2)



3. Update the rule name.

   The rule name appears as the rule reason in the history graph. If you defined a resource bundle, then the rule name is used to obtain the resource bundle and a translated text is displayed.

4. Do any of the following:

   - Add
   - Update
   - Delete
   - Change assertions (which depend on the type of list builder for which the rule has been configured)
   - Add variables

5. Click **Apply** when you have finished making your changes.

   The changes are saved to the rule definitions in the rules dictionary.

6. Click **Deploy** to deploy the changes.

Section 26.5.4.2.1, "How to Add a Variable" and Section 26.5.4.2.2, "How to Define Conditions" discuss additional editing tasks.

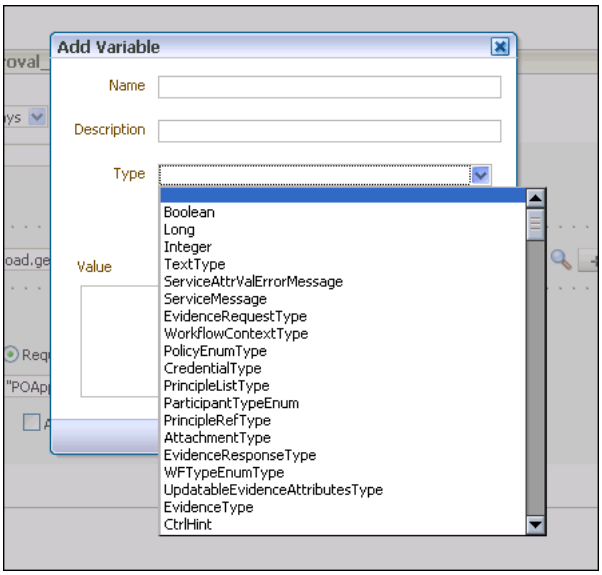
26.5.4.2.1 How to Add a Variable

Use the following procedure to add a variable.

1. Click **Add variable**.

   The Add Variable window displays, as shown in Figure 26–85.

Figure 26–85 Add Variable Window with "Type" Dropdown List



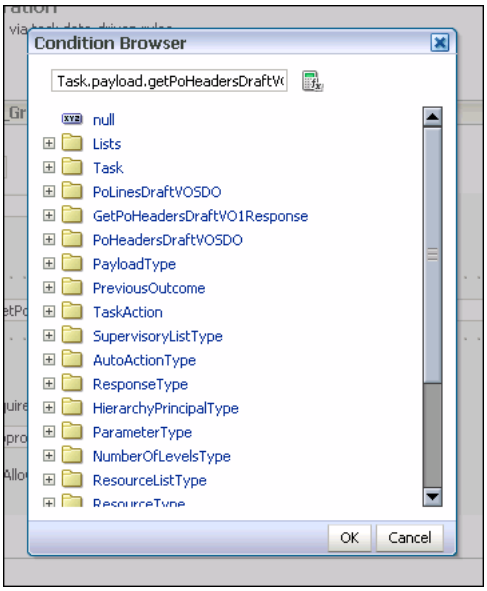
2. Enter a name for the variable.
3. Click the down arrow to select a variable type from the dropdown list.

   The types displayed in the list correspond to those that are available in the rule dictionary (built-in and others that have been registered).
4. Enter a value.
5. Click **OK**.

   The variable can now be used to define conditions.

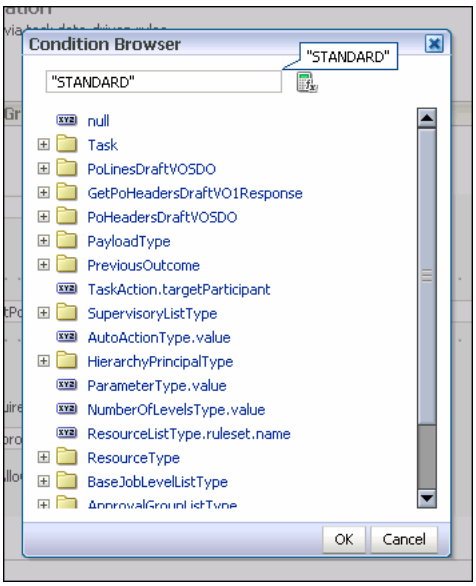
26.5.4.2.2 How to Define Conditions You can set the left and right sides of a condition by selecting operands from condition browsers. Clicking the magnifying glass icon displays the browsers. Figure 26–86 shows the left condition browser.

Figure 26–86 Left Condition Browser



The operator for comparing the operands of the condition changes based on the type of operand selected for the left side of the condition. Figure 26–87 shows the right condition browser.

Figure 26–87 Right Condition Browser



You also can define more complex conditions using the Expression Builder.

For more information, see the section "Creating ADF Data Binding EL Expressions" in *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*. Also, see the section "Creating EL Expressions" in *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework.*

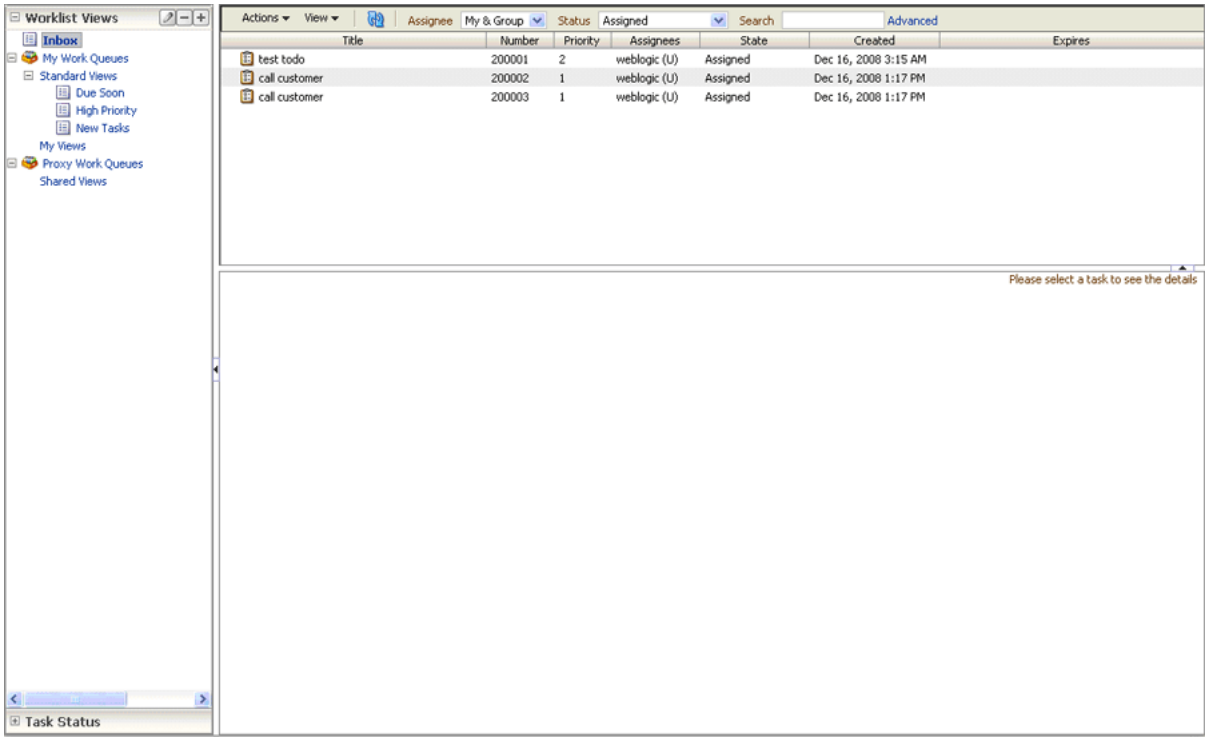
26.5.4.2.3 How to Define Assertions You can specify an assertion by selecting the appropriate rule function from the dropdown list.

26.5.5 Using the Task Listing Region

The task listing region in the BPM Worklist is available as standalone, reusable component that you can use to display a list of tasks in an embedding application. It is provided as an ADF library that you can include in the embedding application.

Figure 26–88 shows the task listing region.

Figure 26–88 Task Listing Region



26.5.5.1 How to Embed the Task Listing Region in an Application

The task listing region is exposed as a portlet and can be embedded in other applications.

The consumer application is developed using JDeveloper. The task list portlet then is embedded in a page in the consumer application. At run time, the login credentials passed to the consumer application are propagated to the WSRP producer and the portlet content is displayed on the page. The standalone task list portlet is deployed on the WLS PORTLET server, which would contact the remote WLS SOA server for workflow services. A separate portlet ear is provided for deployment on the portlet server.

Any consumer can use the task list portlet after registering to the portlet producer (WLS Portlet server).

To embed the task listing region in an application:

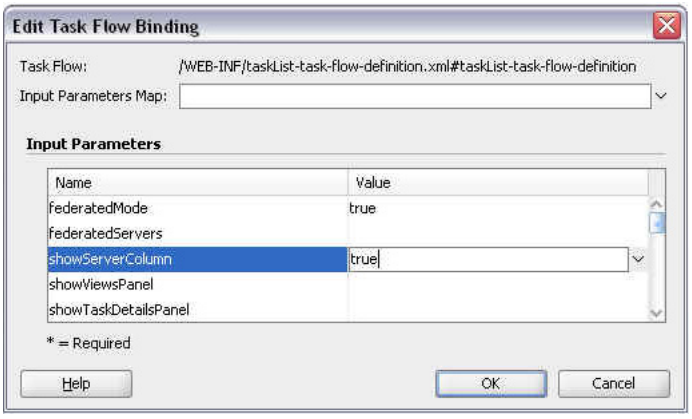
The task list ADF library `adflibTaskListTaskFlow.jar` file must be in the class path. This jar is available in JDeveloper in the BPM Worklist Components library.

1. In JDeveoloper, create a new application and provide a name; for example, "TaskListTaskFlowSample."
2. Add `adflibTaskListTaskFlow.jar` task flow in the project's class path.
3. Add the following libraries from **Libraries and Classpath** to the class path:
   - BPM Worklist Components
   - BPM Workflow
   - WSRP Container

4. If you run your application on a non-SOA server, perform the steps listed in Section 26.5.5.1.1, "How to Deploy a Human Task Detail Page on a Remote Non-SOA Server." Otherwise, continue to step 5.
5. Create a `.jspx` page. You can name it something like "testSample.jspx."
6. Choose **adflibTaskListFlow.jar** from the **Component Palette**, and drag **Tasklist-flow-definition** onto the `.jspx` page as a region.

   The Edit Task Flow Binding dialog displays.

Figure 26–89 Edit Task Flow Binding Dialog



Example 26–7 shows what is created in `testSamplePagedef.xml`.

Example 26–7 testSamplePagedef.xml Code

```
<taskFlow id="taskListtaskflowdefinition1"
   taskFlowId="/WEB-INF/taskList-task-flow-definition.xml
     #taskList-task-flow-definition"
   xmlns="http://xmlns.oracle.com/adf/controller/binding">
  <parameters>
    <parameter id="federatedMode" value="true"
       xmlns="http://xmlns.oracle.com/adfm/uimodel"/>
    <parameter id="showServerColumn" value="true"
       xmlns="http://xmlns.oracle.com/adfm/uimodel"/>
  </parameters>
</taskFlow>
```

7. Add the following shared libraries to `weblogic-application.xml`:

```
<library-ref>
    <library-name>oracle.soa.workflow</library-name>
</library-ref>
```

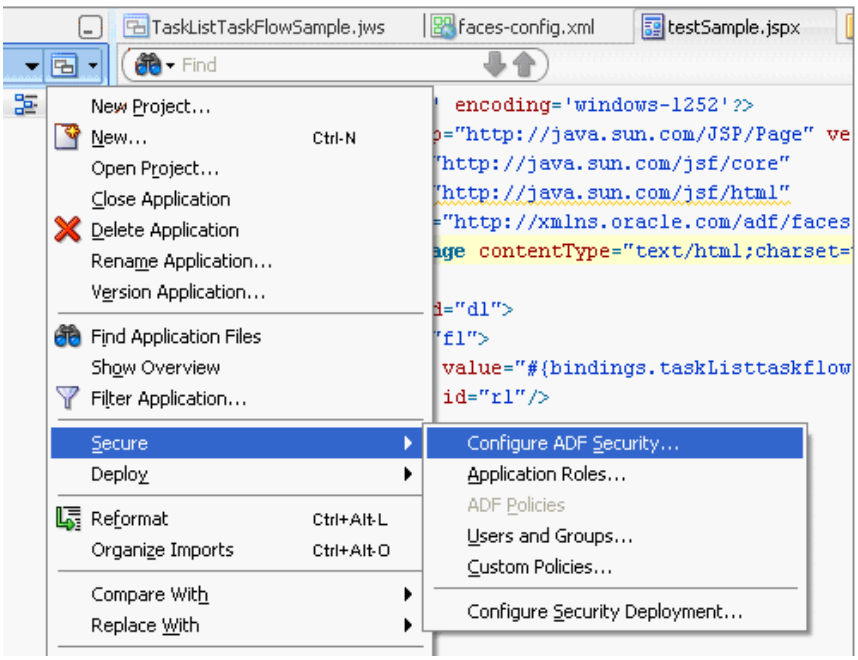
> **Note:** Add the libraries appropriately if you have oracle.soa.workflow.wc installed on your server.

8. From the Edit WAR Deployment Profile Properties dialog, select the following:
   - adflibTaskListTaskFlow.jar
   - adflibWorklistComponents.jar

- wsrp-container.jar.

9. From the Level menu, choose **Secure > Configure ADF Security** to secure the application, as shown in Figure 26–90.

Figure 26–90 Securing the Application



10. From the Configure ADF Security dialog, do the following:
    a. Select **ADF Authentication**.
    b. Select **HTTP Basic Authentication**.
    c. Click **Finish**.
11. If you are using the task flow in the federated mode, see Section 26.5.5.1.2, "How to Use the Task Flow in the Federated Mode," for additional steps you must perform. Otherwise, continue to step 12.
12. Create an EAR deployment profile, build an ear and deploy it.
13. Go to the following URL:
    `http://server:port/TaskListTaskFlowSample-ViewController-context-root/faces/testSample.jspx`.
14. From the popup dialog that displays, log in as any user. The task list for that user displays.

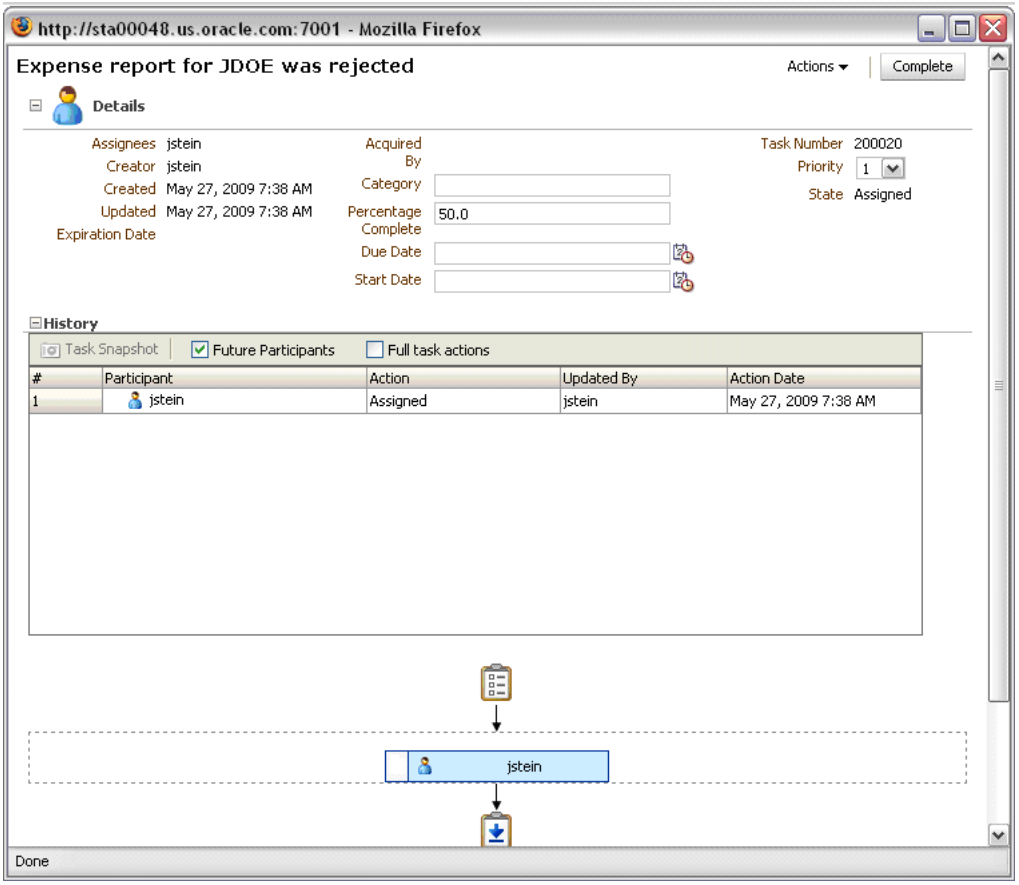
    The dropdown list contains all available servers. Selecting any combination of servers refreshes the task list to display all tasks that belong to those servers.

    If "showServerColumn" was passed as **true**, the server column, indicating the server to which the task belongs, displays in the task.

Figure 26–91 Server Column

15. Click any of the tasks titles to display a dialog containing the details of the task.

Figure 26–92 Task Details



26.5.5.1.1 How to Deploy a Human Task Detail Page on a Remote Non-SOA Server The following procedure is required if you run your application on a non-SOA server.

To deploy on a non-SOA server:

1. Deploy the oracle.soa.workflow shared library on the non-SOA WLS server. Do the following:

   a. Navigate to `http://remote_hostname:remote_port/`console, where `remote_hostname` and `remote_port` are the host name and port for the remote non-SOA WLS server.

   b. Click the "Deployments" link, then click **Install**.

   c. Ensure the following value in the **Path** field is specified. Be sure to replace `$ADE_VIEW_ROOT` with the actual directory. For example:

$ADE_VIEW_ROOT/fmwtools/fmwtools_home/jdeveloper/soa/modules/oracle.soa.workflow_11.1.1.

d. Select **oracle.soa.workflow.jar**.

e. Click **Finish**.

f. Confirm that the oracle.soa.workflow(11.1.1,11.1.1) library has an Active State.

2. Define the foreign JNDI on the non-SOA WLS server. Do the following:

a. Navigate to `http://remote_hostname:remote_port/console`, where `remote_hostname` and `remote_port` are the host name and port for the remote non-SOA WLS server.

b. Navigate to **Domain Structure > Services > Foreign JNDI Providers**.

c. Click **New**.

d. Enter the name `ForeignJNDIProvider-SOA`.

e. Click **OK**.

f. Click the "ForeignJNDIProvider-SOA" link.

g. Replace `soa_hostname` and `soa_port` with the host name and port for the SOA WLS server.

h. Enter the following values:

- Initial Context Factory: `weblogic.jndi.WLInitialContextFactory`
- Provider URL: `t3://soa_hostname:soa_port/soa-infra`
- User: `weblogic`
- Password: `weblogic`

Note: The Provider URL is referring to the soa-infra application, not the domain. Do not change soa-infra to soa.

---

i. Click **Save**.

3. Define the JNDI links on non-SOA WLS server. Do the following:

a. Navigate to `http://remote_hostname:remote_port/console`, where `remote_hostname` and `remote_port` are the host name and port for the remote non-SOA WLS server.

b. Navigate to **Domain Structure > Services > Foreign JNDI Providers**.

c. Click `ForeignJNDIProvider-SOA`.

d. Select the **Link** tab, then click **New**.

e. Enter the following values:

- Name: `RuntimeConfigService`
- Local JNDI Name: `RuntimeConfigService`
- Remote JNDI Name: `RuntimeConfigService`

f. Click **OK**.

g. Repeat steps a through f to enter the following values for Name, Local JNDI Name, and Remote JNDI name:

- `ejb/bpel/services/workflow/TaskServiceBean`
- `ejb/bpel/services/workflow/TaskMetadataServiceBean`

```
- TaskReportServiceBean
- TaskEvidenceServiceBean
- TaskQueryService
- UserMetadataService
```

Note: Specify "ejb/bpel/services/workflow/" for ejb/bpel/services/workflow/TaskServiceBean and ejb/bpel/services/workflow/TaskMetadataServiceBean only.

4. Change `system-jazn-data.xml` on the remote non-SOA WLS server to include the grant for `bpm-services.jar`, as shown in Example 26–8. If you are deploying the ADF Task Flow for Human Task to the integrated WLS server, then you can locate the `system-jazn-data.xml` file in `$ADE_VIEW_ROOT/system11.1.1.1.32.53.26/DefaultDomain/config/fmwconfig`.

   Important: Be sure to replace $ADE_VIEW_ROOT with the actual path.. For example, `/scratch/skaneshi/view_storage/skaneshi_d7b4/fmwtools/fmwtools_home/jdeveloper/soa/modules/oracle.soa.workflow_11.1.1/bpm-services.jar`.

Example 26–8 bpm-services.jar Grant Code

```
<grant>
     <grantee>
             <codesource>
                <url>file:$ADE_VIEW_ROOT/fmwtools/fmwtools_
                  home/jdeveloper/soa/modules/oracle.soa.workflow_11.1.1/bpm-services.jar</url>
             </codesource>
     </grantee>
         <permissions>
             <permission>
                 <class>oracle.security.jps.JpsPermission</class>
                 <name>VerificationService.createInternalWorkflowContext</name>
             </permission>
             <permission>
                 <class>oracle.security.jps.service.credstore.CredentialAccessPermission</class>
                 <name>credstoressp.credstore.BPM-CRYPTO.BPM-CRYPTO</name>
                 <actions>read,write</actions>
             </permission>
             <permission>
                 <class>oracle.security.jps.JpsPermission</class>
                 <name>IdentityAssertion</name>
                 <actions>*</actions>
             </permission>
         </permissions>
     </grant>
```

5. Restart the remote non-SOA WLS server.
6. Deploy your application containing the human task detail UI to the remote non-SOA WLS server.
7. Return to step 5 in Section 26.5.5.1, "How to Embed the Task Listing Region in an Application."

26.5.5.1.2 How to Use the Task Flow in the Federated Mode If you are using the task flow in the federated mode, you must put `wf_client_config.xml` in the class path of the

application. For more information, see Section 26.5.5.2, "How to Use Task Listing Region Parameters."

You also must enable global trust between the two servers. This is done so that the authenticated user is passed to all the federated servers defined in the client configuration file.

Important: You must restart all the servers.

To use the task flow in the federated mode:

Perform the following procedure for all the servers defined in `wf_client_config.xml`:

1. Log in to the WebLogic console.
2. Under **Domain Structures**, click the **soainfra** domain name.
3. Select the **Security** tab.
4. Click the **Advanced** link located near the **Save** button.
5. Enter a credential password of your choosing.

Note: You must use this password for all SOA servers.

6. Click **Save**.
7. Restart the server.
8. Return to step 12 in Section 26.5.5.1, "How to Embed the Task Listing Region in an Application."

Example 26–9 shows sample `wf_client_config.xml` code.

Note: Put `excludeFromFederatedList="true"` in the `<server>` element if you do not want to include the server in the federated servers list.

Example 26–9 Sample wf_client_config.xml Code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<workflowServicesClientConfiguration
xmlns="http://xmlns.oracle.com/bpel/services/client">
   <server name="default" default="true" excludeFromFederatedList="true">
      <localClient>
         <participateInClientTransaction>false</participateInClientTransaction>
      </localClient>
      <remoteClient>
         <serverURL>t3://sta00048.us.oracle.com:7001</serverURL>
         <initialContextFactory>weblogic.jndi.WLInitialContextFactory
         </initialContextFactory>
         <participateInClientTransaction>false</participateInClientTransaction>
      </remoteClient>
      <soapClient>
         <rootEndPointURL>http://sta00048.us.oracle.com:7001</rootEndPointURL>
         <identityPropagation mode="dynamic" type="saml">
            <policy-references>
               <policy-reference enabled="true" category="security"
```

```
                                uri="oracle/wss10_saml_token_client_policy"/>
            </policy-references>
         </identityPropagation>
      </soapClient>
   </server>
   <server name="ERP">
      <soapClient>
         <rootEndPointURL>http://sta00147.us.oracle.com:7001</rootEndPointURL>
         <identityPropagation mode="dynamic" type="saml">
            <policy-references>
               <policy-reference enabled="true" category="security"
                                uri="oracle/wss10_saml_token_client_policy"/>
            </policy-references>
         </identityPropagation>
      </soapClient>
      <remoteClient>
         <serverURL>t3://sta00147.us.oracle.com:7001</serverURL>
         <initialContextFactory>weblogic.jndi.WLInitialContextFactory
         </initialContextFactory>
         <participateInClientTransaction>false</participateInClientTransaction>
      </remoteClient>
   </server>

   <server name="CRM">
      <soapClient>
         <rootEndPointURL>http://sta00048.us.oracle.com:7001</rootEndPointURL>
         <identityPropagation mode="dynamic" type="saml">
            <policy-references>
               <policy-reference enabled="true" category="security"
                                uri="oracle/wss10_saml_token_client_policy"/>
            </policy-references>
         </identityPropagation>
      </soapClient>
      <remoteClient>
         <serverURL>t3://sta00048.us.oracle.com:7001</serverURL>
         <initialContextFactory>weblogic.jndi.WLInitialContextFactory
         </initialContextFactory>
         <participateInClientTransaction>false</participateInClientTransaction>
      </remoteClient>
   </server>
</workflowServicesClientConfiguration>
```

26.5.5.2 How to Use Task Listing Region Parameters

Task listing region parameters control the display behavior of the embedded region. This section describes these parameters.

Display Parameters

- *federatedMode*

  The task list displays in federated mode if this paramater is passed as **true**. To run the task flow in federated mode, you must pass the list of federated servers to the task flow using one of these options:

  – Put the client configuration file `wf_client_config.xml` in the class path (`APP-INF\classes\wf_client_config.xml` at the ear level or the WEB-INF\classes of the web application). The client configuration file contains all federated server details. For more information, see Section 26.5.5.1.2, "How to Use the Task Flow in the Federated Mode."

  - Construct a JAXB object, which contains the federated servers list. This JAXB object can be passed to the task flow through the *federatedServers* parameter. For information on how to construct the JAXB object, see the details of that parameter.

- *federatedServers*

  This parameter displays a list of servers if the task flow is run in federated mode. Construct a JAXB object (WorkflowServicesClientConfigurationType) using the code shown below, and then pass it as a parameter to the task flow. Ensure that you set one server as the "default," as indicated (in **bold** type) in the code.

  A default server is used when you have many servers defined in `wf_client_config.xml` or in the JAXB object. However, the workflow client is preferred for a single server. There are a few legacy APIs that do not take the server name as parameter. To support these APIs, you must define one single server as default server; otherwise these APIs do not work.

Example 26–10 Defining a Single Server

```
import oracle.bpel.services.workflow.client.config.IdentityPropagationType;
import oracle.bpel.services.workflow.client.config.PolicyReferenceType;
import oracle.bpel.services.workflow.client.config.PolicyReferencesType;
import oracle.bpel.services.workflow.client.config.RemoteClientType;
import oracle.bpel.services.workflow.client.config.ServerType;
import oracle.bpel.services.workflow.client.config.SoapClientType;
import oracle.bpel.services.workflow.client.config.
WorkflowServicesClientConfigurationType;

WorkflowServicesClientConfigurationType wscct = new
WorkflowServicesClientConfigurationType();

List<ServerType> servers = wscct.getServer();

/**** Setting default server in the list ****/

ServerType defaultServer= new ServerType();
servers.add(defaultServer);

defaultServer.setDefault(true);
defaultServer.setExcludeFromFederatedList(true);

defaultServer.setName("default");

RemoteClientType rct = new RemoteClientType();
rct.setServerURL("t3://sta00048.us.oracle.com:7001");
rct.setInitialContextFactory("weblogic.jndi.WLInitialContextFactory");
rct.setParticipateInClientTransaction(false);
server.setRemoteClient(rct);

SoapClientType sct = new SoapClientType();
PolicyReferencesType prts = new PolicyReferencesType();

PolicyReferenceType prt = new PolicyReferenceType();
prt.setEnabled(true);
prt.setCategory("security");
prt.setUri("oracle/wss10_saml_token_client_policy");
prts.getPolicyReference().add(prt);

IdentityPropagationType ipt = new IdentityPropagationType();
ipt.setMode("dynamic");
```

```
ipt.setType("saml");
ipt.setPolicyReferences(prts);

sct.setRootEndPointURL("http://sta00048.us.oracle.com:7001");
sct.setIdentityPropagation(ipt);

defaultServer.setSoapClient(sct);

/****** Setting Federated Server 1 to the list ****/

ServerType server1 = new ServerType();
servers.add(server1);
server1.setName("Human Resource");

RemoteClientType rct1 = new RemoteClientType();
rct1.setServerURL("t3://stadl28.us.oracle.com:7001");
rct1.setInitialContextFactory("weblogic.jndi.WLInitialContextFactory");
rct1.setParticipateInClientTransaction(false);
server1.setRemoteClient(rct1);

SoapClientType sct1 = new SoapClientType();
PolicyReferencesType prts1 = new PolicyReferencesType();

PolicyReferenceType prt1 = new PolicyReferenceType();
prt1.setEnabled(true);
prt1.setCategory("security");
prt1.setUri("oracle/wss10_saml_token_client_policy");
prts1.getPolicyReference().add(prt1);

IdentityPropagationType ipt1 = new IdentityPropagationType();
ipt1.setMode("dynamic");
ipt1.setType("saml");
ipt1.setPolicyReferences(prts1);

sct1.setRootEndPointURL("http://stadl28.us.oracle.com:7001");
sct1.setIdentityPropagation(ipt1);

server1.setSoapClient(sct1);

/****** Setting Federated Server 2 to the list ****/

ServerType server2 = new ServerType();
servers.add(server2);
server2.setName("Financials");

RemoteClientType rct2 = new RemoteClientType();
rct2.setServerURL("t3://sta00048.us.oracle.com:7001");
rct2.setInitialContextFactory("weblogic.jndi.WLInitialContextFactory");
rct2.setParticipateInClientTransaction(false);
server2.setRemoteClient(rct2);

SoapClientType sct2 = new SoapClientType();
PolicyReferencesType prts2 = new PolicyReferencesType();

PolicyReferenceType prt2 = new PolicyReferenceType();
prt2.setEnabled(true);
prt2.setCategory("security");
prt2.setUri("oracle/wss10_saml_token_client_policy");
prts2.getPolicyReference().add(prt2);
```

```
IdentityPropagationType ipt2 = new IdentityPropagationType();
ipt2.setMode("dynamic");
ipt2.setType("saml");
ipt2.setPolicyReferences(prts2);

sct2.setRootEndPointURL("http://sta00048.us.oracle.com:7001");
sct2.setIdentityPropagation(ipt2);

server2.setSoapClient(sct2);
```

- *showServerColumn*

  If the task flow is run in federated mode, the server column in the task list, by default, is not displayed. To display the column, this parameter must be passed as **true**.

- *wfCtxID*

  This is a workflow context token string and is used to create workflow context inside the task flow. If the application Single Sign On is enabled or is secured using ADF security, this parameter is not required. The workflow context is shown below.

Example 26–11 Workflow Context

```
IWorkflowContext wfCtx =  wfSvcClient.getTaskQueryService().authenticate(username,
  password,
  realm,
  null;
wfCtxID = wfCtx.getToken();
```

- *showViewsPanel*

  The views panel displays only if passed as **true**. By default, it is not displayed.

- *showTaskDetailsPanel*

  The task details panel displays only if passed as **true**. By default, it is not displayed.

- *refreshURL*

  This string enables task details in the task listing region to display in an inline frame. If action is taken on the task details page, the action refreshes the task listing area with the p age URL in which the task flow/portlet is contained.

  Since the task flow does not know the URL of the container page, the URL must be passed as a parameter. Get the parameter by calling the `getRequestURL()` method on the request object. You can pass the full URL either by calling the `getRequestURL()` method on the request object, or by passing the URL using the following format:

  ```
  /context-root/faces/page-name
  For example: /FederatedWorklist/faces/test.jspx
  ```

- *localeSource*

  A string parameter that passes the locale source. It can be from the browser (BROWSER) or from the identity context (IDENTITY).

- *showActionDropdown*

  The action dropdown does not display only if passed as **false**. By default, it is displayed.

- *showViewFilter*

  The view filter dropdown does not display only if passed as **false**. By default, it is displayed.

- *showCreateTODOTaskAction*

  Specifies if the Actions menu displays the TODO action. To hide the TODO action, set this parameter to false. By default the Actions menu inclueds the TODO action.

- *showAssignmentFilter*

  The assignment filter dropdown does not display only if passed as **false**. By default, it is displayed.

- *showStatusFilter*

  The status filter dropdown does not display only if passed as **false**. By default, it is displayed.

- *showSearchControl*

  The search box does not display only if passed as **false**. By default, it is displayed.

- *displayColumnsList*

  This comma-delineated list of strings enables the list of columns to be displayed in the region in the order specified.

- *sortColumn*

  This string specifies the name of the column to be used for sorting tasks by default in the region.

- *sortOrder*

  This string specifies the sort order for sorting tasks, that is ascending (ASC) or descending (DESC).

Filter Parameters

- *assignmentFilter*

  This string specifies the value to be used as the selection of the assignment filter for tasks. If not set, the values default to My and Group. For more information, see Section 26.5.5.2.1, "Using Assigment Filter Constraints."

- *viewFilter*

  This string specifies the value to be used as the selection of the view name to filter the tasks. If not set, the value defaults to Inbox.

- *taskTypesFilterList*

  A strings parameter that specifies the comma-delineated list of the task type(s) to be used for filtering the tasks that are displayed.

- *attributesFilterOperator*

  A string that specifies the operator (and/or) to be used as the predicate join criterion for the field-specified in Attribute Filter list.

- *attributesFilterList*

  A string parameter that specifies the comma-delineated list of name value pairs to be used to filter tasks based on attribute values. (The name is task column name and value is column value.)

Example

If the user wants to see the task with attribute filter values as priority = 1 and status = ASSIGNED and promoted flexfield textAttribute1 = NorthAmerica, then the user would set the values as the following:

```
attributeFilterList: priority=1, status=ASSIGNED, textAttribute1=NorthAmerica
```

The attribute filter operator would be the following:

```
attributeFilterOperator: and
```

26.5.5.2.1 Using Assigment Filter Constraints The following is a list of assignment filter constraints:

- My
- Group
- My+Group
- Reportees
- Creator
- Owner
- Reviewer
- Previous
- Admin

26.5.5.2.2 Passing Column Constraints The list below contains the column constants that can be passed in the*displayColumnList* parameter. The constant value is the value that should be passed. For example, for `TITLE_COLUMN = "title"`, the "title" should be passed, not "TITLE_COLUMN".

Primary Column Contraints

STARTDATE_COLUMN = "startDate";
TASKDEFINITIONNAME_COLUMN = "taskDefinitionName";
OWNERROLE_COLUMN = "ownerRole";
UPDATEDDATE_COLUMN = "updatedDate";
COMPOSITEVERSION_COLUMN = "compositeVersion";
CREATOR_COLUMN = "creator";
FROMUSER_COLUMN = "fromUser";
PERCENTAGECOMPLETE_COLUMN = "percentageComplete";
OWNERGROUP_COLUMN = "ownerGroup";
ENDDATE_COLUMN = "endDate";
COMPOSITENAME_COLUMN = "compositeName";
DUEDATE_COLUMN = "dueDate";
COMPOSITEDN_COLUMN = "compositeDN";
TASKDISPLAYURL_COLUMN = "taskDisplayUrl";
UPDATEDBY_COLUMN = "updatedBy";
OUTCOME_COLUMN = "outcome";
TASKNAMESPACE_COLUMN = "taskNamespace";
APPROVERS_COLUMN = "approvers";
APPLICATIONCONTEXT_COLUMN = "applicationContext";
OWNERUSER_COLUMN = "ownerUser";
CATEGORY_COLUMN = "category";
ACQUIREDBY_COLUMN = "acquiredBy";

```
ORIGINALASSIGNEEUSER_COLUMN = "originalAssigneeUser";
```

Other Column Constraints

```
ACQUIREDBY_COLUMN = "acquiredBy";
ASSIGNEDDATE_COLUMN = "assignedDate";
APPROVERS_COLUMN = "approvers";
ASSIGNEES_COLUMN = "assignees";
ASSIGNEESDISPLAYNAME_COLUMN = "assigneesDisplayName";
ASSIGNEEGROUPS_COLUMN = "assigneeGroups";
ASSIGNEEGROUPSDISPLAYNAME_COLUMN = "assigneeGroupsDisplayName";
ASSIGNEEUSERS_COLUMN = "assigneeUsers";
ASSIGNEEUSERSDISPLAYNAME_COLUMN = "assigneeUsersDisplayName";
OUTCOME_COLUMN = "outcome";
CREATEDDATE_COLUMN = "createdDate";
ELAPSEDTIME_COLUMN = "elapsedTime";
DIGITALSIGNATUREREQUIRED_COLUMN = "digitalSignatureRequired";
PASSWORDREQUIREDONUPDATE_COLUMN = "passwordRequiredOnUpdate";
ENDDATE_COLUMN = "endDate";
EXPIRATIONDATE_COLUMN = "expirationDate";
EXPIRATIONDURATION_COLUMN = "expirationDuration";
FROMUSER_COLUMN = "fromUser";
FROMUSERDSIPLAYNAME_COLUMN = "fromUserDisplayName";
HASSUBTASK_COLUMN = "hasSubtask";
STATE_COLUMN = "State";
TASKID_COLUMN = "taskId";
VERSION_COLUMN = "version";
TASKNUMBER_COLUMN = "taskNumber";
UPDATEDBY_COLUMN = "updatedBy";
UPDATEDBYDISPLAYNAME_COLUMN = "updatedByDisplayName";
UPDATEDDATE_COLUMN = "updatedDate";
VERSIONREASON_COLUMN = "versionReason";
CREATOR_COLUMN = "creator";
OWNERUSER_COLUMN = "ownerUser";
OWNERGROUP_COLUMN = "ownerGroup";
OWNERROLE_COLUMN = "ownerRole";
PRIORITY_COLUMN = "priority";
DOMAINID_COLUMN = "domainId";
INSTANCEID_COLUMN = "instanceId";
PROCESSID_COLUMN = "processId";
PROCESSNAME_COLUMN = "processName";
PROCESSTYPE_COLUMN = "processType";
PROCESSVERSION_COLUMN = "processVersion";
TITLE_COLUMN = "title";
TITLERESOURCEKEY_COLUMN = "titleResourceKey";
IDENTIFICATIONKEY_COLUMN = "identificationKey";
TASKDEFINITIONID_COLUMN = "taskDefinitionId";
TASKDEFINITIONNAME_COLUMN = "taskDefinitionName";
APPLICATIONNAME_COLUMN  = "applicationName";
ASSIGNEETYPE_COLUMN = "assigneeType";
CATEGORY_COLUMN = "category";
COMPONENTNAME_COLUMN  = "componentName";
COMPOSITEDN_COLUMN  = "compositeDN";
COMPOSITEINSTANCEID_COLUMN  = "compositeInstanceId";
COMPOSITENAME_COLUMN  = "compositeName";
COMPOSITEVERSION_COLUMN  = "compositeVersion";
```

```
CONVERSATIONID_COLUMN  = "conversationId";
DUEDATE_COLUMN  = "dueDate";
PARTICIPANTNAME_COLUMN  = "participantName";
PERCENTAGECOMPLETE_COLUMN  = "percentageComplete";
READBYUSERS_COLUMN  = "readByUsers";
STARTDATE_COLUMN  = "startDate";
PARENTTASKVERSION_COLUMN  = "parentTaskVersion";
TASKGROUPINSTANCEID_COLUMN  = "taskGroupInstanceId";
SUBTASKGROUPINSTANCEID_COLUMN  = "subTaskGroupInstanceId";
ROOTTASKID_COLUMN = "rootTaskId";
PARENTTASKID_COLUMN = "parentTaskId";
CORRELATIONID_COLUMN = "correlationId";
TASKDISPLAYURL_COLUMN = "taskDisplayUrl";
STAGE_COLUMN = "stage";
LOCALE_COLUMN = "locale";
```

26.5.6 How to Use the Task History Region to Preview Approvers

The History component is an ADF declarative UI component used to render the past and future participants for an initiated task, and for a task before initiation. The component can edit the future participants, and can be embedded in any ADF page. Parameters are the following:

- Task Object
- Workflow Service Client
- Workflow context
- showTabularView
- showGraphicalView

If any of the first three parameters are passed as null, or if the correlation ID of a task object is null, the component does not show the approval list. Instead, it shows a message explaining the possible cause why it failed to show the approval list. The last two parameters are used to indicate whether tabular or graphical views are to be shown. By default, both tabular and graphical views are shown. The value true/false can be passed to last two parameters.

26.6 Using the User Metadata Migration Utility

The user metadata migration utility, **hwfMigrator**, is a tool that automates the process of migrating Workflow user-configurable data from one SOA server to another by executing a shell script.

For more information about the user metadata migration utility, see Moving Human Workflow Data from a Test to a Production Environment in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.