Oracle® Fusion Middleware

Developing Business Processes with Oracle Business Process Management Studio

12*c* (12.1.3)

E37119-06

February 2016

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

ORACLE

Oracle Fusion Middleware Developing Business Processes with Oracle Business Process Management Studio, 12c (12.1.3)

E37119-06

Copyright © 2010, 2016, Oracle and/or its affiliates. All rights reserved.

Primary Author: Oracle Corporation

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface xxv

Audience xxv
Related Documents xxv
Conventions xxvi

What's New in This Guide xxvii

New and Changed Features for 12c Release 1 (12.1.3) xxvii
Other Significant Changes in this Guide for 12c Release 1 (12.1.3) xxvii

Part I Using Oracle BPM Studio

1 Introduction to Oracle BPM Studio

Working with Oracle BPM Suite 1-1
Overview of the Application Development Life Cycle 1-2
Introduction to the Oracle BPM Studio User Interface 1-2
Applications Window 1-3
BPMN Process Editor 1-4
Components Window 1-5
Process Asset Manager Navigator 1-5
Structure View 1-6
Thumbnail View 1-7
Simulation View 1-8
Log Window 1-8
Documentation Window 1-8

2 Working with BPM Projects

Introduction to BPM Projects 2-1
Introduction to Project Resources 2-1
Sharing Projects Between Oracle BPM Users 2-2
Creating and Working with Projects 2-2
How to Create a New Project 2-2

How to Open a Project from the File System ... 2-3
How to Export a Project ... 2-3
How to Import a Previously Exported Project ... 2-4
How to Edit Project Preferences ... 2-4

3 Working with Processes and the Process Editor

Getting Started with Processes ... 3-1
Introduction to Business Processes ... 3-1
How to Create a New Business Process ... 3-2
How to Open a Business Process ... 3-3
How to Delete a Business Process ... 3-3
Introduction to the Process Editor ... 3-4
Working with Processes ... 3-5
How to Export a Process As an Image ... 3-5
How to Change the Highlight Level for Messages in a Process ... 3-6
How to Change the Zoom Level in a Process ... 3-7
How to Configure Layout Properties and Use a Grid in a Process ... 3-7
Working with Flow Objects in Your Process ... 3-8
How to Add Flow Objects from the Component Window ... 3-8
How to Add Flow Objects from the Process Editor Toolbar ... 3-8
How to Add Flow Objects from a Context Menu ... 3-9
How to Edit Flow Object Properties ... 3-9
How to Display and Fix Errors or Warnings in Flow Objects ... 3-10
How to Mark and Unmark a Flow Object as Draft ... 3-11
How to Copy and Paste Flow Objects ... 3-12
How to Add and Use Sequence Flows ... 3-13
Working with Draft Processes ... 3-14
Introduction to Draft Processes ... 3-14
How to Mark a Flow Object as Draft ... 3-15
Documenting Your Process ... 3-15
Introduction to the Documentation Editor ... 3-15
How to Add Documentation to Your Process ... 3-15
Generating Process Reports for Your Project ... 3-16

Part II Modeling a Process

4 Modeling Your Organization

Introduction to Organizations ... 4-1
Introduction to the Organization Editor ... 4-2
Working with Organizations ... 4-2
How to Create an Organizational Unit ... 4-2
How to Create a Calendar ... 4-3
How to Create Holidays ... 4-3

Introduction to Roles 4-4
Working with Roles 4-4
How to Create a New Role 4-4
How to Add Members to a Role 4-4
Introduction to Organizational Charts 4-5
Introduction to Organizational Units 4-5
Introduction to Calendars 4-5
Introduction to Holidays 4-6
Introduction to Business Parameters 4-6
Working with Business Parameters 4-6
How to Add a Business Parameter 4-6
How to Assign a Value to a Business Parameter 4-7

5 Handling Information in Your Process Design

Introduction to Handling Information in Your Process Design 5-1
Basic Data Objects versus Complex Data Objects 5-2
Introduction to Data Objects 5-3
Supported Data Types for Data Objects 5-4
Default Values 5-4
Working with Process Data Objects 5-5
How to Add a Process Data Object 5-5
How to Edit a Process Data Object 5-6
How to Delete a Data Object 5-6
How to Assign a Value to a Process Data Object 5-6
Introduction to Activity Instance Attributes 5-7
Working with Activity Instance Attributes 5-8
Introduction to Subprocess Data Objects 5-8
Working with Subprocess Data Objects 5-9
Adding a Data Object to a Subprocess 5-9
Editing a Data Object in a Subprocess 5-10
Deleting a Data Object from a Subprocess 5-10
Introduction to Project Data Objects 5-11
Business Indicators 5-11
Supported Data Types for Project Data Objects 5-11
Working with Project Data Objects 5-11
How to Add a Project Data Object 5-12
How to Edit a Project Data Object 5-12
How to Delete a Project Data Object 5-13
How to Assign a Value to a Project Data Object 5-13
Introduction to Arguments 5-14
Naming Conventions 5-14
Scope and Access 5-15
Introduction to Data Associations 5-16

Introduction to the Data Association Editor 5-17
Introduction to Transformations 5-18
Defining Transformations 5-18
How to Define a Transformation 5-18
What Happens When You Define a Transformation 5-19

Part III Analyzing Process Performance

6 Running Simulations in Oracle BPM

Introduction to Running Simulations in Oracle BPM 6-1
Simulation Models and Simulation Definitions 6-1
Creating Simulation Models 6-2
How to Create a Simulation Model from a Business Process 6-2
How to Create and Configure a Simulation Model 6-3
Configuring Boundary Events 6-6
Creating Simulation Definitions 6-7
How to Create a Simulation Definition 6-8
Running Simulations 6-10
How to Run a Simulation 6-10
What Happens When You Run a Simulation 6-10
Understanding the Simulation View 6-11
Analyzing the Results of a Simulation 6-12
How to Analyze the Results of a Simulation Using a Chart 6-12
How to Generate a Simulation Report 6-13
What Happens when You Generate a Simulation Report 6-14

7 Using Process Analytics

Introduction to Process Analytics 7-1
Process and Activity Performance Metrics 7-3
Workload Metrics 7-3
Human Resource Metrics 7-3
Typical Process Analytics Workflow 7-3
How to Enable Global Flags for Publishing Analytics 7-4
Configuring Projects, Processes, and Activities to Generate Sampling Points 7-5
User-Defined Measurements 7-6
Enable HWF and Case Measurements 7-6
How to Configure the Sampling Point Generation of a Project 7-6
What Happens When You Configure a Project To Generate Sampling Points 7-7
How to Configure the Sampling Point Generation for an Activity 7-7
What Happens When You Configure the Sampling Points for an Activity 7-7
Adding Business Indicators to Projects 7-8
How to Add a Business Indicator to a Project 7-9
What Happens When You Add a Business Indicator to a Process 7-13

Adding Measurement Marks to Processes ... 7-13
How to Add Single Measurement Marks to a Process ... 7-15
What Happens When You Add a Single Measurement to a Process ... 7-16
How to Measure a Business Indicator in a Process Section Using Measurement Marks ... 7-16
What Happens When You Use Measurement Marks to Measure Business Indicator Values for a Section of a Process ... 7-18
Adding Counters to the Activities in a Process ... 7-18
How to Add a Counter Mark to an Activity in a Process ... 7-18
What Happens When You Add a Counter Mark to an Activity in a Process ... 7-19
How to Delete a Counter Mark ... 7-19
What Happens When You Delete a Counter Mark ... 7-19
Defining Analytics View Identifier ... 7-19
How to Define the Analytics View Identifier ... 7-20
Configuring BAM 12c Process Metrics Generation in a Project ... 7-20
BAM 12c Process Metrics ... 7-20
How to Configure BAM 12c Process Metrics Generation in a Project ... 7-21
What Happens When You Enable BAM 12c Process Metrics in a Project ... 7-21
Enabling Oracle BAM 11g in a Project ... 7-22
How to Enable Oracle BAM 11g in a Project ... 7-22
What Happens When You Enable Oracle BAM 11g ... 7-22

Part IV Working with Business Components

8 Using the Business Catalog

Introduction to the Business Catalog ... 8-1
Non-Synthesized Components ... 8-3
Synthesized Components ... 8-3
Adding Components to the Business Catalog ... 8-4
Using Modules to Organize Business Components ... 8-5
Adding a New Module ... 8-6
How to Add a New Module ... 8-6
What Happens When You Add a New Module ... 8-6
Deleting a Module ... 8-7
How to Delete a Module ... 8-7
What Happens When You Delete a Module ... 8-7
Customizing Synthesized Types ... 8-7
How to Customize a Synthesized Type ... 8-7
What Happens When You Customize a Synthesized Type ... 8-8
Creating an Enumeration ... 8-8
How to Create an Enumeration ... 8-8
How to Add Attributes to an Enumeration ... 8-9
Using an Enumeration in a Simple Expression ... 8-9

9 Sharing BPM Projects Using the Process Asset Manager

Introduction to the Process Asset Manager 9-1
Working with BPM Projects Stored in the Process Asset Manager 9-2
How to Set Up an Environment to Work with Projects Stored in the Process Asset Manager 9-2
How to Modify a BPM Project Stored in the Process Asset Manager 9-3
How to add a BPM Project to the Process Asset Manager 9-3
How to Export a BPM Project Stored in the Business Process Manager 9-3
Working with the Process Asset Manager 9-4
How to Create a Process Asset Manager Connection 9-4
How to Check Out a BPM Project from the Process Asset Manager 9-5
How to Save a BPM Project to the Process Asset Manager 9-5
How to Update Local BPM Projects 9-6
How to Delete a BPM Project from the Process Asset Manager 9-7
How to View the Change History 9-7

10 Modeling Business Objects

Introduction to Business Objects 10-1
Types of Business Objects 10-3
Benefits of Modeling Using Business Objects 10-3
Naming Conventions for Business Objects 10-4
Working with Business Objects 10-4
How to Add a Business Object 10-4
What Happens When You Add a Business Object 10-5
How to Modify a Business Object 10-5
How to Delete a Business Object 10-5
What Happens When You Delete a Business Object 10-6
How to Document a Business Object 10-6
What Happens When You Document a Business Object 10-6
Using a Business Object in a Process 10-6
How to Use a Business Object in a Process 10-6
What Happens When You Use a Business Object in a Process 10-7
Adding Business Objects Based on a XML Schema Element or Type 10-7
How to Add a Business Object Based on a XML Schema Element or Type 10-7
What Happens When You Create a Business Object Based on an XML Schema Element or Type 10-8
How to add an XML Schema to Your BPM Project 10-8
What Happens When You Add a Schema File to Your Project 10-8
Introduction to Business Object Attributes 10-8
Supported Data Types for Business Object Attributes 10-9
Naming Conventions for Business Object Attributes 10-10
Working with Business Object Attributes 10-10

How to Add a Business Object Attribute ... 10-10
How to Remove a Business Object Attribute ... 10-11
How to Document a Business Object Attribute ... 10-11
What Happens When You Document a Business Object Attribute ... 10-12
Working with Business Object Methods ... 10-12
How to Add a Business Object Method ... 10-12
How to Change the Signature of Business Object Method ... 10-12
How to Remove a Business Object Method ... 10-13
How to Document a Business Object Method ... 10-13
Sharing Business Objects ... 10-13
How to Export a Business Object ... 10-13
How to Import Business Objects from a File ... 10-14
Introduction to Business Object Inheritance ... 10-14
Method Overloading ... 10-14
Polymorphism ... 10-15
Method Overriding ... 10-15
Attribute Shadowing ... 10-15
Abstract Business Objects ... 10-15
Working with Business Object Inheritance ... 10-15
How to Create a Child Business Object ... 10-15
What Happens When You Create a Child Object ... 10-16
How to Mark a Business Object as Abstract ... 10-16
What Happens When You Mark a Business Object as Abstract ... 10-16

11 Working with Human Tasks

Introduction to Human Tasks in BPM ... 11-1
Using Human Task Patterns in Oracle BPM ... 11-2
Updating User Tasks Using Update Tasks ... 11-3
Update Task Operations ... 11-3
How to Update a User Task Using Update Tasks ... 11-4
How to Configure Update Tasks ... 11-4

12 Working with Services and References

Introduction to Services and References ... 12-1
Introduction to Services ... 12-2
Introduction to References ... 12-2
Introduction to Callbacks ... 12-2
Introduction to Service Adapters in Oracle BPM ... 12-3
Introduction to Oracle Mediator in Oracle BPM ... 12-5
Introduction to BPEL Processes in Oracle BPM ... 12-8
Using Services in Oracle BPM ... 12-9
Using References in Oracle BPM ... 12-9
Customizing Services and References ... 12-10

How to Customize a Service or a Reference 12-10
How to Customize an Operation 12-11
What Happens When You Customize a Service or a Reference 12-11

13 Using Business Rules

Introduction to Business Rules in Oracle BPM 13-1
Using Business Rules in a BPMN Process 13-2
Assigning an Existing Business Rule to a Business Rule Task 13-3
How to Assign an Existing Business Rule to a Business Rule Task 13-3
What Happens When You Assign an Existing Business Rule to a Business Rule Task 13-4
How to Edit the Business Rule Associated to a Business Rule Task 13-4
Creating a Business Rule from Oracle BPM Studio 13-4
How to Create a Business Rule from Oracle BPM Studio 13-5
How to Add Input and Output Arguments When Creating a Business Rule Component 13-5
How to Configure the Advanced Properties When Creating a Business Rule Component 13-6
What Happens When You Create a Business Rule Task from Oracle BPM 13-6

14 Sending Notifications

Introduction to Notifications 14-1
Sending Email Notifications 14-1
How to Send an Email Notification 14-2
How to Configure Email Notification General Properties 14-2
How to Configure Email Notification Content Properties 14-3
How to Configure Email Notification Attachment Properties 14-3
How to Configure Email Notification Header Properties 14-3
Sending a User Notification 14-4
How to Send a User Notification 14-4
How to Configure User Notification General Properties 14-4
How to Configure User Notification Properties 14-5
Sending an SMS Notification 14-5
How to Send an SMS Notification 14-5
How to Configure SMS Notification General Properties 14-6
Sending a Voice Notification 14-6
How to Send an Voice Notification 14-6
How to Configure Voice Notification General Properties 14-6
Sending an IM Notification 14-7
How to Send an IM Notification 14-7
How to Configure IM Notification General Properties 14-7

15 Using SOA Composites with BPM Projects

Introduction to SOA Composites 15-1
Understanding the Relationship Between SOA Composites and SOA Components 15-2
Working with SOA Components 15-2

BPMN Process in SOA Composites ... 15-3
How Do BPMN Errors Affect the SOA Composite Status ... 15-3
Opening the SOA Composite in a BPM Project ... 15-4
How to Open the SOA Composite in a BPM Project ... 15-4
Opening BPMN Processes from the SOA Composite in a BPM Project ... 15-4
How to Open a BPMN Process from the SOA Composite in a BMP Project ... 15-4
Adding a BPMN Process from the SOA Composite Editor ... 15-4
How to Add a BPMN Process from the SOA Composite Editor ... 15-4
What Happens When You Add a BPMN Process from the SOA Composite Editor ... 15-5
Integrating with BPEL Processes Using the SOA Composite ... 15-5
Adding a BPMN Process as a Partner Link in a BPEL Process ... 15-5
How to Add a BPMN Process as a Partner Link in a BPEL Process ... 15-6
What Happens When You Add a BPMN Process as a Partner Link in a BPEL Process ... 15-6
Connecting to a BPMN Process Using Web Services ... 15-6
Building a BPM Project ... 15-6
How to Build a BPM Project ... 15-7

Part V Controlling the Process Flow

16 Controlling the Process Flow

Introduction to Controlling the Process Flow ... 16-1
Gateways ... 16-1
Timer Events ... 16-1
Errors ... 16-2
Message Events ... 16-2
Send and Receive Tasks ... 16-2
Loop Markers ... 16-2
Multi-Instance Loop Markers ... 16-2
Suspending the Current Process Flow ... 16-2
Introduction to Loop and Multi-Instance Markers in Subprocesses ... 16-2
How to Configure Loop Markers ... 16-3
How to Configure Multi-Instance Markers ... 16-4
Suspending the Current Process Flow to Run an Alternative Process Flow ... 16-5
How to Configure a Flow Object to Suspend the Current Process Flow ... 16-5
How to Resume the Suspended Process Flow ... 16-5

17 Adding Delays, Deadlines, and Time-Based Cycles to a Process

Introduction to Timer Events ... 17-1
Adding a Delay to the Process Flow ... 17-2
How to Add a Delay to the Process Flow ... 17-2
What Happens When You Add a Delay to the Process Flow ... 17-3
Designing a Process to Start Based on a Time Condition ... 17-3
How to Design a Process to Start Based on a Time Condition ... 17-3

What Happens When You Design a Process to Start Based on a Time Condition 17-4
Configuring a Deadline for an Activity 17-4
How to Configure a Deadline for an Activity 17-4
What Happens When You Configure a Deadline for an Activity 17-5
Configuring a Deadline for a BPMN Process 17-5
How to Configure a Deadline for a BPMN Process 17-6
What Happens When You Configure a Deadline for a BPMN Process 17-6
Running Additional Activities 17-6
How to Run Additional Activities While an Activity is Running 17-7
What Happens When You Run Additional Activities While an Activity is Running 17-7
How to Run Additional Activities While a Process is Running 17-7
What Happens When You Run Additional Activities While a Process is Running 17-8
Configuring Timer Events 17-8
How to Configure a Timer Event To Use a Specific Date and Time 17-8
What Happens When You Configure a Timer Event to Use a Specific Date and Time 17-9
How to Configure a Timer Event to Use an Interval 17-9
What Happens When You Configure a Timer Event to Use an Interval 17-9
How to Configure a Timer Event to Run Periodically 17-10

18 Handling Errors

Introduction to Error Handling 18-1
Handling Errors Using Exceptions 18-2
Using Business Exceptions 18-2
Using System Exceptions 18-2
Typical Flow of an Exception 18-4
Typical Flow of an Exception Thrown in a Task 18-4
Typical Flow of an Exception in a Subprocess 18-5
Typical Flow of an Exception in a Reusable Process 18-5
Handling Exceptions in a Business Process 18-6
How to Handle an Exception Using a Boundary Error Catch Event 18-7
What Happens When You Handle an Exception Using a Boundary Catch Event 18-8
How to Handle an Exception Using an Event Subprocess 18-8
What Happens When You Handle an Exception Using an Event Subprocess 18-8
How to Configure an Error Event to Catch Business Exceptions 18-8
How to Configure a Catch Event to Catch System Exceptions 18-9
Configuring Catch Events to Recover from an Exception 18-10
Throwing Exceptions in Subprocesses or Reusable Processes 18-10
How to Throw an Exception 18-10
What Happens When You Throw an Exception 18-11
How to Create a Business Exception 18-12
What Happens When You Create a Business Exception 18-12
Handling Exceptions in Subprocesses 18-12
Handling Errors in a Peer Process Using Message Events 18-12

How to Handle Errors in a Peer Process Using Message Events 18-13
What Happens When You Handle Errors in a Peer Process Using Message Events 18-14

19 Using Fault Handling in BPM

Handling Faults with the Fault Management Framework 19-1
Designing Fault Policies for Oracle BPM Suite 19-1
Designing Composite Level Fault Policies 19-2
Designing Service Component Level Fault Policies 19-3
Designing Reference Level Fault Policies (Calling a BPM Process) 19-4
Designing Reference Level Fault Policies (Calling a File Adapter) 19-6
What You May Need to Know About the Difference Between Reference Naming Conventions in Oracle SOA Suite and Oracle BPM Suite 19-7

20 Communicating With Other BPMN Processes and Services

Introduction to Communication with Other BPMN Processes and Services 20-1
Introduction to Synchronous and Asynchronous Operations 20-2
Communicating With Other BPMN Processes and Services Using Message Events 20-2
Using Message Events to Invoke Asynchronous Services and Asynchronous BPMN Processes 20-4
How to Invoke Asynchronous Service Operation Using Message Events 20-5
How to Receive the Callback Operation of an Asynchronous Service Using Message Events 20-6
What Happens When You Invoke an Asynchronous Service Operation Using Message Events 20-6
How to Invoke an Asynchronous BPMN Process Operation Using Message Events 20-7
How to Invoke the Callback Operation of an Asynchronous BPMN Process Using Message Events 20-8
What Happens When You Invoke an Asynchronous BPMN Process Using Message Events 20-8
Using Message Events Configured as Boundary Events 20-9
Using Service Tasks to Invoke Synchronous Operations in Services and BPMN Processes 20-9
How to Invoke a Synchronous Service Operation Using a Service Task 20-10
What Happens When You Invoke a Synchronous Service Operation Using a Service Task 20-11
How to Invoke a Synchronous BPMN Process Operation Using a Service Task 20-11
What Happens When You Invoke a Synchronous BPMN Process Operation Using a Service Task 20-12
Communicating With Other BPMN Processes and Services Using Send and Receive Tasks ... 20-12
Using Send and Receive Tasks to Invoke Asynchronous Services and Asynchronous BPMN Processes 20-13
How to Use a Send Task to Invoke an Asynchronous Service Operation 20-14
How to Use the Receive Task to Invoke the Callback Operation of an Asynchronous Service 20-15

What Happens When You Invoke an Asynchronous Service Using Send and Receive Tasks 20-16
How to Use the Send Task to Invoke an Asynchronous BPMN Process Operation 20-16
How to Use a Receive Task to Invoke the Callback Operation of an Asynchronous BPMN Process 20-17
What Happens When You Invoke an Asynchronous BPMN Process Using Send and Receive Tasks 20-18
Introduction to Invoking a Process Using Call Activities 20-18
Invoking a Process Using Call Activities 20-18
How to Invoke a Process Using Call Activities 20-18
Introduction to Communication Between Processes Using Signal Events 20-19
Communicating Between Processes Using Signal Events 20-20
How to Broadcast a Signal to Multiple Processes 20-20
What Happens When You Broadcast a Signal 20-21
How to Configure Your Process to React to a Specific Signal 20-21
What Happens When You Configure a Process To React to a Specific Signal 20-21

21 Defining the Process Interface

Defining the Process Interface 21-1
Using Message Events to Define the BPMN Process Interface 21-2
Using Message Events to Define the Callback Interface for BPMN Processes 21-3
Using Message Events to Define Asynchronous Operations in a BPMN Processes 21-4
How to Configure the Start Operation of a BPMN Process as Asynchronous Using Message Events 21-4
How to Define a Callback Operation Using Message Events 21-5
What Happens When You Configure a BPMN Process Start Operation as Asynchronous Using Message Events 21-6
How to Add an Asynchronous Operation to a BPMN Process Interface Using Intermediate Message Events 21-6
What Happens When You Add an Asynchronous Operation to a BPMN Process Interface Using Message Events 21-7
Using Message Events to Define a Synchronous Operation in a BPMN Processes Interface 21-7
How to Configure the Start Operation of a BPMN Process as Synchronous Using Message Events 21-7
How to Configure the End Event of a Synchronous Process 21-8
What Happens When You Configure the Start Operation of a BPMN Process as Synchronous Using Message Events 21-9
Using Message Events with an Interface from the Business Catalog to Define Your Process Interface 21-9
How to Use an Interface from the Business Catalog to Define an Operation in a BPMN Process Interface Using Message Start and Catch Events 21-10
How to Configure a Message End or a Message Throw Event to Use an Interface from the Business Catalog Using Message Events 21-11

What Happens When You Use an Interface from the Business Catalog to Define an Operation ... 21-12
Defining the BPMN Process Interface Using Send and Receive Tasks ... 21-12
Defining the Callback Interface for BPMN Processes Using a Send Task ... 21-13
Defining Asynchronous Processes Operations Using Send and Receive Tasks ... 21-13
How to Define an Asynchronous Process Operation Using Send and Receive Tasks ... 21-14
How to Add an Asynchronous Process Operation to the Process Interface Using a Receive Task ... 21-15
How to Define a Callback Process Operation Using a Send Task ... 21-15
What Happens When You Define an Asynchronous Operation Using Send and Receive Tasks ... 21-16
Using Send and Receive Tasks to Define a Synchronous Operation in a BPMN Process ... 21-16
How to Configure a Process Operation as Synchronous Using Send and Receive Tasks. 21-17
What Happens When You Define a Synchronous Operation Using Send and Receive Tasks ... 21-17
Using Send and Receive Tasks with an Interface from the Business Catalog to Define Your Process Interface ... 21-18
How to Use an Interface from the Business Catalog to Define an Operation in a BPMN Process Interface Using Send and Receive Tasks ... 21-19
How to Configure a Message End or a Message Throw Event to Use an Interface from the Business Catalog Using Send and Receive Tasks ... 21-19
What Happens When You Use Send and Receive Tasks with an Interface from the Business Catalog to Define an Operation ... 21-20
Defining the Process Input and Output ... 21-21
How to Add Input and Output Arguments to a BPMN Process ... 21-21
How to Edit the Input and Output Arguments of a BPMN Process ... 21-21
How to Delete an Input or Output Argument of a BPMN Process ... 21-22

22 Communicating Business Processes Using Correlations

Introduction to Correlations ... 22-1
Understanding the Components of a Correlation ... 22-2
Typical Design Workflow ... 22-2
Defining Correlations for a BPMN Element ... 22-3
How to Define a Correlation for a Flow Object ... 22-3
How to Define a Correlation Using Simple Mode ... 22-3
How to Define a Correlation Using Advanced Mode ... 22-4
Creating Correlations Keys ... 22-5
How to Create a Correlation Key ... 22-5
How to Configure a Correlation Key ... 22-5

23 Defining Conversations

Introduction to Conversations ... 23-1
Defining the Default Conversation ... 23-2

Understanding the Different Types of Conversations 23-2
Creating Conversations 23-2
How to Create a Conversation 23-2
Defining Conversations for a BPMN Element 23-3
How to Define a Conversation for a BPMN Element 23-3
What Happens When You Define a Conversation for a BPMN Element 23-4
Viewing the Collaboration Diagram 23-4
How to View the Collaboration Diagram 23-4
How to Hide a Collaboration 23-4
How to Show a Collaboration 23-4

24 Writing Expressions

Introduction to Expressions in Oracle BPM 24-1
Writing Conditions in Conditional Sequence Flows 24-3
How to Implement a Conditional Sequence Flow 24-3
Writing Expressions in Complex Gateways 24-3
How to Implement a Complex Gateway 24-3
Writing Expressions in Timer Events 24-4
How to Use an Expression in a Timer Event 24-4
Writing Expressions in Data Associations 24-5
How to Use an Expression in a Data Association 24-5
Writing Conditions in Loop and Multi-Instance Markers in Subprocesses 24-6
How to Configure Loop Markers 24-6
How to Configure Multi-Instance Markers 24-7
Writing Expressions and Conditions Using the Simple Expression Builder 24-8
How to Use a Data Object in an Expression 24-9
How to Use a Function in an Expression 24-9
Simple Expression Builder Supported Operators 24-10
Operators Precedence 24-12
Simple Expression Builder Supported Functions 24-12
String Functions 24-12
Numeric Functions 24-13
DateTime and Duration Functions 24-15
Writing Expressions Using the XPath Expression Builder 24-18
How to Add a Variable to an XPath Expression 24-19
How to Use a Function in an XPath Expression 24-19
Using Arrays 24-20
Accessing an Attribute of an Element Within an Array 24-20
Obtaining the Length of an Array 24-21
Using Literals 24-21
Using String Literals 24-21
Using Time Literals 24-21
Using Duration Literals 24-22

Using Array Literals ... 24-23
XPath BPM Extension Functions ... 24-23
getActivityInstanceAttribute ... 24-24
getDataInput ... 24-24
getDataObject ... 24-24
getDataOutput ... 24-24
getGatewayInstanceAttribute ... 24-25
getProcessInstanceAttribute ... 24-25
getBusinessParameter ... 24-25

25 Writing BPM Scripts

Introduction to BPM Scripting ... 25-1
Introduction to the BPM Code Editor ... 25-1
Introduction to the Scripting Catalog ... 25-2
Importing Custom Libraries ... 25-3
How to Import a Custom Library ... 25-3
Working with the Elements of a BPM Project ... 25-4
How to Work with Business Objects ... 25-4
How to Work with Business Parameters ... 25-5
Importing Business Objects from the Business Catalog ... 25-5
Predefined Variables ... 25-5
Implementing Script Tasks ... 25-7
How to Implement a Script Task ... 25-7
Type Description Mapping for XML Schema Types ... 25-7

26 Debugging a BPM Project

Introduction to Debugging a BPM Project ... 26-1
Adding a Breakpoint to a BPMN Flow Object ... 26-2
How to Add a Breakpoint to a BPMN Flow Object ... 26-2
Adding a Breakpoint to a BPMN Component ... 26-2
How to Add a Breakpoint to a BPMN Component ... 26-3
Disabling a Breakpoint ... 26-3
How to Disable a Breakpoint ... 26-3
Debugging a BPM Project ... 26-3
How to Attach a BPM Project to the Debugger ... 26-3

Part VI Using Human Interaction Components

27 Getting Started with Human Workflow

Introduction to Human Workflow ... 27-1
Introduction to Human Workflow Concepts ... 27-3
Introduction to Design and Runtime Concepts ... 27-3
Introduction to the Stages of Human Workflow Design ... 27-12

Introduction to Human Workflow Features 27-12
Human Workflow Use Cases 27-12
Introduction to Human Workflow Architecture 27-14
Human Workflow Services 27-14
Use of Human Task 27-17
Service Engines 27-18
Human Workflow and Business Rule Differences Between Oracle SOA Suite and Oracle BPM Suite 27-18

28 Designing Human Tasks in Oracle BPM

Introduction to Designing Human Tasks in Oracle BPM 28-1
Typical Design Workflow 28-1
Creating a Human Task from Oracle BPM Studio 28-2
How to Create a Human Task from Oracle BPM Studio 28-3
How to Configure the Outcome of a Human Task 28-4
How to Add a Parameter to Human Task 28-5
How to Configure the Outcome Target of a Human Task 28-5
What Happens When You Create a Human Task from Oracle BPM Studio 28-6
Editing a Human Task from Oracle BPM Studio 28-6
How to Edit a Human Task Using the User Task Properties Dialog 28-7
Creating a Human Task from the SOA Composite Editor 28-7
How to Create a Human Task from the SOA Composite Editor 28-7
What Happens When You Create a Human Task from the SOA Composite Editor 28-8
Implementing a User Task with an Existing Human Task 28-8
How to Implement a User Task With an Existing Human Task 28-8
What Happens When You Implement a User Task With an Existing Human Task 28-9
How to Associate the Process Payload to the Human Task Payload 28-9
Editing a Human Task Using the Human Task Editor 28-10
How to Edit a Human Task Using the Human Task Editor 28-10
Configuring a Human Task Using the Human Task Editor 28-11
How to Specify an E-mail Address for the Recipient of a Notification 28-12
How to Configure Oracle UCM Repository to Store Task Attachments 28-13
Working with Screenflows 28-14
Creating a Screenflow 28-14

29 Configuring Human Tasks

Accessing the Sections of the Human Task Editor 29-1
How to Access the Sections of the Human Task Editor 29-2
Specifying the Title, Description, Outcome, Priority, Category, Owner, and Application Context 29-3
How to Specify the Title, Description, Outcome, Priority, Category, Owner, and Application Context 29-4
How to Specify a Task Title 29-4

How to Specify a Task Description ... 29-5
How to Specify a Task Outcome ... 29-6
How to Specify a Task Priority ... 29-8
How to Specify a Task Category ... 29-8
How to Specify a Task Owner ... 29-8
How To Specify an Application Context ... 29-14
Specifying the Task Payload Data Structure ... 29-15
How to Specify the Task Payload Data Structure ... 29-15
Assigning Task Participants ... 29-17
How to Specify a Stage Name and Add Parallel and Sequential Blocks ... 29-19
How to Assign Task Participants ... 29-20
How to Configure the Single Participant Type ... 29-21
How to Configure the Parallel Participant Type ... 29-35
How to Configure the Serial Participant Type ... 29-39
How to Configure the FYI Participant Type ... 29-43
Selecting a Routing Policy ... 29-44
How to Route Tasks to All Participants in the Specified Order ... 29-46
How to Specify Advanced Task Routing Using Business Rules ... 29-50
How to Use External Routing ... 29-55
How to Configure the Error Assignee and Reviewers ... 29-57
Specifying Multilingual Settings and Style Sheets ... 29-59
How to Specify WordML and Other Style Sheets for Attachments ... 29-60
How to Specify Multilingual Settings ... 29-60
Specify What to Show in Task Details in the Worklist ... 29-61
Escalating, Renewing, or Ending the Task ... 29-62
Introduction to Escalation and Expiration Policy ... 29-62
How to Specify a Policy to Never Expire ... 29-63
How to Specify a Policy to Expire ... 29-63
How to Extend an Expiration Policy Period ... 29-64
How to Escalate a Task Policy ... 29-64
How to Specify Escalation Rules ... 29-65
How to Specify a Due Date ... 29-66
Specifying Participant Notification Preferences ... 29-67
How to Notify Recipients of Changes to Task Status ... 29-69
How to Edit the Notification Message ... 29-71
How to Set Up Reminders ... 29-71
How to Change the Character Set Encoding ... 29-72
How to Secure Notifications to Exclude Details ... 29-72
How to Display the Oracle BPM Worklist URL in Notifications ... 29-72
How to Make Email Messages Actionable ... 29-72
How to Send Task Attachments with Email Notifications ... 29-73
How to Send Email Notifications to Groups and Application Roles ... 29-73
How to Customize Notification Headers ... 29-74

Specifying Access Policies and Task Actions on Task Content 29-74
How to Specify Access Policies on Task Content 29-74
Creating and Implementing Digital Certificates 29-78
How to Create a Digital Certificate Authority 29-79
How to Create Digital User Certificates 29-80
How to Generate Digital Certificate Revocation List 29-80
How to Specify a Certificate Authority 29-81
How to Specify a Workflow Digital Signature Policy 29-81
Specifying Restrictions on Task Assignments 29-83
How to Specify Restrictions on Task Assignments 29-83
Specifying Java or Business Event Callbacks 29-84
How to Specify Callback Classes on Task Status 29-84
How to Specify Task and Routing Customizations in BPEL Callbacks 29-88
How to Disable BPEL Callbacks 29-89
Storing Documents in Oracle Enterprise Content Management 29-89
How to Configure Oracle UCM Repository to Store Task Attachments 29-90

30 Working with Guided Business Processes

Introduction to Guided Business Processes 30-1
Guided Business Process Design Time Architecture 30-4
Components of a Guided Business Process 30-5
Guided Business Process Runtime Architecture 30-5
Guided Business Process Use Cases 30-10
Online Public Sector Form Processing 30-10
Online Loan Application Procedure 30-11
Standards and Guidelines for Working with Guided Business Processes 30-13
The Typical Flow of Developing a Guided Business Process 30-13
Introduction to Developing a Guided Business Process 30-14
Developing a BPMN Guided Business Process 30-15
How to Develop a BPMN Guided Business Process 30-15
What Happens When You Develop a BPMN Guided Business Process 30-15
How to Add a New Milestone to a Guided Business Process 30-15
What Happens When You Add a Milestone to a Guided Business Process 30-16
How to Add a User Task to a Milestone 30-16
What Happens When You Add a User Task to a Milestone 30-16
How to Move a User Task to Another Milestone 30-16
What Happens When You Move a User Task to Another Milestone 30-17
How to Order the Milestones in a BPMN Guided Business Process 30-17
What Happens When You Order the Milestones in a Guided Business Process 30-17
How to Delete a Task from a Guided Business Process 30-17
What Happens When You Delete a Task from a Guided Business Process 30-17
How to Delete a Milestone 30-17
What Happens When You Delete Milestone 30-18

How to Configure an Optional Task 30-18
What Happens When You Configure an Optional Task 30-18
How to Configure a Parallel Task Flow in a BPMN Guided Business Process 30-18
How to Branch the Task Flow in a BPMN Guided Business Process 30-19
How to Configure a Task to Display a Blocked Icon 30-19
What Happens When You Configure a Task to Display a Blocked Icon and Message 30-19
How to Configure an Icon for a Guided Business Process 30-19
What Happens When You Configure an Icon for a Guided Business Process 30-20
How to Configure an Icon for a Milestone 30-20
What Happens When You Configure an Icon for a Milestone 30-20
How to Configure the Display Mode for a Guided Business Process 30-20
What Happens When You Configure the Display Mode for a Guided Business Process. 30-21
How to Configure the Display Mode for a Milestone 30-21
What Happens When You Configure the Display Mode for a Milestone 30-21
How to Configure the Display Mode for a User Task 30-21
What Happens When You Configure the Display Mode for a User Task 30-22
How to Configure the Task Access Mode for a Guided Business Process 30-22
What Happens When You Configure the Task Access Mode for a Guided Business Process 30-22
How to Localize a BPMN Guided Business Process 30-23
How to Localize a Milestone 30-24
How to Localize a User Task 30-24
What Happens When You Localize a Guided Business Process 30-25
Configuring Activity Guide Properties 30-25
Deploying a Guided Business Process to Oracle WebLogic Server 30-27
How to Deploy a Guided Business Process 30-27
What Happens When You Deploy a Guided Business Process to Oracle WebLogic Server 30-28
Testing Guided Business Processes 30-28
What Happens When You Create a Guided Business Process Instance 30-28

31 Building a Guided Business Process Client Application

Introduction to Building a Guided Business Process Client Application 31-1
Developing a Guided Business Process Client Application with Oracle ADF 31-1
How to Develop a Guided Business Process Client Application 31-1
What Happens When You Develop a Guided Business Process Application with Oracle ADF 31-4
What Happens at Runtime: How a Guided Business Process Application Is Developed with Oracle ADF 31-4
Securing the Guided Business Process Client Application 31-5
Localizing a Guided Business Process Client Application 31-5
How to Configure the Supported Locales for a Guided Business Process Client Application 31-6

Guided Business Process Runtime APIs 31-7
Guided Business Process query Service API 31-7
JNDI Names for the Guided Business Process Enterprise Java Beans 31-9
Developing an Example of a User Interface for Guided Business Process Tasks Using Guided Business Process Runtime Services 31-9
Using Guided Business Process Logging 31-13
How to Enable Client Side Logging 31-13
How to Enable Server-Side Logging 31-13
Configuring Log Levels 31-14
How to View Guided Business Process Log Messages 31-14
Understanding Guided Business Process Log Messages 31-14

32 Using Approval Management

Introduction to Approval Management 32-1
AMX Components 32-2
Understanding Approval Management Concepts 32-4
Task 32-4
Service Data Objects 32-6
Stages 32-7
List Builders 32-8
Task Operations 32-9
Business Rules for Approval 32-9
Designing Approval Management Tasks in Oracle JDeveloper 32-11
Introduction to the Modeling Process 32-11
Before You Begin 32-12
Specifying General Information 32-12
Specifying Task Parameters 32-14
Specifying Mapped Attributes 32-16
Specifying Routing and Approval Policies 32-19
Defining Escalation and Renewal Policies 32-42
Specifying Notification Settings 32-42
Using Advanced Settings 32-42
Using the End-to-End Approval Management Samples 32-45
Using the User Metadata Migration Utility 32-45

33 Working with Adaptive Case Management

Introduction to Adaptive Case Management 33-1
Differences Between Adaptive Case Management and Business Processes 33-2
Adaptive Case Management Artifacts 33-3
Use Cases 33-4
Case State Model 33-4
Creating a Case 33-5
How to Create a Case 33-5

Configuring a Case 33-6
How to Edit a Case 33-6
Configuring Case General Properties 33-7
Case Deadlines 33-8
How to Configure the Case General Properties 33-8
How to Add Case Milestones 33-8
How to Define Case Outcomes 33-9
Configuring Case Data and Documents 33-9
Case Document Operations 33-9
Specifying Permission Tags for Case Documents 33-9
Using the BPM Database for Data Storage 33-10
Case Links in WebCenter Case Documents 33-10
Customizing Case Links in WebCenter Case Documents 33-10
Creating Case Fields in Oracle WebCenter Content 33-11
How to Configure Case Data 33-11
Configuring Case Flex Fields 33-12
How to Create a Case Flex Field 33-12
How to Configure the Document Location 33-13
How to Configure Enterprise Content Management 33-14
Configuring Case User Events 33-14
How to Add User Events 33-14
How to Publish Case User Events 33-15
Defining Case Stakeholders and Permissions 33-15
How to Add Case Stakeholders 33-18
How to Add Case Permissions 33-19
How to Manage Case Permissions 33-20
Defining Case Tag Permissions 33-20
How to Manage Case Tag Permissions 33-21
Localizing a Case 33-21
How to Configure Case Localization 33-23
Localizing Case Objects 33-24
Case Activities and Sub Cases 33-25
Case Activity and Sub Case Attributes 33-25
Predefined Case Activities 33-26
Specifying the Order of Case Activities 33-26
How to Promote a BPMN Process to a Case Activity 33-26
How to View the BPMN Process 33-27
How to Promote a Human Task to a Case Activity 33-28
How to View the Human Task 33-29
How to Create a Custom Case Activity 33-29
Creating Sub Cases 33-30
How to Create a Sub Case 33-30
Defining Input Parameters for Case Activities 33-30

How to Add a Case Activity Input Parameter 33-31
Defining Output Parameters for Case Activities 33-32
How to Add a Case Activity Output Parameter 33-33
Configuring Case Activities 33-33
How to Edit a Case Activity 33-33
Configuring Case Activity Basic Properties 33-33
Creating a Global Case Activity 33-34
Using Business Rules with Cases 33-35
Defining the Condition of a Case Business Rule 33-35
Understanding the Case Business Rule Dictionary 33-36
How to Generate a Case Business Rule Dictionary 33-37
Closing Cases 33-38
Integrating with Oracle BPM 33-39
Invoking a Case From a BPMN Process 33-39
How to Use Correlations with Case Events 33-39
Schema Reference 33-40
Simple Workflow Payload Schema 33-40
Email Notification Payload Schema 33-43
Example of Global Case Activity Metadata Schema 33-44
CaseEvent.edl 33-44

Part VII Appendices

A Process Star Schema Views

Standard Data Objects A-1
Composite-Specific Data Objects A-2

Preface

Welcome to *Developing Business Processes with Oracle Business Process Management Studio*. 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.

Related Documents

For more information, see the following Oracle resources:

Oracle Business Process Management

See the following for more information about Oracle BPM Suite:

- *Developing Business Processes with Oracle Business Process Composer*
- *Managing and Monitoring Processes with Oracle Business Process Management*

Oracle SOA Suite

- *Developing SOA Applications with Oracle SOA Suite*
- *Understanding Technology Adapters*
- *Designing Business Rules with Oracle Business Process Management*

Oracle SOA and BPM Suite Installation and Administration

- *Administering Oracle SOA Suite and Oracle Business Process Management Suite*
- *Installing SOA Suite and Business Process Management Suite Quick Start for Developers*
- *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.

What's New in This Guide

This preface introduces the new and changed features of Oracle Business Process Management Studio and provides pointers to additional information.

Screens shown in this guide may differ from your implementation, depending on the skin used. Any differences are cosmetic.

For a list of known issues (release notes), see `http://www.oracle.com/technetwork/middleware/soasuite/documentation/soaknown-2644661.html`.

New and Changed Features for 12*c* Release 1 (12.1.3)

For Oracle Business Process Management Studio Release 12*c* (12.1.3) this guide has been updated to include the following new and changed features:

- Support for business parameters. See Introduction to Business Parameters and Working with Business Parameters.
- Support for sharing Business Process Management (BPM) processes using the Process Asset Manager. See Sharing BPM Projects Using the Process Asset Manager.
- Support for business object inheritance. See Introduction to Business Object Inheritance and Working with Business Object Inheritance.
- Support for suspending the current process flow. See Suspending the Current Process Flow to Run an Alternative Process Flow.
- Support for recovering from an exception in a catch event. See Configuring Catch Events to Recover from an Exception.
- Support for BPM Scripting. See Writing BPM Scripts.
- Support for BPM project debugging. See Debugging a BPM Project.
- Additional Adaptive Case Management features including runtime case linking, support for sub cases, milestone due dates, and built-in support for Business Activity Monitoring analytics. See Working with Adaptive Case Management.

Other Significant Changes in this Guide for 12*c* Release 1 (12.1.3)

For Oracle Business Process Management Studio Release 12*c* (12.1.3) this guide has been updated to include the following ways:

- Basic types changed so procedures were updated. See Handling Information in Your Process Design.
- Business indicators changed so procedures were updated. See Using Process Analytics.
- The human tasks UI changed so procedures were updated. See Configuring Human Tasks.
- The Oracle Business Process Management Studio UI has changed significantly. Procedures describing how to accomplish tasks using the UI have changed accordingly.

Part I

Using Oracle BPM Studio

This part provides an overview of Oracle Business Process Management Studio. It also describes how create and work with BPM projects and how to create processes and use the process editor.

This part contains the following chapters:

- Introduction to Oracle BPM Studio
- Working with BPM Projects
- Working with Processes and the Process Editor

1

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:

- Working with Oracle BPM Suite
- Overview of the Application Development Life Cycle
- Introduction to the Oracle BPM Studio User Interface

Working with Oracle BPM Suite

Oracle BPM Suite consists of Business Process Composer for modeling processes and Business Process Management Studio for implementing the processes.

Oracle Business Process Composer is a web-based, collaborative application that enables business analysts to model business processes and perform the basic implementation of some of the artifacts used in the business process. For more information on Business Process Composer, see Developing Business Processes with Oracle Business Process Composer.

Oracle Business Process Management Studio is a desktop IDE application that enables process developers to implement the processes modeled by business analysts. Process developers use Oracle BPM Studio to edit the BPMN process models and artifacts in order to complete their implementation.

It is also possible to create and model business processes using Oracle BPM Studio, but Business Process Composer is better suited to modeling BPMN processes.

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

See Overview of the Application Development Life Cycle for more information on how these tools fit into the application development life cycle.

You can edit the following assets in BPM Studio:

- BPMN processes
- Reusable process components
- Process data
- Organizational data
- Human tasks
- Business rules
- Activity Guides

- Adaptive Case Management
- Simulations
- Web services, XML files and other artifacts needed for complex implementations

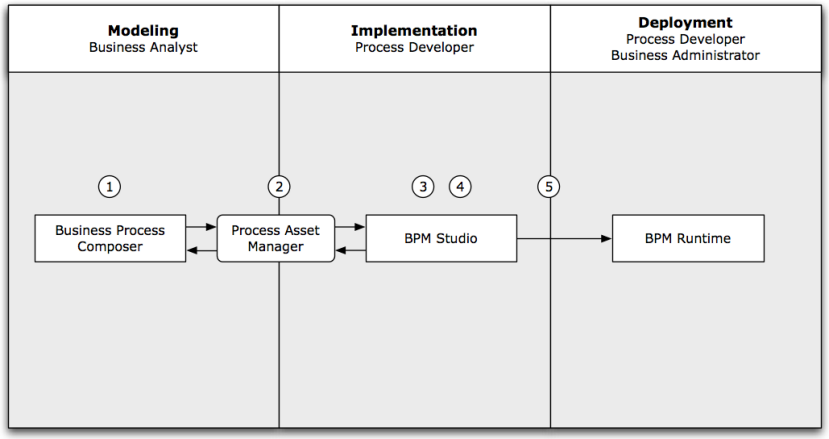
Overview of the Application Development Life Cycle

The life cycle of application development can be divided into three stages: modeling, implementation, and deployment. In a real-world application development environment, the distinctions between these stages may not be clearly defined.

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 and then passed back to the modeling and implementation stages for further refinement before being deployed to a production environment.

Figure 1-1 shows a typical workflow in which application design is performed using both Process Composer and BPM Studio.

Figure 1-1 Modeling, Implementation, and Deployment from Studio



The following steps describe each stage of the workflow:

1. Create process models using Business Process Composer (business analyst).
2. Share process models with process developers using the business asset manager.
3. Implement the required services and application resources using Oracle BPM Studio (process developer).

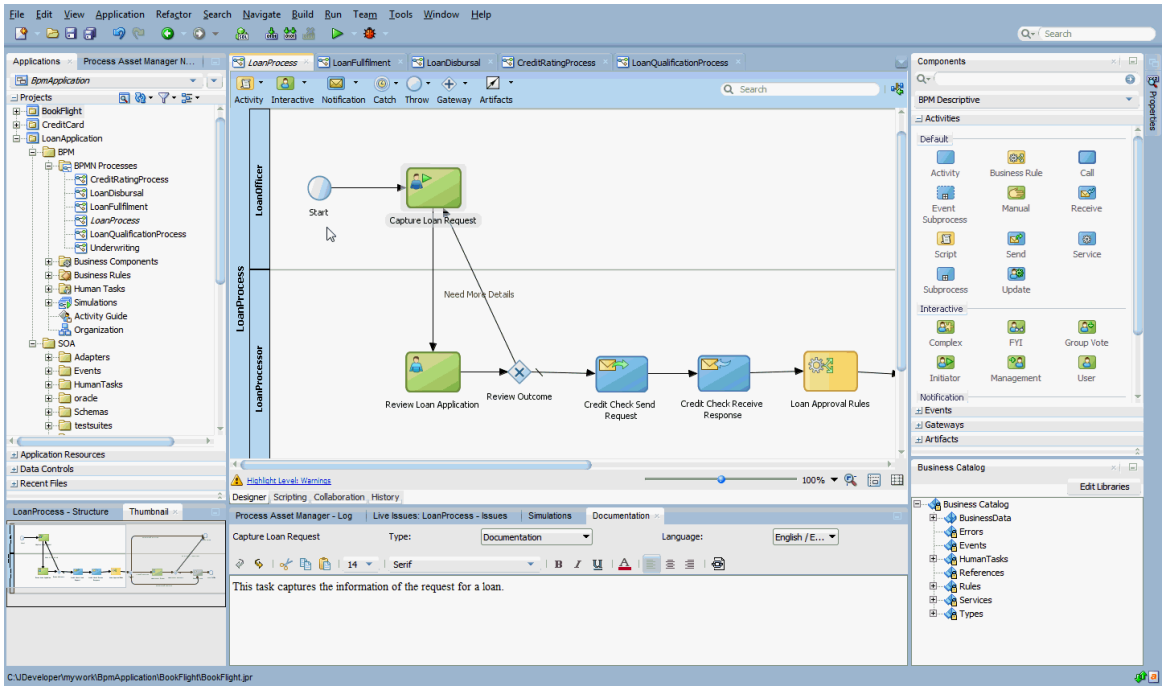
   After implementing the processes, developers can share the business processes with business analyst using the business asset manager.
4. Compile the application (process developer).
5. Deploy to Oracle BPM runtime (process developer / process administrator).

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.

Figure 1-2 shows the layout of Oracle BPM Studio displaying the LoanApplication example project.

Figure 1-2 Oracle BPM Studio



Applications Window

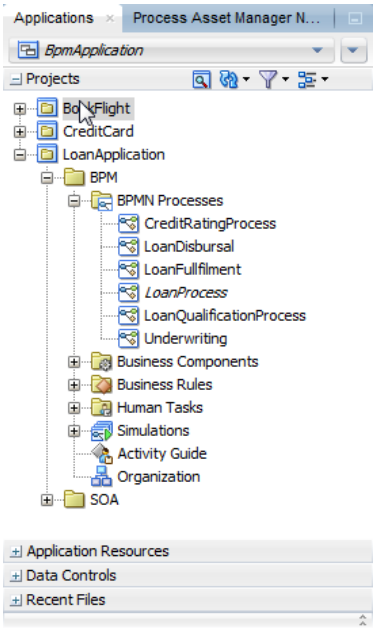
The Applications window displays a hierarchical view of the components of a project. The components displayed in the navigator are related to the modeling and implementation of business processes.

Table 1-1 Oracle BPM Project Components

Component	Description
BPMN Processes	Contains the business processes of this project. The Applications window shows the ID of the processes.
Business Components	Contains the business components defined for this project.
Simulations	Contains the process and project simulation models defined for this project.
Business Rules	Contains the business rules defined for this project.
Human Tasks	Contains the human tasks defined for this project.
Activity Guide	Contains the milestones defined for this project.
Organization	Contains the organizational elements defined for this project.

Figure 1-3 shows some of the files of the LoanApplication example that appear in the Applications window.

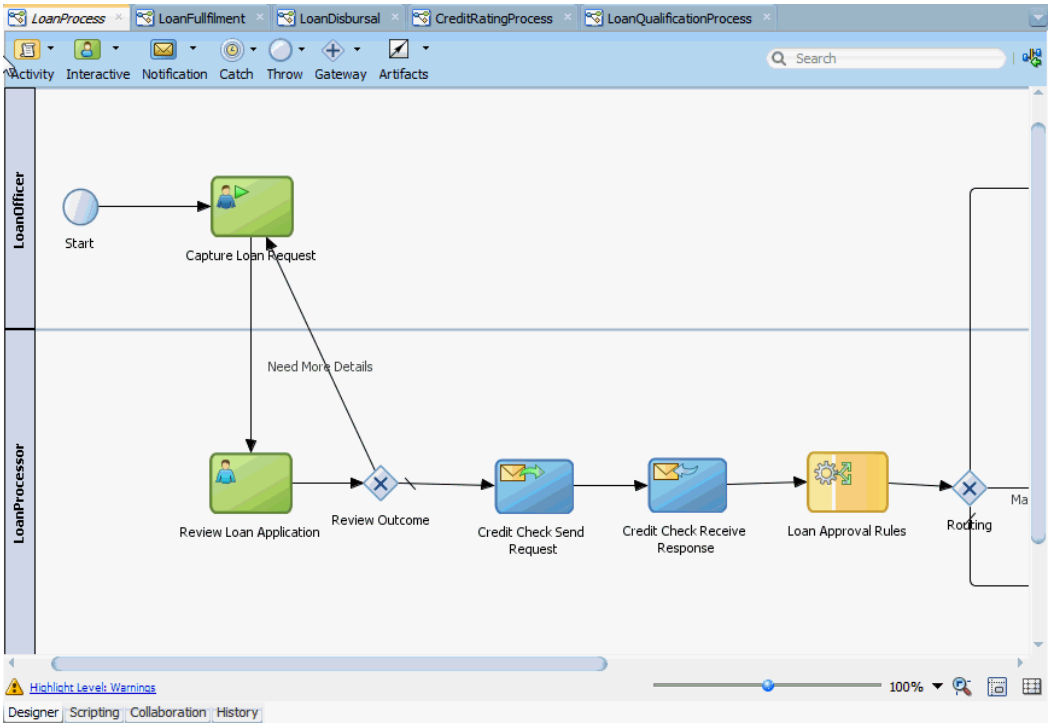
Figure 1-3 Application Navigator



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 1-4 shows the Request Quote example process opened in the process editor. You can also use the flow object drop-down menus on the toolbar to insert objects.

Figure 1-4 Process Editor

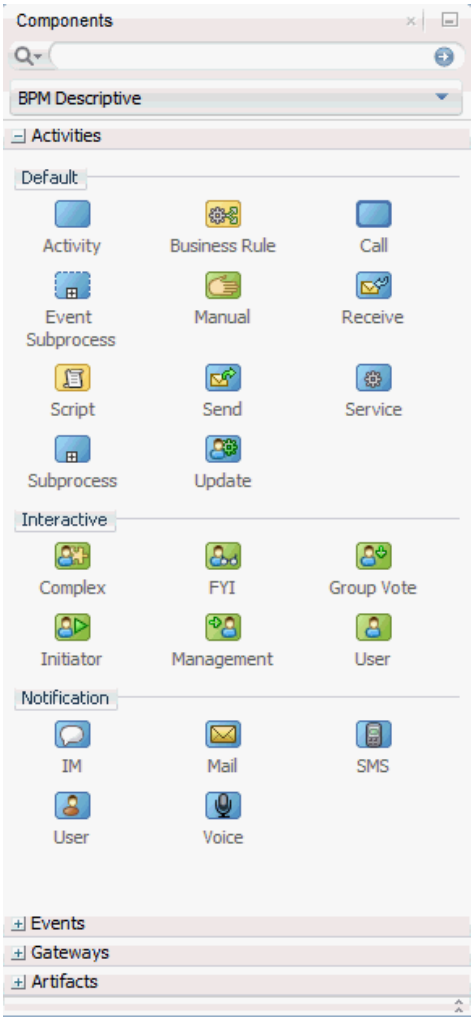


Components Window

The Components window 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 Components window to the process editor.

The BPMN flow objects are grouped in the Component window according to type, such as Activities, Events, and Gateways. A group may be further divided into subtypes. Figure 1-5 shows the Activities group in the BPM Components window expanded to show two types of activities, default and interactive.

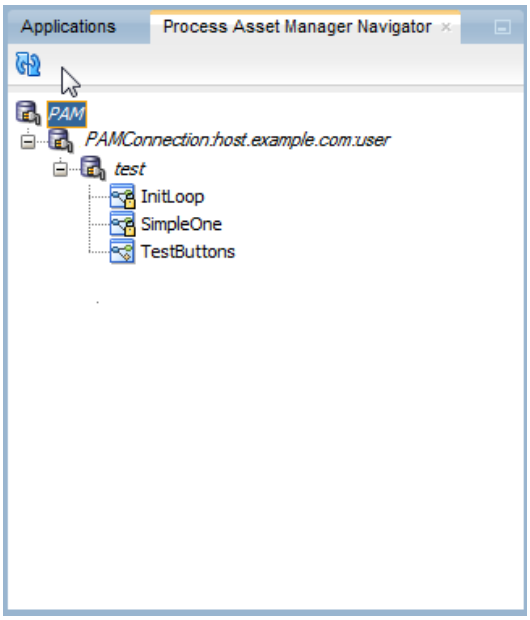
Figure 1-5 Component Window



Process Asset Manager Navigator

The Process Asset Manager Navigator allows you to view and use projects stored in the process asset manager repository. For more information about process asset manager, see Sharing BPM Projects Using the Process Asset Manager.

Figure 1-6 Process Asset Manager Navigator



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.

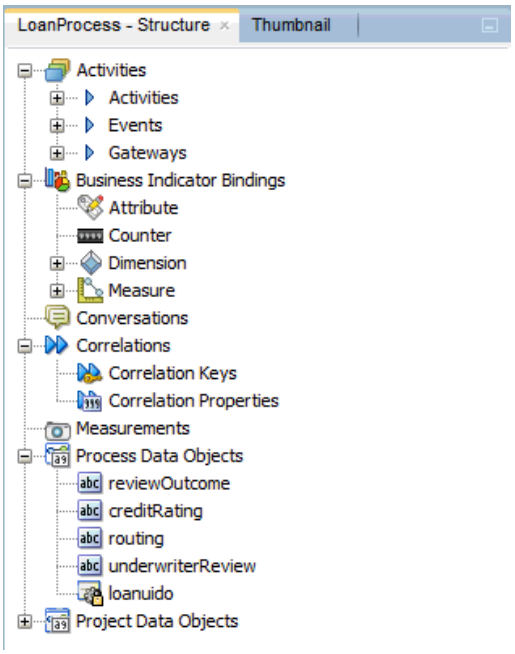
The structure components displayed in the Structure window are usually of components selected in the Project Navigator or Application Navigator.

You can perform a variety of tasks from the Structure 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 1-7 shows the Structure window when a BPMN process is selected in the Applications window.

Figure 1-7 Structure View



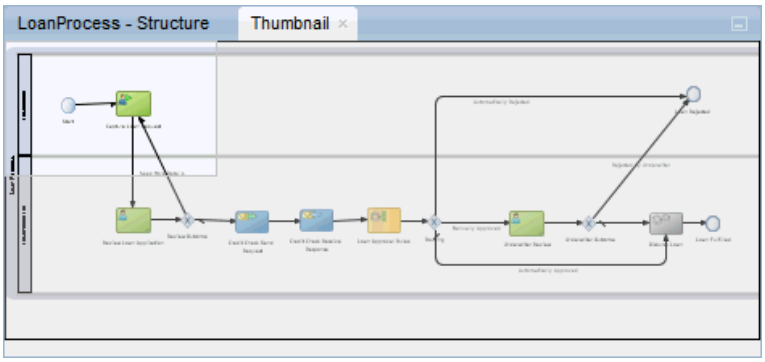
The structure of a process is divided into the following categories in the Structure window: Activities, Business Indicators, Conversations, Correlations, Measurements, Process Data Objects, and Project Data Objects. If the process is a reusable process, two additional categories are provided: Process Input Arguments and Process Output Arguments. A category may have subcategories, for example, Activities is further subdivided into Activities, Gateways, and Events. For information on the BPMN flow objects supported by Oracle BPM see the *Developing Business Processes with Oracle Business Process Composer*.

Thumbnail View

The Thumbnail window allows you to display a thumbnail representation of the active business process that is opened in the process editor. Because the Thumbnail view is synchronized with the process editor, it is useful for navigating large processes that do not fit in the process editor window.

By default the Thumbnail window is located next to the Structure window, as shown in Figure 1-8. A white rectangle on the Thumbnail view shows the section of the process that is currently in view in the process editor. By dragging the white rectangle on the Thumbnail view, you can pan the process in the process editor, change which section of the process to view in the process editor, and quickly navigate to a specific object on the process.

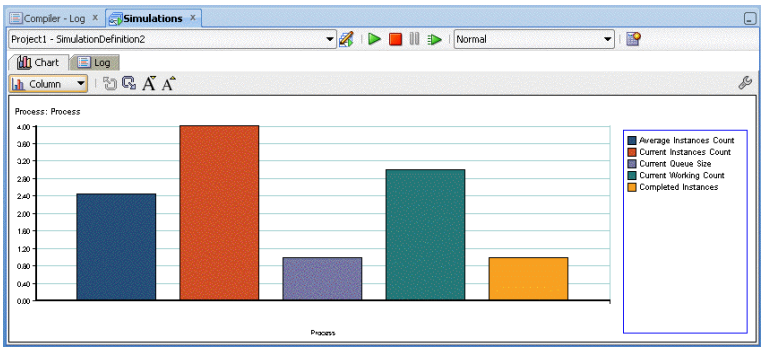
Figure 1-8 Thumbnail View



Simulation View

The Simulation View allows you to run and see the result of project simulation models. Figure 1-9, shows the results of a simulation displayed as a bar chart.

Figure 1-9 Simulation View



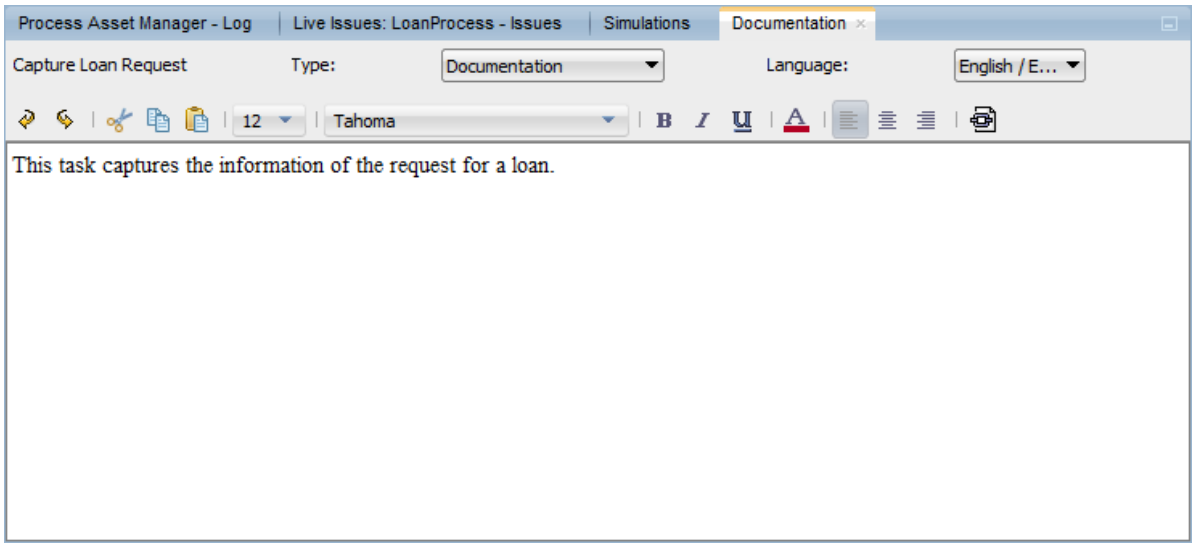
Log Window

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

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 1-10 shows the Documentation Window.

Figure 1-10 Documentation WIndow



2

Working with BPM Projects

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

This chapter includes the following sections:

- Introduction to BPM Projects
- Creating and Working with Projects

Introduction to BPM Projects

A BPM project is a container for the resources used to create and support business applications created using Oracle BPM.

You create new projects in Business Process Composer and edit them in Oracle BPM Studio to develop the process implementation.

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

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:

- BPMN Processes: includes BPM processes.
- Business Components: includes reusable components including services, adapters, business objects (data definitions), business events, enumerations and business exceptions.
- Business Rules: includes the business rules used in the BPM project.
- Human Tasks: includes the human tasks used to implement the interactive activities of the BPMN process in the BPM project.
- Simulations: includes the simulations models defined for a project and individual BPMN processes.
- Activity Guide: provides a milestone view of the BPMN process. This node appears after you create an activity guide. Note that you cannot use activity guides and adaptive case management at the same time.
- Organization: includes the organization elements used to mimic the organizational structure of your organization within BPMN process models.

- Resources: contain the XML transformations defined for your project.

It contains related business processes and process related assets such as human task definition, services, rule definitions, data definitions (Business Object), Business Exceptions, Business Events and more. These reusable, shared assets are contained in a Business Catalog inside the BPM Project. In addition to Business Catalog, the BPM Project contains simulations, organization model and XSLT mapping etc

Each of these resources are accessible from the Projects section in the Applications window. Additional application resources are accessible from other sections in the Applications window.

Sharing Projects Between Oracle BPM Users

Oracle BPM uses the process asset manager (PAM) repository to share projects and project templates between other Oracle BPM Studio and Business Process Composer users.

The process asset manager the design-time repository for Oracle Business Process Management Suite. By default it connects to an embedded source control system, you can configure it to use and external source control system and it connects with an identity manager for authentication and authorizations. The process asset manager provides a unified source for the development of processes across Business Process Composer and Oracle BPM Studio.

See 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 Sharing BPM Projects Using the Process Asset Manager for more information on the process asset manager.

Creating and Working with Projects

As you work with projects, you create new projects, open existing ones, import and export projects, and edit the preferences.

See Sharing BPM Projects Using the Process Asset Manager for more information on working with projects using the process asset manager.

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. Choose **File > From Gallery** from the menu.
2. Under **Categories**, select **BPM Tier**, then select **BPM Project** and click **OK**.
3. Enter a name for your project.

> **Note:** Use only ASCII chars A-Z, a-z, 0-9 and "_" for project names, and the name must begin with A-Z,a-z.

4. Ensure that BPM and SOA appear in the **Selected** column, then click **Next**.

5. Enter a name for the SOA composite.

   By default a BPM project is created and configured using the Composite With BPMN Process template.

6. Click **Finish**.

The new project is created and appears in the Applications window. After the project and composite file are created, the Create BPMN Process wizard starts automatically. You can choose to create a new process or cancel the wizard.

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

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. Choose **File > Open** from the menu.
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 Applications window.

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.

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

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

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, layout properties, severity level for process-related messages to highlight, and the default mode to use in data associations.
- Add localization languages to a project.

To edit project preferences:

1. In the Application window, right-click the project whose preferences you want to edit, then select BPM.
2. Select **Project Preferences**.
3. Edit the project preferences as necessary, then click **OK**.

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

3

Working with Processes and the Process Editor

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

This chapter includes the following sections:

- Getting Started with Processes
- Introduction to the Process Editor
- Working with Processes
- Working with Flow Objects in Your Process
- Working with Draft Processes
- Documenting Your Process

Getting Started with Processes

Business processes are the core components of process-based business applications created with Oracle BPM Suite.

Projects are higher level wrappers that contain all the resources of a business application, while the processes within the project determine how the application works.

Introduction to Business Processes

A business process is a sequence of tasks which, after they are performed, result in a well-defined outcome.

Business processes are generally created by business analysts who determine the business requirements that must be addressed and define the corresponding process flow using Business Process Composer. They then share the business process with process developers using the process asset manager, the design-time repository provided by Oracle BPM Suite.

The flow is defined by various BPMN flow objects. BPMN (Business Process Modeling Notation) is a graphical notation for capturing business process models. It captures the visual flow and the implementation properties. Oracle BPM Suite uses BPMN 2.0 for modeling and implementing BPMN processes.

Types of Processes

Use Oracle BPM to create different types of BPMN processes, depending on what needs to be done. Table 3-1 describes the types of processes supported by Oracle BPM.

Table 3-1 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. Immediately after the start event they have an initiator task that triggers the process when a participant submits a UI form.
Reusable Process	A process that can be invoked from a call activity. Reusable processes can only be invoked using the call activity. Reusable processes also begin and end with none start and end events. In Oracle BPM, a reusable process is identified as having only one none start event; in addition there is no initiator node in the process flow. If the none start event is changed to another type or if an initiator node is added to the process flow, the process is no longer considered to be reusable. For example, if a user task with the initiator pattern or a receive task implemented as a create instance is added immediately after the none start event, the process can no longer be reused or called by another process.

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 Applications window.
3. Right-click **BPMN Processes**, then select **New** then **BPMN 2.0 Process**.

   The BPMN 2.0 Process Wizard appears
4. Enter a name and optional description.
5. Select the type of process you want to create, then click **Next**.

   See Types of Processes for more information on process types.
6. Optionally, define the process arguments and initial implementation properties.

7. Define advanced properties, such as Process Sampling Points, Is Primary Process, Suspend instance on data association failure, and Service namespace.

   Select **Is Primary Process** to mark the process as the primary orchestrating process flow in the composite. When checked, Business Activity Monitoring (BAM) can filter and analyze data objects for only the primary process flow in the composite.

8. Click **Finish**.

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.

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 Applications window.
3. Expand **BPMN Processes**.
4. Double-click the process you want to open.

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

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 **BPMN Processes** in the Applications window.
3. Right-click the process you want to delete, then select **Delete**.

   The Confirm Delete dialog box appears.

4. Optionally, click Show Usages to view the usages of the process to delete.
5. Click Yes.

What You Need to Know About Deleting a Business Process

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

For example, if the deleted process is invoked from another process through a message throw event, reconfigure the invoking process so it no longer refers to the deleted process.

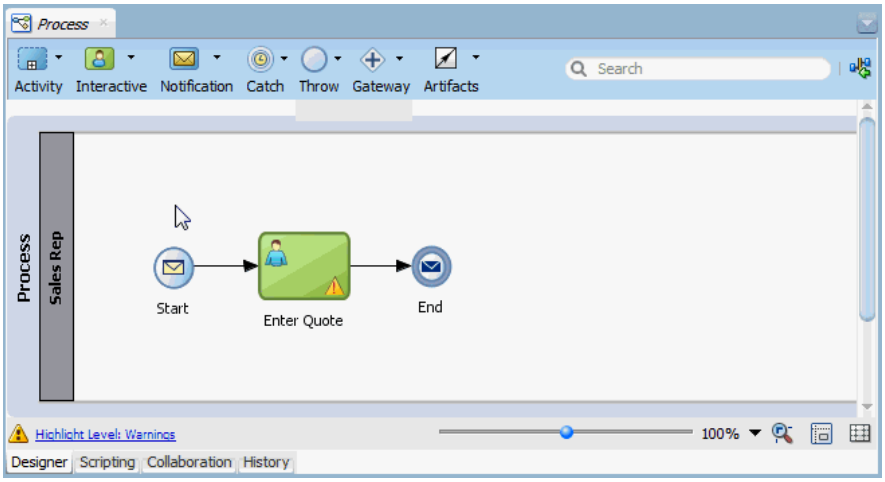
Review the usages of the process you want to delete before deleting it and removing the references.

Introduction to the Process Editor

The process editor has a canvas on which you can create a model of the process. You can also work with other BPMN processes through conversations.

Figure 3-1 shows an example of the process editor.

Figure 3-1 The Process Editor



There are two editor tabs at the bottom of each process editor:

- **Designer** editor tab: Provides a canvas and lets you create and model business processes using Business Process Management Notation and Modeling (BPMN). By default, a process opens in Designer mode.
- Scripting editor tab: Displays the scripts used in the process. You can select the script to display from the list on the top of the editor.
- **Collaboration** editor tab: Enables you to work with other BPMN processes and services through conversations.
- History editor tab: Displays the history of changes. You can also view a list of changes and compare different versions of the process.

The term 'process editor' refers to the Designer mode of the editor, unless explicitly specified otherwise.

When a process is opened in the process editor window, a flow object toolbar at the top of the canvas enables you to insert various BPM notations. The status bar below the canvas provides controls that enable you to show and fix errors or warnings, configure the layout, and change the zoom level. The process editor is also synchronized with a view of the process in the Thumbnail window. For more information about thumbnails, see Thumbnail View.

Flow Object Toolbar and Drop-down Menus

The flow object toolbar provides easy access to common BPM flow objects. The flow objects are available through the following drop-down menus on the toolbar:

- **Activity**: Includes activities such as Call, Business Rule, and Send.

- **Interactive**: Includes interactive activities such as User and Initiator.
- **Notification**: Includes activities such as Mail and Voice.
- **Catch**: Includes events such as Error and Message.
- **Throw**: Includes events such as Throw Signal and End Signal.
- **Gateway**: Includes Exclusive and Parallel.
- **Artifacts**: Includes Measurement and Sequence Flow.

The drop-down menus provide the same flow objects as found in the Component Palette. For more information on BPMN flow objects, see "BPMN Flow Object Reference" in the *Developing Business Processes with Oracle Business Process Composer*.

Search

Enables you do run a search for a specific flow object using its name.

Go To Composite Editor

This toolbar item opens the SOA Composite Editor.

Figure 3-2 Status Bar of Process Editor

Highlight Level

This status bar item enables you to change the severity level of messages to be highlighted in the process by special overlay symbols. For information on how to use this item, see How to Change the Highlight Level for Messages in a Process.

Zoom

This status bar item enables you to change the scale of the process. For information on how to use the zoom tool, see How to Change the Zoom Level in a Process.

Layout and Show Grid

These status bar items enable you to use and configure the automatic layout utility, and turn on or turn off a grid overlay in the process. For more information, see How to Configure Layout Properties and Use a Grid in a Process.

Working with Processes

As you work with processes in the process editor, you can open more than one process, export a process as an image, change the highlight level for messages, change the zoom level, or configure layout properties.

You can open one or more processes in the editor window. Each process is identified by a process name in a document tab at the top of the editor window. Only one process can be in focus (active) at any time.

How to Export a Process As an Image

You can export an entire process design to a PNG file only.

To export a process:

1. Open the process you want to export as an image.

2. Right-click anywhere on the canvas where there is no flow object and choose **Generate Process Image**.
3. In the Select Object File dialog box, navigate to the directory of your choice.
4. Enter a file name and click **OK**.

How to Change the Highlight Level for Messages in a Process

When a flow object in a process has an error or a warning message, the flow object icon in the process editor is highlighted by a red error symbol or a yellow warning symbol overlay.

You can change the severity level of messages to be highlighted in the process. The highlight levels available are:

- **None**: No errors or warnings are displayed.
- **Errors**: Only errors are displayed.
- **Warnings**: Both warnings and errors are displayed.

Note:

The highlight level you set in the process editor affects flow objects in the active process only. Other processes already opened in other process editor tabs are not affected. However the highlight level set in the active process editor does affect all new processes that you subsequently open or create in the same project. This is because the **Highlighting Level** preference in the Project Preferences dialog box is updated to the same value at the time you make the process-level highlight change.

To set the severity level of messages to be highlighted for all processes in a project, see How to Edit Project Preferences.

To change the highlight level for messages to show in a process:

1. Open the process in the process editor.
2. On the status bar, click **Highlight Level** to open a slider.
3. Slide to the level you want to use.

   Error or warning symbols on the affected flow objects turn off or turn on, according to the level you set.

   The current highlight level in the process is indicated by an icon and a label on the status bar of the process editor: a red 'x' circle for Errors and a yellow '!' (exclamation mark) triangle for Warnings. If no severity level is set, only the label 'None' displays.

Note:

To display messages associated with a flow object in the process, see How to Display and Fix Errors or Warnings in Flow Objects.

How to Change the Zoom Level in a Process

Three status bar items are provided in the process editor to enable you to change the scale of the active process quickly.

To change the zoom level in a process:

1. Open the process in the process editor.
2. To change the scale, use one of the following ways:
   - Use the slider to change the zoom level. Double-click the slider to return the scale to 100%.
   - Click the drop-down arrow next to the current percent value to select another percent zoom level.
   - Click the Zoom Reset icon to return the scale to 100%.

How to Configure Layout Properties and Use a Grid in a Process

[This is not consistent with the highlight level behavior, where the action affects the active process only. Here, changing the layout/grid in one process immediately affects the setting in all other opened processes in the editor window, except the run once utility which affects the active process only. Dev said this inconsistent behavior will be revisited later.]

You can configure a process to use the automatic layout utility: When automatic layout is turned on, JDeveloper automatically aligns the placement of flow objects horizontally and vertically as they are added to the process. When automatic layout is off, you can use a grid of horizontal and vertical lines in the process background to help you align flow objects.

You can also activate lanes optimization in the process, which means JDeveloper would remove unnecessary lanes by moving all activities to other named lanes, where possible.

To use the automatic layout utility and optimize lanes in a process:

1. Open the process in the process editor.
2. On the status bar click **Layout**.
3. In the pop-up dialog, click **OFF** to turn on the automatic layout utility or click **ON** to turn it off.
4. When automatic layout is on, select **Optimize lanes** to prune interactive roles with non-interactive activities.

   Lane optimization is available only when the automatic layout utility is turned on.
5. When automatic layout is off, click **Run layout once** to automatically align flow objects in the process without turning the automatic layout utility on in the process and project.

To use a grid in the process:

1. Open the process in the process editor.
2. On the status bar, select **Show Grid** to add a grid on the process design.
3. To turn off the grid, deselect **Show Grid**.

Note:

With the exception of the **Run layout once** utility, when you activate or deactivate lanes optimization, automatic layout or the grid in a process, similar configuration settings in the Project Preferences dialog box are simultaneously updated to the same values. This means all processes in the project will use the same settings until you change any preference in the dialog box or in the process editor status bar.

To set the layout or grid preferences in a project, see How to Edit Project Preferences.

Working with Flow Objects in Your Process

You can use several different methods to add flow objects to the process. You can also edit the properties, copy and paste, and mark flow objects as draft. You can also use sequence flows and fix errors or warnings.

For more information on BPMN flow objects, see "BPMN Flow Object Reference" in the *Developing Business Processes with Oracle Business Process Composer*.

How to Add Flow Objects from the Component Window

You can add BPMN flow objects from the Component window.

To add flow objects from the Component window:

1. Open the process into which you want to add a flow object.
2. From the **Window** menu, choose **Components**.
3. In the Component window, click the flow object to add.
4. In the process editor, click the point in the process where you want to add the flow object.
5. Edit the flow object properties as needed in the dialog box that opens, then click **OK**.

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

How to Add Flow Objects from the Process Editor Toolbar

The process editor toolbar contains drop-down menus for the same BPMN flow objects as found in the Component Palette. This is useful when you maximize the process editor window to full screen mode and you cannot use the Component Palette to add flow objects.

To add flow objects from the process editor toolbar:

1. Open the process where you want to add a flow object.
2. On the toolbar, click the drop-down arrow next to the flow object icon that you wish to add. Then choose an object from the drop-down menu.

   The menu icon on the toolbar changes to the icon of the object you selected. The cursor also changes to the same icon. To deselect the chosen object without adding it to the process, press Esc.
3. On the process editor canvas, position the cursor at the point in the process where you want to add the flow object, and click to insert.
4. Edit the flow object properties as necessary in the dialog box that opens, then click **OK**.

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

Note:

The last object you chose from a drop-down menu becomes the default selected object for that menu. The default object is indicated by the menu icon on the toolbar. This means the next time you click that menu icon and then click the canvas, the default object is added to the process.

How to Add Flow Objects from a Context Menu

You can add BPMN flow objects using a context menu on the process editor canvas.

To add flow objects from a context menu:

1. Open the process to which you want to add a flow object.
2. Position the cursor at the point in the process where you want to add a flow object and right-click.
3. From the context menu, choose **Add Activity** and then choose a flow object from one of the submenus: **Tasks**, **Subprocess**, **Events**, **Gateways**.
4. Right-click the flow object you just added and choose **Properties**.
5. Edit the flow object properties as necessary in the dialog box that opens, then click **OK**.

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

How to Edit Flow Object Properties

You can use the Properties dialog box to 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**.

How to Display and Fix Errors or Warnings in Flow Objects

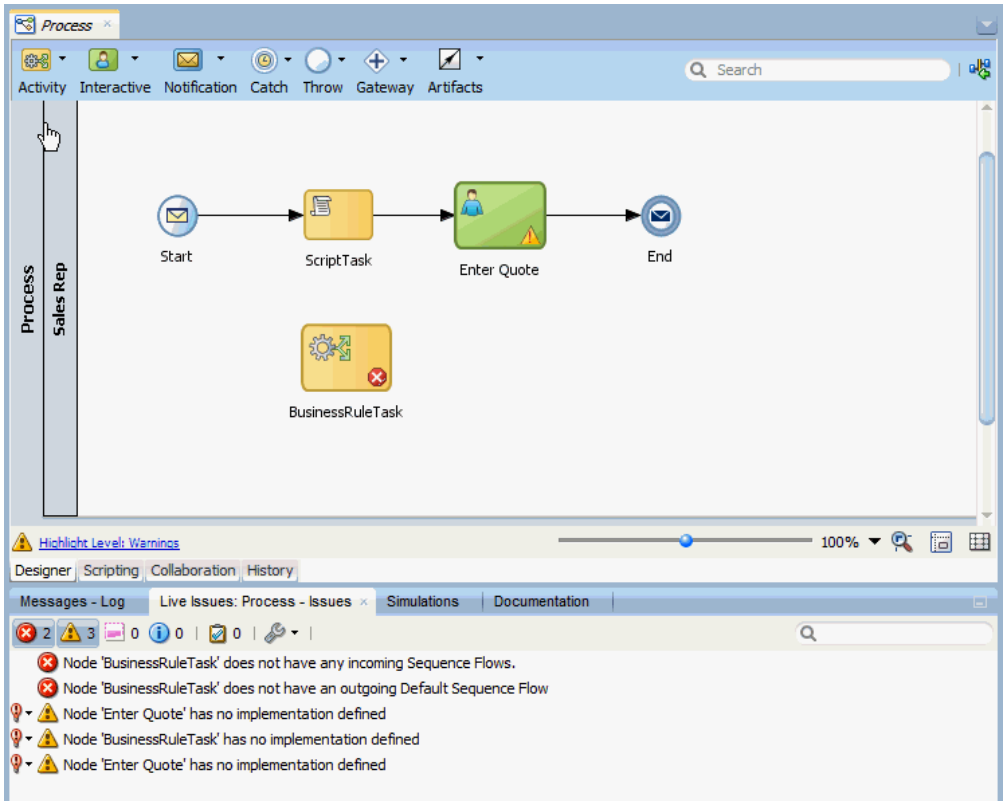
When a process or a flow object in the process has an error or a warning, the process icon in the Applications window or the flow object icon in the process editor has a red error 'x' circle symbol or a yellow warning '!' (exclamation mark) triangle symbol.

In the process editor, you can use the message area above the status bar to display error and warning messages related to a flow object or you can display a list of all the errors or warnings in the process. Then you can select a problem for fixing, if a fix suggestion is available.

To display and fix errors or warnings in flow objects:

1. Open the process containing flow objects with errors or warnings.
2. Change the severity level of messages to be highlighted in the process, if necessary. For more information, see How to Change the Highlight Level for Messages in a Process.
3. Select the **Live Issues** tab to view the messages in the process.

Figure 3-3 Process Editor With Live Issues Tab Selected



4. Select a flow object that has an error or a warning symbol.

   Messages related to the selected flow object are listed in the message area, as shown in Figure 3-3. Errors have a red 'x' circle symbol; warnings have a yellow '!' triangle symbol.

   Where a fix suggestion is available, a light bulb icon displays in the margin next to the message. Not all problems have fix suggestions. For example, nodes that do not have implementations, or nodes that have problems in user task properties and data associations offer fix suggestions.

5. Where applicable, click the light bulb icon in the margin next to a message.

   A popup opens, displaying one or more possible fix suggestions for the problem.

6. Select a fix suggestion in the popup.

7. Where applicable, do one of the following:

   - If a dialog box opens, make the changes as necessary, then click **OK**.
   - If you have multiple start events or unconditional outgoing sequence flows, select the one you want to keep when prompted.

Note:

To display all errors and warnings for all the flow objects that have problems, deselect the selected object in the process editor with the message area already expanded. If the message area is hidden, click **Show** without selecting any object in the process.

How to Mark and Unmark a Flow Object as Draft

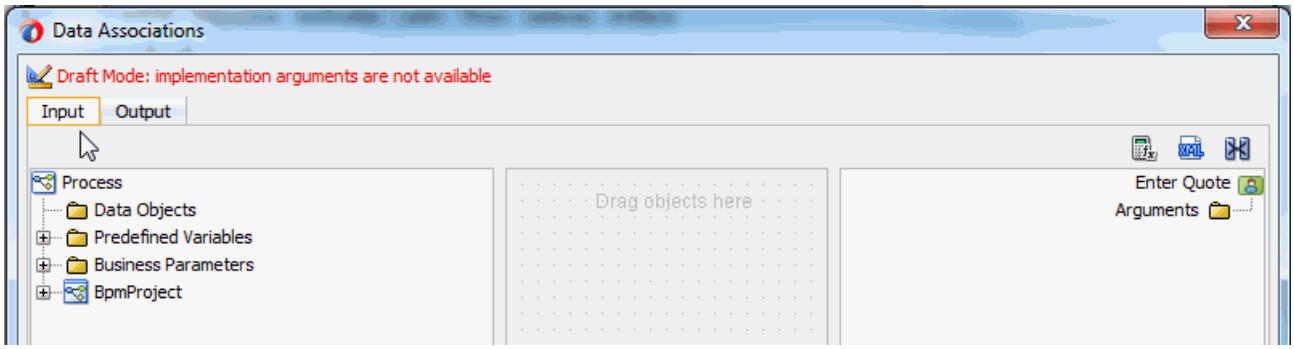
A flow object marked as Draft means the object has a default implementation where data object values can be set. A Draft flow object is indicated in a process by a gray icon in place of its default color icon.

Flow objects marked as Draft are considered to be unimplemented. However a process that contains Draft flow objects can still be deployed, but a warning will be issued.

Only events and activities can be marked as Draft. Events and activities that already have implementations defined can also be marked as Draft. Existing data associations, however, will be removed in the implementation when you change the Draft status.

Data objects in a Draft flow object can be initialized using custom assignments in the Data Associations dialog box, but implementation arguments are not available in the dialog box. An error message appears at the top of the Data Associations dialog box when you attempt to define data associations in a Draft flow object, as shown in Figure 3-4.

Figure 3-4 Data Associations Dialog



For more information about data objects and data associations, see Chapter 8, "Handling Information in Your Process Design".

To mark and unmark a flow object as draft:

1. Open the process containing the flow object you want to edit.
2. Right-click the flow object you want to mark and choose **Mark Node as Draft**.

   The color of the flow object icon changes to gray.
3. To unmark the flow object, right-click the icon and choose **Unmark Node as Draft**.

   The color of the flow object returns to its original color.

Note:

You can also toggle the Draft state of a flow object through the **Is Draft** check box in the Properties dialog box of the flow object. For information about how to use the Properties dialog box, see How to Edit Flow Object Properties.

How to Copy and Paste Flow Objects

You can copy and paste one or more flow objects within a process or between processes. Note that sequence flows, boundaries and measurements are copied only if the following conditions are met:

- the transitions, boundaries and measurements are selected
- all the elements the transitions, boundaries and measurements are attached to are also selected for copy

---

Note:

You cannot copy and paste objects inside an event subprocess.

To copy and paste a flow object in a process:

1. Open the process containing the flow object you want to copy. If you are copying between processes, open both processes.

2. Do one of the following:

   - Click the flow object you want to copy to select it.
   - To select more than one flow object, press and hold Ctrl and then click each object.
   - Click and drag your cursor around a group of flow objects to select the objects. A box is drawn on the canvas as you click and drag.

3. Right-click and choose **Copy**.

   If **Copy** does not appear in the context menu, this means the object you selected cannot be copied.

4. Navigate to the process where you want to insert the copied flow object.

5. At a blank location in the process canvas where you want to add the new flow object, right-click and choose **Paste**.

   If you paste outside a swimlane, a new role is added to the process and the new flow object is pasted there. For information about roles and swimlanes, see "BPMN Flow Object Reference" in the *Developing Business Processes with Oracle Business Process Composer*.

If necessary, use the **Run layout once** utility in the Layout popup to align the objects automatically. For more information about automatic layout in the process editor, see How to Configure Layout Properties and Use a Grid in a Process.

How to Add and Use Sequence Flows

A new business process is created with a start and end event already connected by a sequence flow. As you add flow objects to the process, inserting them anywhere along the sequence flow, JDeveloper automatically connects the new objects into the flow.

You can, however, create your own sequence flows where they are applicable and needed. For example, you may need to add a gateway with two conditional sequence flows and one default sequence flow.

For more information about controlling process flow with sequence flows and gateways, see "BPMN Flow Object Reference" in the *Developing Business Processes with Oracle Business Process Composer*.

You can also change the style of the arrow line used in a sequence flow and move a connected object from one sequence flow to another.

To connect two objects with a sequence flow:

1. Open the process.

2. In the Components window, expand **Artifacts** and then click **Sequence Flow**.

3. In the process editor, place the cursor over the first object where the outgoing sequence flow starts.

   The cursor changes to a + plus symbol when it is over an object where an outgoing sequence flow can start. For example, you can start an outgoing flow from a gateway object but not from a start event that already has an outgoing flow.

4. Click to anchor the start of the sequence flow, then drag the cursor to the second object where the sequence flow should end.

   As you drag the cursor, a line with an arrow at the end is drawn on the canvas. When the cursor is over an object where an incoming sequence flow can be placed, you should see a + plus symbol.

5. Click to anchor the end of the sequence flow.

   By default the two objects are connected by an orthogonal style sequence flow.

To change the style of a sequence flow:

1. Open the process.
2. Right-click the sequence flow line that you wish to modify.
3. From the context menu, choose **Style** and then choose **Straight**, **Curved** or **Orthogonal**.

   The existing style of the selected line is grayed out in the context submenu.

To move a connected object from one sequence flow to another:

1. Open the process.
2. Click the connected object that you want to move.

   The connected object in the sequence flow already has incoming and outgoing connecting flows.

3. Drag the object to a new point in the process.

   The new point must be located in another sequence flow in the same process. The target flow changes to blue to indicate that you are allowed to insert the dragged object into that new point.

4. Release the cursor when you have located a suitable target sequence flow.

   The selected object is disconnected from the original sequence flow and reconnected into the target sequence flow.

Working with Draft Processes

A draft process is a process that has one or more flow objects which do not have their implementation defined. With draft projects, you can test the parts of the process that have been completed before all flow objects have been implemented.

You create draft processes by marking one or more flow objects within the process as draft.

Introduction to Draft Processes

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

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

How to Mark a Flow Object as Draft

Flow objects are marked as draft within the basic properties.

To mark a flow object as draft:

1. Open your process
2. In the process editor, right-click the flow object you want to mark as draft.
3. Select **Mark Node as Draft**.

Documenting Your Process

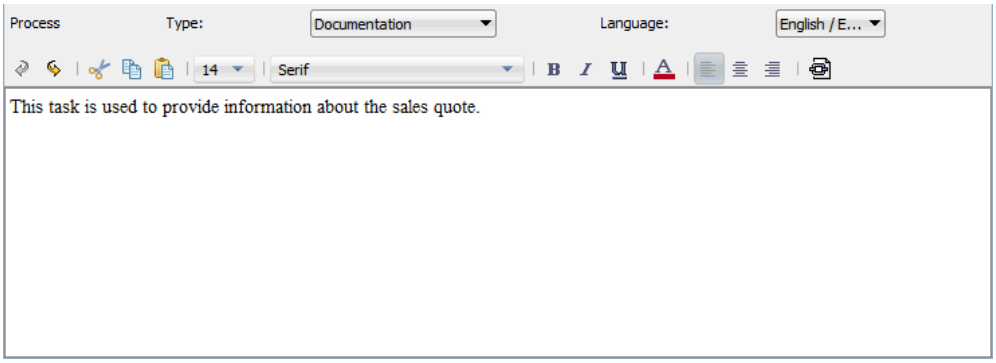
With the documentation editor, you can add process or use case documentation for the process and the flow objects.

The documentation editor contains a toolbar and editor pane where you enter the documentation.

Introduction to the Documentation Editor

Figure 3-5 shows the documentation editor.

Figure 3-5 The Documentation Editor



The toolbar allows you to select the type of documentation and the language, if you have defined additional languages for the product.

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

How to Add Documentation to Your Process

Use the Documentation editor to add one of these types of documentation to your process.

- **Documentation**: This is the documentation the process participants see using the Process 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 **Window** 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.

Generating Process Reports for Your Project

You can generate reports that list each process in your project and show detailed information about each process.

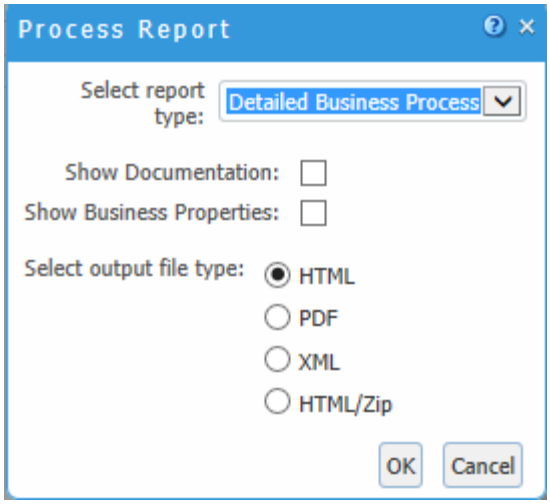
- Detailed Business Process
- Business Requirements
- Issues and Comments
- Data Objects
- Process vs Data
- Data vs Process
- Human Tasks vs Process
- Service vs Process
- User Tasks
- Process image
- RACI (responsible, accountable, consulted, and informed)

To generate a process report:

1. Right-click on the process and select **process report** or right-click on the project and select **BPM**, **Process Report**.

   The Process Report dialog is displayed.

Figure 3-6 Process Report Dialog



2. Select the following report details, then click **OK**:

   - Select report type
   - Show Documentation - Select to show descriptions
   - Show Business Properties
   - Select output file type

For more information about these reports, see Documenting Your Process.

Part II

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:

- Modeling Your Organization
- Handling Information in Your Process Design

4

Modeling Your Organization

This chapter describes how to model your business organization using Oracle BPM. It shows you how to define roles in your BPMN processes and how to add organizations to your BPM project.

This chapter includes the following sections:

- Introduction to Organizations
- Working with Organizations
- Introduction to Roles
- Working with Roles
- Introduction to Organizational Charts
- Introduction to Business Parameters
- Working with Business Parameters

Introduction to Organizations

Using Oracle BPM, you can create a model that represents the people, roles, and other aspects of your real-world organization. During deployment of your project, the components of the modeled organization are mapped to your real-world organization.

Oracle BPM organizations include:

- Roles
- Organizational Chart
- Business Parameters
- Holidays
- Calendars

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

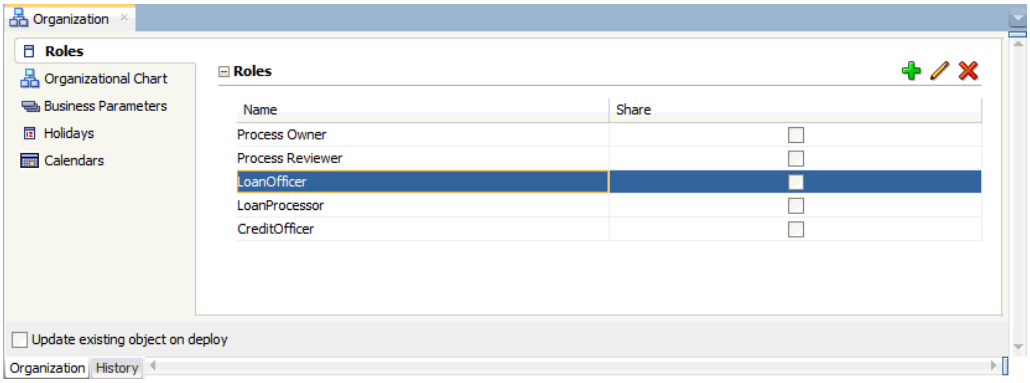
Note:

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

Introduction to the Organization Editor

Use the Organization Editor to create and edit the components within an organization. It contains tabbed pains for each of these components. Figure 4-1shows an example of the Organization editor with the Roles tab selected.

Figure 4-1 The Organization Editor



Working with Organizations

Organizations have several kinds of components: organizational units, calendars, and holidays.

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

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.

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. Select the check box 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.

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: Select the type of holiday you want to create. See Introduction to Holidays for information on the types of holidays.
   - Date: The date for this holiday rule. To specify a range, you must create a new entry for each day.
7. Click **OK**.

Introduction to Roles

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.

Working with Roles

Roles define who is responsible for performing the activities and tasks in your process. Adding members to a role identifies the members of your real-world organization who are responsible for the activities and tasks.

The following sections describe how to create and edit roles.

How to Create a New Role

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**.

How to Add Members to a Role

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, then click Select.
8. Click **OK**.

Introduction to Organizational Charts

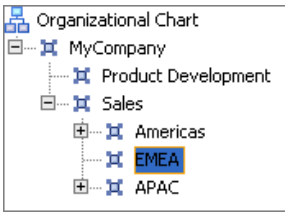
An organizational chart models the structure of your organization.

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

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 4-2 shows how an organization can be structured using organizational units.

Figure 4-2 Example of Nested Organizational Units



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

Introduction to Calendars

Calendars specify when resources in your organization are available to work, when they are on holiday, and so on. Calendars include:

- 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.

Introduction to Holidays

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

You can define two types of holidays:

- Fixed: holidays that occur on the same date every year.
- Common: holidays that do not occur on the same date every year.

Introduction to Business Parameters

Business parameters are global variables that specify information. Any process in a BPM project can use global variables.

For example, your company has a business process that requires approval for any employee to work more than 40 hours in a week. A business parameter MAX_WORKED_HOURS is defined and set to 40. The business parameter is used to drive your process flow. Suppose one month your the company has an unusual influx of work. You decide that all employees can work 45 hours in a week without requiring approval. The process owner monitoring the process changes the value of MAX_WORKED_HOURS to 45 to effect the change. You do not have to redeploy or modify the process.

Working with Business Parameters

Business parameters can be modified while a process is deployed and running, without changing the design of the process.

Changing the value of a business parameter affects the process behavior without having to change the process definition and without having to re-deploy the process.

How to Add a Business Parameter

Business parameters enable you to create variables to drive the process flow that you can modify while the process is running.

To add a business parameter:

1. In the Applications window, double-click the Organization node.

   The Organization editor opens.

2. Click the Business Parameters tab.

   The Business Parameters page appears.

3. Click the Add button.

   The Add Business Parameter dialog box appears.

4. Enter a name to identify the business parameter.

   The business parameter name must be in uppercase.

5. From the Type list, select a type for the business parameter.

   Available types are: string, boolean, int, and double.

6. Enter a default value.

   This is the value the business parameter uses when you deploy the BPM project.

7. Click OK.

   The business parameter appear in the Business Parameters table.

How to Assign a Value to a Business Parameter

You can assign a value to a business parameter in the following ways:

- Using data associations

  The business parameter appears in the input and output arguments section of the data association. To view the business parameters, expand the Business Parameters node. For more information on how to use data associations, see Introduction to Data Associations.

- Using BPM Scripts

  You can use BPM scripts to assign a value to a business parameter. For more information, see How to Work with Business Parameters.

5

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:

- Introduction to Handling Information in Your Process Design
- Introduction to Data Objects
- Working with Process Data Objects
- Introduction to Activity Instance Attributes
- Working with Activity Instance Attributes
- Introduction to Subprocess Data Objects
- Working with Subprocess Data Objects
- Introduction to Project Data Objects
- Working with Project Data Objects
- Introduction to Arguments
- Naming Conventions
- Scope and Access
- Introduction to Data Associations
- Introduction to Transformations
- Defining Transformations

Introduction to Handling Information in Your Process Design

Processes make use of information and also generate information from their tasks. Process flows can be affected by information values.

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

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

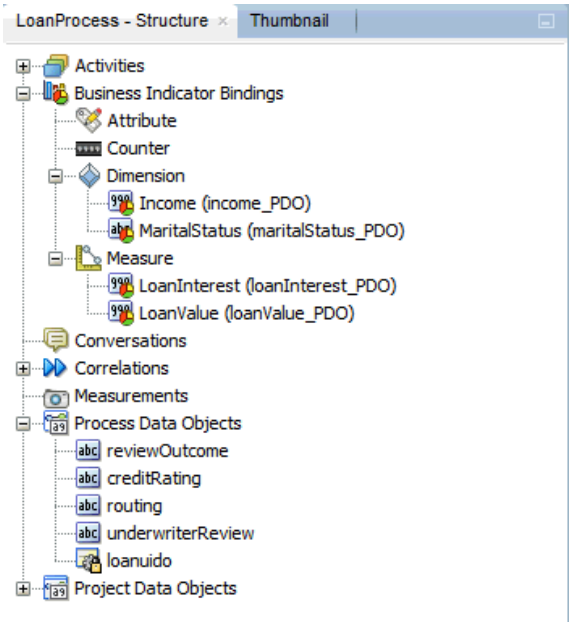
- Arguments

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

The Structure window shows the different data structures in your project: data objects, project data objects, and business indicators. For callable process it also shows arguments.

Figure 5-1 Shows the Structure window for a process that defines business indicators and process data objects.

Figure 5-1 Structure Window



Basic Data Objects versus Complex Data Objects

Basic data objects are defined using basic data types such as int, boolean or string.

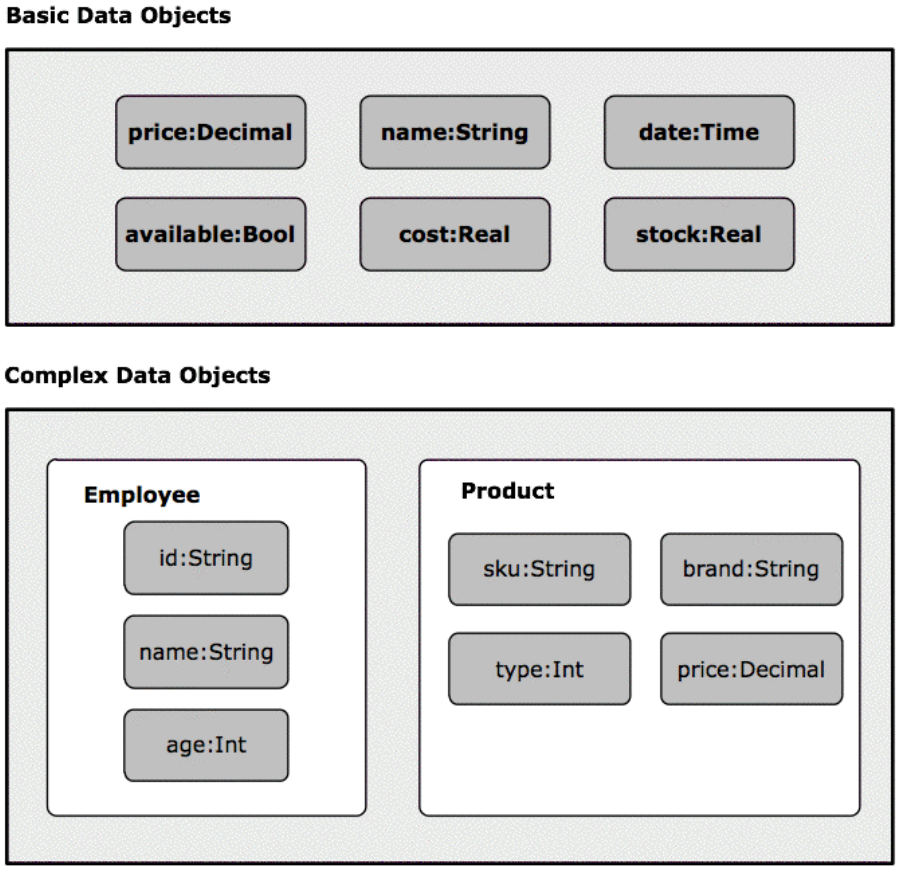
Complex data objects are defined using business objects and can group data. See Modeling Business Objects for more information on how to define business objects.

Business objects include 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 values they contain are specific to each instance of a process.

Figure 5-2 shows the relationship between basic data objects and complex data objects.

Figure 5-2 Basic Data Objects versus Complex Data Objects



Introduction to Data Objects

The main elements of a business process are tasks and information related to those tasks. The information in a process instance changes as the process executes. The information in a process instance defines the state of the process instance at any given time.

This information also determines how a process behaves and can change its flow of execution. You can monitor this information or store it to an external system.

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.

Supported Data Types for Data Objects

Data objects can be of the following data types:

- string
- int
- boolean
- double
- decimal
- time
- duration
- base64Binary
- Component (enables you to select a complex data type)

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, but they can be passed between different components and flow objects.

Default Values

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

Table 5-1 shows the default values for the supported data types.

Table 5-1 Default Values

Data Type	Default Value
string	""

Table 5-1 (Cont.) Default Values

Data Type	Default Value
time, date	'now'
int, double,decimal	0
boolean	false
duration	'0'

Working with Process Data Objects

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.

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

How to Add a Process Data Object

You can add a process data object to store a value to use in your BPMN process.

To add a process data object:

1. In the Applications window, 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.

   To use a complex type, select **<Component>**.
6. If you selected <Component>, select a complex type:
   a. Click the **Browse Types** button.

      The Browse Types dialog box appears.

   b. Select a type from the list or create a Business Object by clicking the **New** button next to the search list.

      To locate a type, enter the name in the **Search** text box. If the type does not exist, the name you typed appears in red.

   c. Click **OK**.

      The Browse Types dialog box closes and the complete name of the type you selected appears in the field next to the Browse Types button.
7. Optionally, check **Auto Initialize** to initialize the data object with a default value.
8. 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.

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 Applications window, 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 box to edit the data object name and type appears.
5. Make the changes you want.
6. Click **OK**.

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

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 Introduction to Data Associations for information on how to define a data association.

See Introduction to Transformations for information on how to define a transformation.

Introduction to Activity Instance Attributes

Some data, like the status of the process, applies to all the processes 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 5-2 provides detailed information about the activity instance attributes available for the different elements of a process.

Table 5-2 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.

Table 5-2 (Cont.) Activity Instance Attributes

Name	Type	Description	Availability
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.

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.

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.

Working with Subprocess Data Objects

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.

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

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 Applications window, select the process that contains the subprocess where you want to add a data object.
2. In the Structure window, expand the **Activities** node.

   The expanded node shows the subnodes Activities, Events and Gateways.
3. Expand the **Activities** subnode.
4. Expand the node that corresponds to the subprocess.
5. Right-click the **Data Objects** node located under the subprocess node.
6. Select **New**.
7. Provide a name to identify the new data object.
8. From the **Type** list, select a type.

   To use a complex type, select **<Component>**.
9. If you selected <Component>, select a complex type:
   a. Click the **Browse Types** button.

      The Browse Types dialog box appears.

   b. Select a type from the list or create a Business Object by clicking the **New** button next to the search list.

      To locate a type, enter the name in the **Search** text box. If the type does not exist, the name you typed appears in red.

   c. Click **OK**.

      The Browse Types dialog box closes and the complete name of the type you selected appears in the field next to the Browse Types button.
10. Optionally, check **Auto Initialize** to initialize the data object with a default value.
11. Click **OK**.

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 Applications window, select the process that contains the subprocess with the data object you want to edit.
2. In the Structure window, expand the **Activities** node.

   The expanded node shows the subnodes Activities, Events and Gateways.
3. Expand the **Activities** subnode.
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 box to edit the data object name and type appears.
8. Make the changes you want.
9. Click **OK**.

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 Applications window, select the process that contains the subprocess with the data object you want to delete.
2. In the Structure window, expand the **Activities** node.

   The expanded node shows the subnodes Activities, Events and Gateways.
3. Expand the **Activities** subnode.
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 delete.
7. Select **Delete**.

Introduction to Project Data Objects

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 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.

The processes in a BPM project only share the data definition of project data objects, not their actual values. Each BPMN process has its own copy of the project data object with a value that might or might not be different.

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 Using Process Analytics.

Supported Data Types for Project Data Objects

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

- string
- int
- boolean
- double
- decimal
- time
- duration
- base64Binary
- Component

Working with Project Data Objects

The main benefit of defining project data objects is that after publishing your project you can configure Process 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.

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

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.

How to Add a Project Data Object

To add a project data object:

1. In the Applications window, select a process from the Project whose project data object you want to edit.
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.

   Available types are: string, int, double, decimal, boolean, time.

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.

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 Applications window, select a process from the Project whose project data object you want to edit.
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 box to edit the project data object properties appears.
5. Make the changes you want.
6. Click **OK**.

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 Applications window, select a process from the Project whose project data object you want to edit.
2. In the Applications window, select a project.
3. In the Structure windows, expand the **Project Data Objects** node.
4. Right-click the project data object you want to delete.

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 Introduction to Data Associations for information on how to define a data association.

   See Introduction to Transformations for information on how to define a transformation.

Introduction to Arguments

You 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 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.

Naming Conventions

Names of process data objects, projects data objects, and arguments should follow certain conventions.

You should respect the following rules:

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

Scope and Access

The scope and access to process data objects, subprocess data objects, project data objects, and arguments 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 5-3 shows the scope of input arguments, process data objects, subprocess data objects and output arguments. The image shows a BPMN process and the red bars with the variables name show the scope where those variables are available.

Figure 5-3 Scope of the Data Structures in a Process

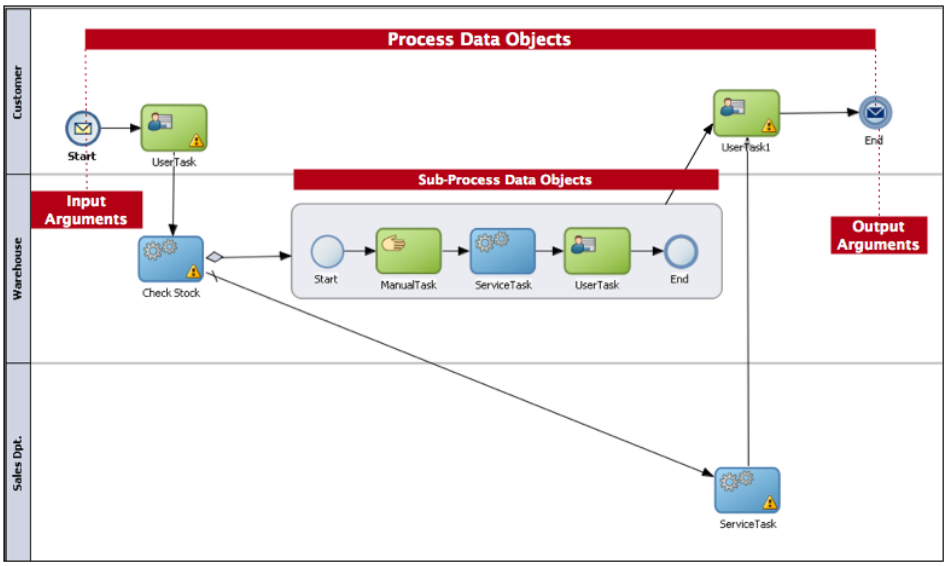
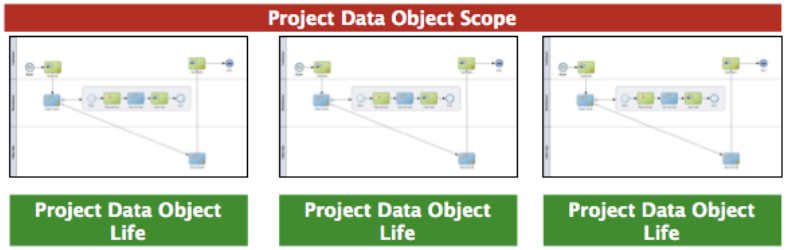


Figure 5-4 shows the scope and life span of project data object. The image the various BPMN processes in a BPM project, above them there is a red bar that indicates the scope of the project data objects, below each process there is a green bar that indicates their life span.

Figure 5-4 Scope and Life Span of Project Data Objects



Introduction to Data Associations

Data associations are used to pass the information stored in data objects in the certain contexts.

Data associations can be used to pass data:

- 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 5-4 lists the flow objects where you can define data associations. It also lists the objects implemented.

Table 5-3 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
Error events	Exception
Signal events	Event

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

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 5-5 shows the data association for the Enter Quote user task in the Sales Quote example.

Figure 5-5 The Data Association Editor

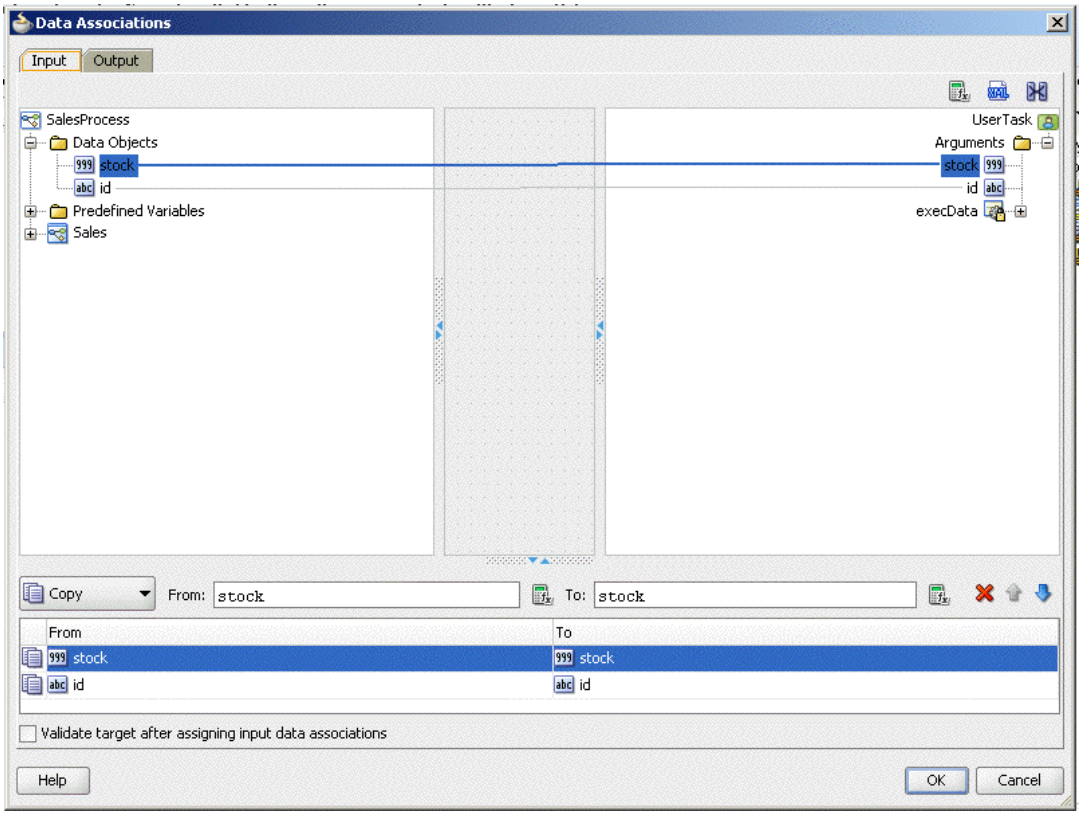


Table 5-4 describes the different areas of the data association editor.

Table 5-4 The Data Association Editor User Interface

UI Area	Description
Input Tab	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
Output Tab	Contains text boxes that display the data objects assigned as outputs from the service or process implemented in the flow object.

Table 5-4 (Cont.) The Data Association Editor User Interface

UI Area	Description
Flow Object Tree	Contains an Arguments node that lists all the expected argument. According to the tab you selected it lists input or output arguments. You can expand complex data objects to map to specific basic data objects within a complex data object.
Data Objects Tree	Displays all the data objects. This tree contains process data objects, predefined data objects and project data objects.You can expand complex data objects to map to specific basic data objects within a complex data object.

Introduction to Transformations

You can use XSL transformations to transform the values of a data object in the process before they are passed to a flow object as input arguments. XSL transformations can also transform the values of the output arguments of a flow object before you assign them to the data objects in the 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.

Defining Transformations

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 *Developing SOA Applications with Oracle SOA Suite* for more information on how to use the SOA XLS Editor.

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. Click the **Data Associations** link.

   The Data Associations dialog box appears.
3. Click the **Transformations** button located in the upper right corner and drag it to the target node.
4. Drop the transformation in the target node.

   The Create Transformation dialog box appears.

5. 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.

6. Click **Add**.

   The source appears in the Selected Elements list.

7. From the Target list, select a target to assign the result of the transformation.

8. 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.

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.

Part III

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:

- Running Simulations in Oracle BPM
- Using Process Analytics

6

Running Simulations in Oracle BPM

This chapter describes how to use simulations to predict the behavior of business processes under specified conditions, to verify that the output meets the metric objectives and identify any bottlenecks. It also explains how to run simulations to test the effects of changes on an existing process design.

This chapter includes the following sections:

- Introduction to Running Simulations in Oracle BPM
- Creating Simulation Models
- Configuring Boundary Events
- Creating Simulation Definitions
- Running Simulations
- Analyzing the Results of a Simulation

Introduction to Running Simulations in Oracle BPM

Run user-defined simulation in Oracle BPM to determine the efficiency of your processes. Your simulation can reflect either real or anticipated data.

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 shared resources, such as human participants

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 an outgoing flow

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.

You can create the simulation model based on a process using the Simulation Wizard or create from scratch.

How to Create a Simulation Model from a Business Process

To create a simulation model from a business process:

1. In the Applications window expand the BPMN Processes node.
2. Right-click the business process to use for the simulation.
3. Select **New Simulation** and then select **Wizard**.

   The Simulation Wizard appears.
4. Enter a name to identify the simulation model.
5. If you want to limit the number of instances to create, select **Specify Number of Process Instances to Be Created** and then specify the number of instances using the field below.
6. In the **User Tasks** section, customize the distribution, fixed resources and fixed based costs.
7. Click **Next**.

   The Simulation Definition wizard page appears.
8. Enter a name for the simulation definition.

9. Enter a duration for the simulation.
10. Select a start time using the arrow buttons or click the Calendar button next to the Start Time field.
11. To configure the simulation to wait for all instances to finish before ending, select **Left In-Flight Instances Finish Before Simulation Ends**.
12. Click **Next**.
13. Select the actions to perform after creating the simulation.

    Available options are:

    - Open Simulation Model
    - Open Simulation Definition
    - Start Simulation
14. Click **Finish**.

How to Create and Configure a Simulation Model

To create and configure a simulation model:

1. In the Applications window, expand the **Simulations** node.
2. Right-click the **Simulation Models** node.
3. Select **New** and then select From Gallery.

   The New Gallery dialog box appears.
4. In the Categories section, expand the BPM Tier node and select the Simulations node.
5. In the Items section, select BPMN Process Model Simulation and click OK.

   The Select Process dialog box appears.
6. Select a process.

   The Model Simulation dialog box appears.
7. Enter a name for the simulation model.
8. Click **OK**.

   The simulation model appears under the Simulation Models node in the Applications window, and the Simulation Model editor opens.
9. 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.

10. 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. Measured in salary per hour.
    - Queue Info: defines the simulated behavior of how process instances are queued for a given activity.
    - Outgoing Flows: determines the probability percentage of instances routed through the different outgoing sequence flows.
    - Threads: specifies the number of threads running this flow object. This tab is only available for flow objects that do not involve human interaction.
    - Instance Creation: enables you to specify the distribution curve to use to create instances for the simulation. This tab is only available for start events.

11. 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 6-1.

Table 6-1 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

Table 6-1 (Cont.) Options in the Simulation Model Flow Nodes Duration Page

Option	Description
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 average number of 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
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 6-2 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, in the **Queue Warning Size** field specify the number of incoming instances that can be waiting for an activity simultaneously.

d. In the Outgoing 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 Configuring Boundary Events.

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 6-1 shows the Approve Quote user task with different types of boundary events. Figure 6-2 shows the simulation configuration page for the Approve Quote user task.

Figure 6-1 The Approve Quote user task with multiple boundary events

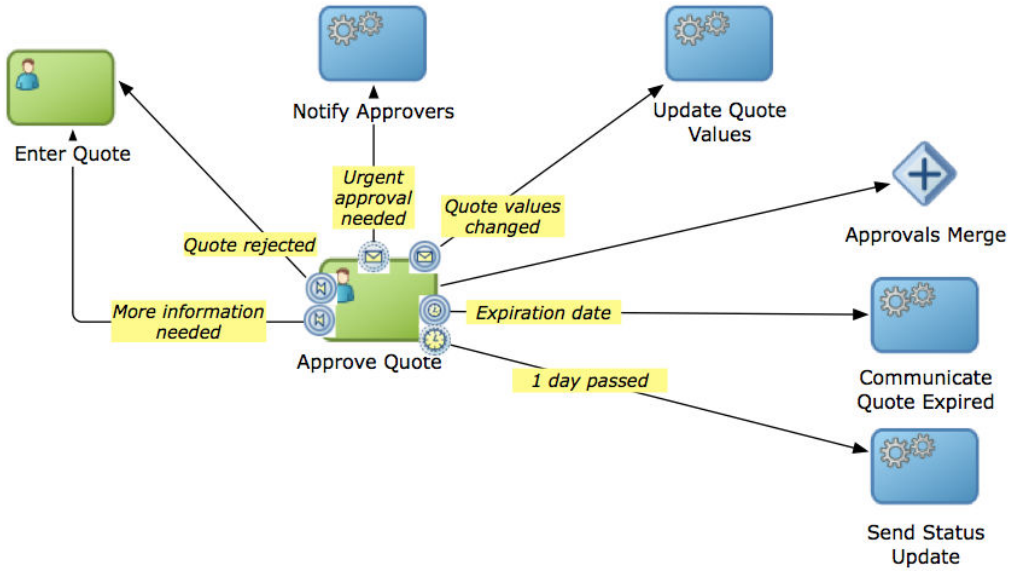
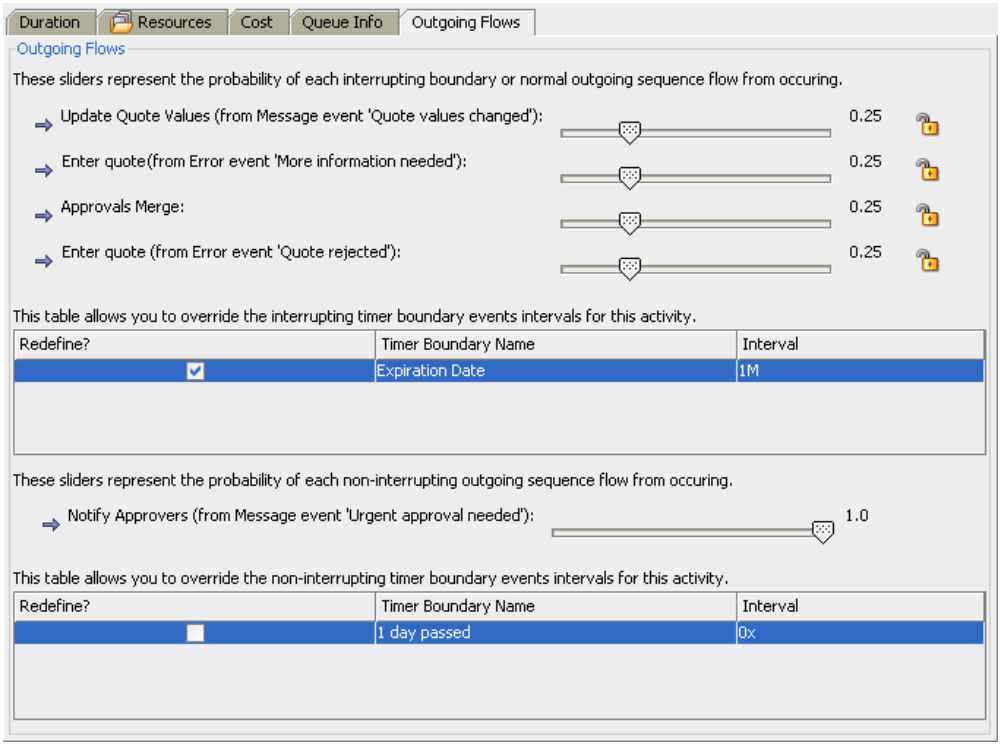


Figure 6-2 Outgoing Flows tab page for the Approve Quote task



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.

You can also create a simulation definition after you create a simulation model using the Simulation Wizard. For more information on how to create a simulation model and a simulation definition using the Simulation Wizard, see How to Create a Simulation Model from a Business Process.

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 simulationThe participant resources you want to include in the simulation

To create and configure a simulation definition:

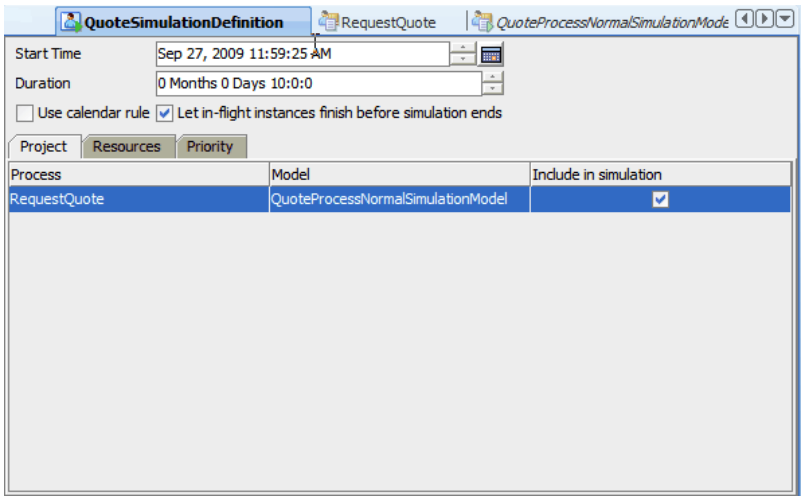
1. In the Applications window, expand the **Simulations** node.
2. Right-click the **Simulation Models** node.
3. Select **New** and then select From Gallery.

   The New Gallery dialog box appears.
4. In the Categories section, expand the BPM Tier node and select the Simulations node.
5. In the Items section, select BPM Project Simulation and click OK.

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

   The simulation definition appears under the Simulation Definitions node and the Simulation Definition editor opens. Figure 6-3 shows the Simulation Definition editor.

Figure 6-3 Simulation Definitions Project Page



8. Specify the general parameters for this simulation as described in Table 6-3.

Table 6-3 General Parameters for Simulation Definitions

Table 6-3 (Cont.) 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.

9. 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 6-4.

Table 6-4 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 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.

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

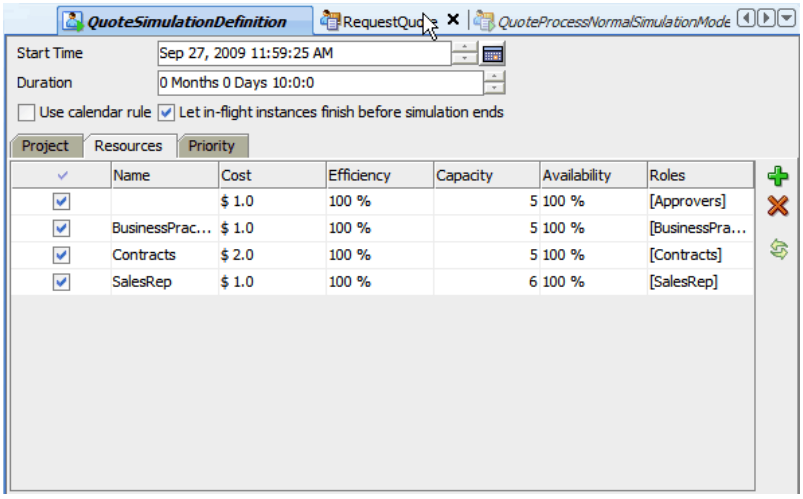
    All processes included in the simulation share these resources. The cost of each resource is defined per hour.

    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 6-4 shows an example of the Resources page.

Figure 6-4 Simulation Definitions Resources Page



11. 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.

Running Simulations

When you run a simulation, the animation appears in the project editor.

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

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 Applications window:
   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.

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.

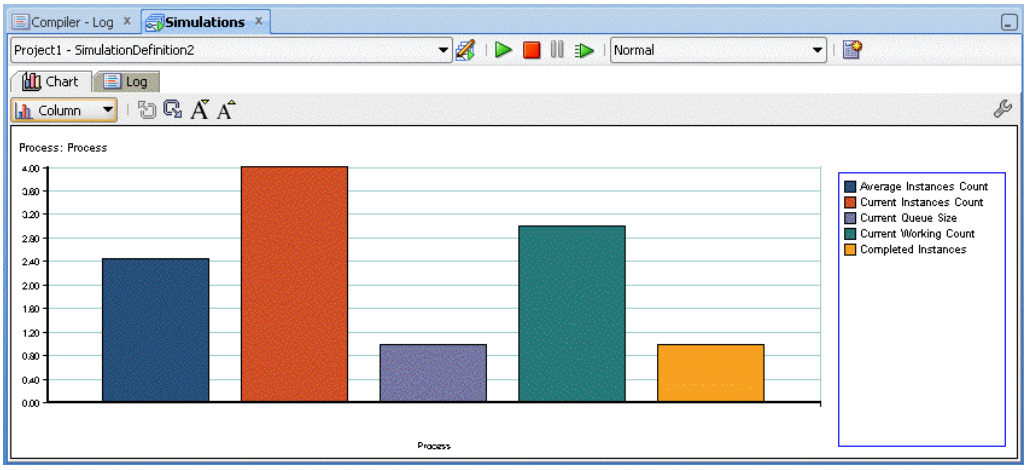
Note:

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

Understanding the Simulation View

The simulation view enables you to configure and run a simulation. Figure 6-5 shows a simulation view.

Figure 6-5 Example of a Simulations Page



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.

The toolbar on the Chart tab enables you to:

- Change the type of graphic to display the results
- Drill up and drill down
- Decrease and increase the font size

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

How to Analyze the Results of a Simulation Using a Chart

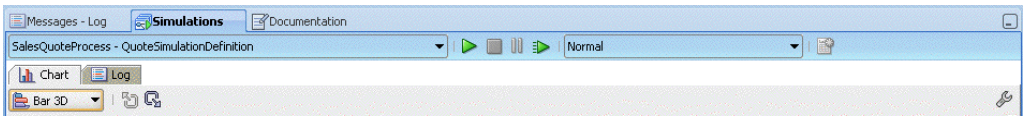
The Chart tab enables you to select a type of chart to display the result of the simulation. You can configure this chart to display the resources to monitor. You can also select the units the chart uses to measure the resources use.

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 6-6 shows the toolbar for a sample Chart page.

Figure 6-6 Simulations Chart Page



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.

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 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. Edit the report name in the **Name** column of the Report Parts table.
5. If you want to generate multiple reports, click the **Add** button to add a new report part.
6. In the Activities tab, select an option:

   - Summary
   - Details
7. 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.

8. In the indicators tab, select an option:
   - Summary
   - Details
9. 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.
10. Select the type of graphic to use in the report.

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

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 HTML 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.

7

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 these metrics to make decisions about your process.

This chapter includes the following sections:

- Introduction to Process Analytics
- Typical Process Analytics Workflow
- Configuring Projects_ Processes_ and Activities to Generate Sampling Points
- Adding Business Indicators to Projects
- Adding Measurement Marks to Processes
- Adding Counters to the Activities in a Process
- Defining Analytics View Identifier
- Configuring BAM 12c Process Metrics Generation in a Project
- Enabling Oracle BAM 11g in a Project

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 BAM 12c using Process Workspace dashboards or Oracle BAM 12c Process Analytics dashboards.

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 can be bound to project data objects. Once bound, the BPMN service engine publishes the business indicator values to process analytics stores when it runs the BPMN processes.

Process developers define the key performance indicators you want to monitor while developing your process. Business analysts will use the out-of-the-box process analytics dashboards as-is or customize them. Additionally, custom metrics defined by custom business indicators in a process are exposed in the BAM 12c process-specific data object at deployment. Business analysts can also create custom dashboards and views off these process-specific data objects.

Process Analytics track:

- Process and Activity Performance Metrics

- Case Metrics
- Human Task Metrics

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

Following are some of the measures and dimensions columns available in BAM 12c `PROCESS` and `ACTIVITY` logical data objects.

The supported pre-defined measures are:

- Number of days since the last process
- Process cycle time in days
- Number of days since the processes started
- Estimated completion time in days
- Estimated completion time in hours
- Process cycle time in hours
- Process open time in hours
- Process running tim in milliseconds
- Process suspend time in milliseconds
- COUNT of ALL, closed, open, open today, closed today, overdue instances, and so on.

The supported pre-defined dimensions are:

- Composite name
- Domain name
- Revision process name
- Process display name
- Fault type
- Process instance status
- Process start time

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 BAM 12c Process Metrics that you can use to store the Process Analytics data.You can also store the data to BAM 11g monitor express or use both systems simultaneously.

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.

- The BAM 12c Process data object tracks the time an instance takes to run that process from the start to the end event.
- The BAM 12c Activity data object tracks 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:

Information about active and completed activity, measurement intervals, marks, and counter instances are stored in the Activity data object.

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.

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.

Workload and performance metrics for the Process and Activity are based on the BAM 12c Process and Activity data objects, respectively.

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. Workload and performance metrics for the Participant are based on the BAM 12c HWF Assignment data object.

Typical Process Analytics Workflow

The typical process analytics workflow has several tasks, including configuring the process to use BAM 12c Process Metrics or BAM 11g monitor express, or both.

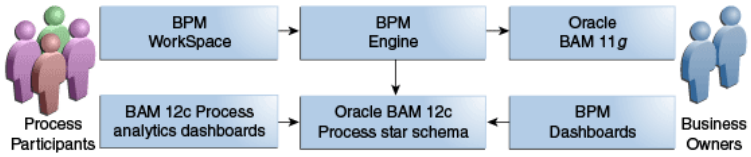
The following list describes the typical tasks you perform when you use Process Analytics in a BPM Project:

- 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 BAM 12c Process Metrics or BAM 11g monitor express, or both.
- Add Project data objects to your project.

- Assign values to the Project data object.
- Define Business indicators and bind them to the appropriate Project data objects.
- Add measurement marks or counter marks in the processes where you want to track the value of the business indicators.
- Deploy your project. See How to Enable Global Flags for Publishing Analytics for information about what you have to do before you can deploy a project.
- Use BAM 12c composer or BAM 11g Architect to configure custom dashboards.

Figure 7-1 shows the cycle the process analytics data goes through after deploying and running a BPMN process.

Figure 7-1 Process Analytics Data Cycle



How to Enable Global Flags for Publishing Analytics

Before you can deploy a project, make sure that the target BAM 12c or BAM 11g server is up and available, and that the global flags for analytics publish are enabled.

Note:

If you deploy a BPM 12c composite configured with a BAM12c analytics data target before you enable global analytics flags or when the BAM 12c server is down, then all data published to the BAM 12c Process star schema from that composite is permanently disabled. If the BAM 12c server went down, the disabling remains in effect even when the BAM 12c server comes back up.

The disabling happens because some mandatory artifacts required for enabling run time analytics population cannot be created in BAM 12c during composite deployment time. To allow these composites to publish data to BAM 12c Process star schema, the BPM 12c composites should be (re)deployed with the global analytics flag enabled with the BAM 12c server in running state.

Ensure that the Global Flags for Analytics Publish are Enabled

1. Log in to the Fusion Middleware Control console.
2. In the Target Navigation pane, expand the **Weblogic Domain** node.
3. Select the domain where the Oracle SOA 12c server is installed.

   For example, the domain might be `soainfra` or `base_domain`.
4. Right-click the domain and select **System MBean Browser**.

   The System MBean Browser displays.
5. In the System MBean Browser, expand the **Application Defined MBeans** node.

6. Under Application Defined MBeans, expand the **oracle.as.soainfra.config** node.
7. Under oracle.as.soainfra.config, expand the **Server: server_name** node.
8. Under Server: server_name, expand the **AnalyticsConfig** node.
9. Under AnalyticsConfig, click **analytics**.

   A list of the analytics attributes displays.
10. Ensure that **disableAnalytics** is set to **false**.
11. If BAM 12c is the analytics target where information needs to be published, ensure that **disableProcessMetrics** is set to **false**.
12. If BAM 11g is the analytics target where information needs to be published, ensure that **disableMonitorExpress** is set to **false**.

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 BAM 12c Process Metrics. This data comes from the sampling points defined in the project. You can configure which processes or activities in your project generate sampling points in these databases.

By default, Process, Activity, HWF task events, and CASE life cycle events are available for measurements. You measure any of these points by specifying override behavior a the Project, Process, and Activity levels. The measurements capture all dimensions, attributes, and measure for Process and Activity standard sampling. The default project-level setting for BPM projects is to generate Interactives and for SOA projects is to *not* generate Interactives. By default, the project is configured to generate sampling points only for interactive activities.

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.

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 interested 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.

User-Defined Measurements

You can define single, counter mark, and interval measurements.

A single measurement enables you to sample indicators at a specific point in the process. Single measurements are specified on the transition, capture only the measures specified in the definition, and capture all dimensions and attributes.

A counter mark measurement enables you to define a counter on an activity. Counter mark measures sample the counters specified in the definition and capture all dimensions and attributes.

An interval measurement enables you to define a logical activity with a start and an end point. Interval measurements capture only the measures specified in the definition and all dimensions and attributes.

Enable HWF and Case Measurements

To enable measurements for HWF, edit the Process Analytics Summary section of the BPM Project preferences and set the Project Sampling Points value to Generate for Interactive(s). For more information on how to configure the sampling point generation of a project, see How to Configure the Sampling Point Generation of a Project.

To enable case measurements, edit the Process Analytics Summary section of the BPM Project preferences and set the Project Sampling Points value to Generate for Case. Not that this option only appears when you define a case for your BPM project. For more information on how to configure the sampling point generation of a project, see How to Configure the Sampling Point Generation of a Project.

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. Right-click the project and select **Project Preferences**.
2. Click the **Process Analytics Summary** tab.
3. In the Process Sampling Points section, select an option:

Option	Description
Generate for Process(es)	Enable process events only.
Generate for Interactive(s)	Enable HWF task and assignment events, user task activity events, and process events.

Option	Description
Generate for case (BPM projects only)	Enable CASE events, HWF task and assignment events, user task activity events, and process events. This option is available only after you define a case for your BPM project.
Generate for All	Enable all events.
Do Not Generate	Disable all events.

4. Click **OK**.

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.

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**.

What Happens When You Configure the Sampling Points for an Activity

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

Adding Business Indicators to Projects

Business indicators are used to capture the key performance indicators for your process. Business indicators can be bound to project data objects.

You can use the business indicator binding or the project data object node in the structure pane to bind a business indicator to a project data object. 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
- Attribute

Note:

Use dimension business indicators for indicators with low cardinality. For example, customer type or order status. For business indicators with high cardinality such as identifications keys, whose value is unique for each instance, use Attribute business indicators.

The type of business indicator determines the available data types you can use. Table 7-1 shows the available data types for each business indicator type.

Table 7-1 Available Data Types for Business Indicator Types

Business Indicator Type	Allowed Data Types
Dimension	• string • boolean • time • int (with ranges) • double (with ranges) • decimal (with ranges)

Table 7-1 (Cont.) Available Data Types for Business Indicator Types

Business Indicator Type	Allowed Data Types
Measure	• int • double • decimal
Counter	• int
Attribute	• string • boolean • time • int • double • decimal

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 business indicator 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 Adding Counters to the Activities in a Process.

Attributes

Attribute business indicators enable you to capture high cardinality values that do not classify as dimension or measures, but enable you to filter information or to make reference to other information. Generally you use Attribute business indicators to store the value of identification keys such as the order ID or the service request ID.

How to Add a Business Indicator to a Project

The process of adding a business indicator to a project involves the following steps:

1. Create the Business Indicator
2. Bind Business Indicators to a Project Data Object

3. Create Data Associations for a Project Data Object.

Create the Business Indicator

Business indicators enable you to define the key performance indicators to measure in your project. You can use any of the following ways to add a business indicator to a project.

- Use the Business Indicator Editor
- Use the Project Structure pane
- Use a User-Defined Measurement Marks User Interface

Use the Business Indicator Editor

1. In the Application Navigator pane, double click **Business Indicators**.

   The Business Indicator screen displays with a list of the different types of business indicators. Available options are: Counters, Measures, Attributes, Dimension.

2. In the Business Indicator screen, click the Add (+) button for the type of business indicator that you want to add.

   The Create dialog box appears.

3. Enter a name to identify the business indicator.

   The maximum length for the name is 28 characters.

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

   Note:

   The available data types vary according to the type of business indicator you selected. Table 7-1 shows the available data types for each business indicator type.

5. If you selected a continuous data type and selected Dimension as its business indicator type, then add at least one range.

6. Click **OK**.

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

Use the Project Structure pane

1. In the Structure window, expand the business indicator bindings node and right-click the business indicator type.

   Available options are: Counters, Measures, Attributes, Dimension.

   The Bind business Indicator dialog box displays.

2. In the Bind Business Indicator dialog box, click the Add (+) button.

   The Create Business Indicator dialog box displays.

3. In the Create Business Indicator dialog box, enter a name to identify the business indicator.

   The maximum length for the name is 28 characters.

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

   Note:

   The available data types vary according to the type of business indicator you selected. Table 7-1 shows the available data types for each business indicator type.

5. If you selected a continuous data type and selected Dimension as its business indicator type, then add at least one range.

6. Click **OK**.

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

Use a User-Defined Measurement Marks User Interface

Be aware that only measure type business indicators can be created from the user-defined Measurement Mark Properties user interface.

1. In the user-defined Measurement Mark user interface, select and right click a user-defined measurement and click **Properties**.

   The Measure Mark Properties dialog box displays.

2. In the Measurement Mark Properties dialog box, select a Measurement Type and provide a Name.

3. Click the Add (+) button.

   The Bind Measure dialog box displays.

4. In the Bind Measure dialog, click the Add (+) button.

   The Create Measure dialog box displays.

5. In the Create dialog box, enter a name to identify the business indicator.

   The maximum length for the name is 28 characters.

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

   Note:

   The available data types vary according to the type of business indicator you selected. Table 7-1 shows the available data types for each business indicator type.

7. If you selected a continuous data type and selected Dimension as its business indicator type, then add at least one range.

8. Optionally, check **Auto Initialize** to initialize the business indicator with a default value.

   For more information about default values, see Default Values.

9. Click **OK**.

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

Bind Business Indicators to a Project Data Object

Once you create a business indicator, you must bind (map) the business indicator to a project data object. At run time, BPM determines the value of a business indicator from its associated data object. Use the Project Structure pane in either of the following nodes to bind a business indicator to a project data object.

- Business Indicator Bindings Node
- Project Data Objects Node

Business Indicator Bindings Node

1. In the Business Indicator Bindings node, right click the business indicator type and click **New**.

   The Bind Business Indicator dialog box displays.

2. In the Bind Business Indicator dialog box, select or create a business indicator and bind it to a new project data object.

   You can also change the name of new project data object to any other unique new name, if needed.

3. Optional. Check Auto Initialize to initialize the business indicator with a default value.

   See Default Values for more information about default values.

4. Click OK.

   The business indicator binding and a new project data object are added.

Project Data Objects Node

1. In the Project Data Objects node, right click the data object, select **Bind to Business Indicator**, and select a business indicator.

   The Bind Business Indicator dialog box displays.

2. In the Bind Business Indicator dialog box, select or create a business indicator.

3. Optional. Check Auto Initialize to initialize the business indicator with a default value.

   See Default Values for more information about default values.

4. Click **Ok**.

   The business indicator binding is added.

Create Data Associations for a Project Data Object.

The next step is to define data associations of the project data object. The business indicators bound to the project data object get their run time values from the project data objects.

You perform BPM process data association on the Activity. To launch the Data Associations user interface, do the following:

1. Right-click an activity.
2. Select **Properties**.
3. Select the **Implementation** tab.
4. Click **Data Associations**.

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 dimension business indicators data at pre-defined and custom sampling points defined for your process to process analytics databases. The dimension business objects are written to the BAM Process Star schema data objects.

- **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 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 indicator. For information on how to add a counter mark, see Adding Counters to the Activities in a Process.

- **Attributes**

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

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:

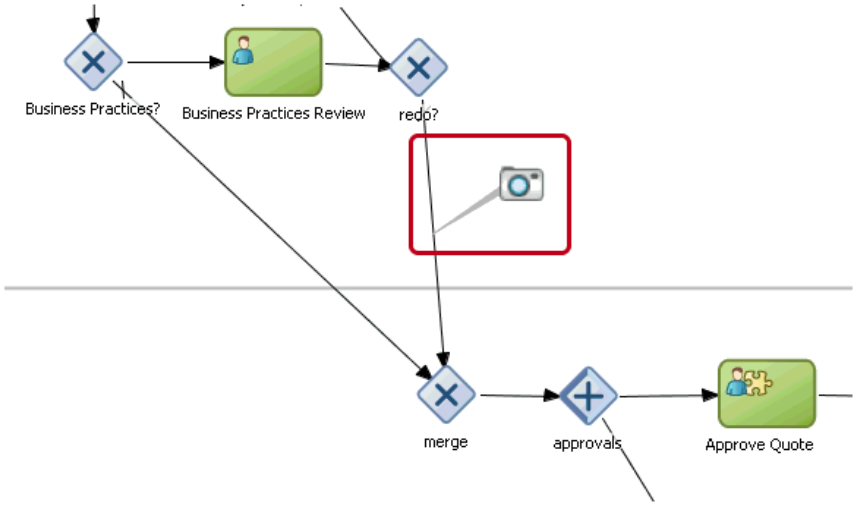
- Single Measurement
- Interval Start
- Interval Stop

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 7-2 Single Measurement Mark

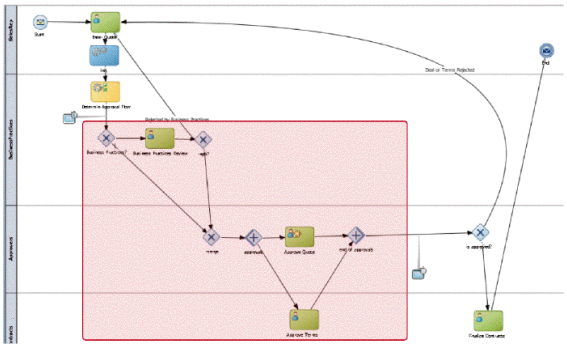


Interval Start and Interval Stop

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 7-3 Interval Start and Interval Stop Measurement Marks



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 7-4 show the Measurement Mark Properties dialog box.

Note:

You can measure multiple business indicators in the same measurement mark.

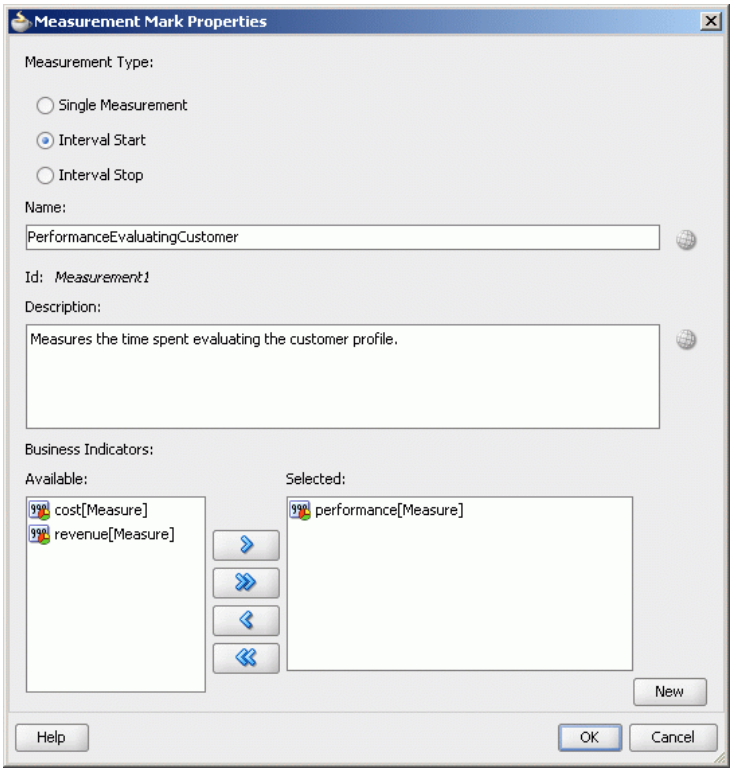
Note:

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 box, then you can click the **New** button under the **Selected** list.

---

8. Click **OK**.

Figure 7-4 Measurement Mark Properties Dialog



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
- Attributes defined for the process

Note:

Information about active and completed activity, measurement intervals, marks, and counter instances are stored in the Activity data object.

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 box, 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**.

What Happens When You Use Measurement Marks to Measure Business Indicator Values for a Section of a Process

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, Attributes, 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.

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 publishes an event containing the value of its associated counters to Process Analytics databases.

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.

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 Adding Business Indicators to Projects.

2. Open the BPMN process.
3. Right-click the activity to which you want to add the counter and click **Add counter mark**.
4. In the Available list, select the counter business indicator that you want to record as part of the counter mark.

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

5. Click **OK**.

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 publishes an event that contains the value of its associated counters to the Process Analytics databases.

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**.

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 publish event with counter business indicator values to the Process Analytics databases.

Defining Analytics View Identifier

Analytics view identifier identifies the process analytics view that provides data across all the existing versions of a composite.

This identifier must be unique across all deployed processes. If you use the same identifier for more than one process, then the process deployment fails.You create an Analytics View identifier so that you can query the view with Oracle SQL.

The purpose of creating an Analytics View identifier is so you can have Oracle SQL access to Process Metrics by querying that view. See Process Star Schema Views for format information.

How to Define the Analytics View Identifier

You can define an analytics view identifier to use across all the existing versions of a process.

To define the analytics view identifier:

1. In Application Navigator, right-click the Project.
2. Select **BPM > Preferences**.
3. Enter a view identifier in the **Analytics View Identifier** field.

Configuring BAM 12*c* Process Metrics Generation in a Project

Oracle Business Activity Monitoring (BAM) 12*c* provides a set of predefined Process star schema and dashboards that you can use to monitor the activity of your processes.

At the composite level, the BPM composite publishes to BAM 12*c* by default. At the system (global) level, population to BAM 12*c* is disabled by default, and this default setting overrides the composite-level setting. To enable the BAM 12*c* process star schema data population at the system level, set the `DisableProcessMetrics` parameter of the `AnalyticsConfig` Mbean to `false`.

You can view the data stored in BAM 12*c* Process star schema data objects with the out-of-the-box BAM 12*c* Process analytics dashboards.

BAM 12c Process Metrics

BAM 12c Process Metrics 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.

BAM 12c Process Metrics 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

BAM 12c Process Metrics store the data related to activities in the `ACTIVITY` data object. It stores the data related to processes in the `PROCESS` data object. It also stores the data related to processes in the Process performance (`BPM_CUBE_PROCESSPERFORMANCE`) and Workload(`BPM_CUBE_WORKLOAD`) tables. It stores the data related to participants in HWF ASSIGNMENT data object

`PROCESS` and `ACTIVITY` data objects contain data about both completed and inflight process and activities respectively. The Task and Process performance tables contain data about completed activities and processes respectively. Similarly, `TASK` and

`ASSIGNMENT` data object contains data about both completed and in-progress task assignments to participants.

Data for activity and process data objects is computed and persisted when an activity or process is started or completed.

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.

How to Configure BAM 12c Process Metrics Generation in a Project

If you use BAM 12c Process Metrics to monitor the performance of your project, then you must enable the generation of the process metrics at development time. When you deploy your application, BPM 12c uses this configuration to enable BAM 12c Process Metrics.

To enable BAM 12c Process Metrics in a Project:

1. In the Application Navigator, right click the project.
2. Select **BPM > 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 12c** to enable BAM 12c Process Metrics generation, or deselect it otherwise.
6. Click **OK**.

What Happens When You Enable BAM 12c Process Metrics in a Project

The BPMN Service Engine publishes analytics information to the pre-defined BAM 12c process petrics each time it runs an activity or completes a process using the sampling point configuration you have defined. If you configure a process not to generate sampling points, then the BPMN Service Engine does not publish this information.

Note:

In order to work together, Oracle BAM and Oracle BPM must reside in the same WebLogic domain.

When a BPM project that is configured with measurement sampling points and business indicators is deployed, the following actions are performed in the BAM 12c process star schema:

1. Composite-specific fact data objects are created or updated with columns for each business indicator defined in the deployed composite. Additionally, if an analytics view identifier has been specified for the composite, the database view synonyms are generated for composite-specific physical data objects. See Process Star Schema Views for information about the names of composite-specific physical data objects and the database views that are created.
2. Standard Dimensions data objects, such as `COMPOSITE_DEFINITION`, `ACTIVITY_DEFINITION`, `PROCESS_DEFINITION`, `TASK_DEFINITION`, and

`ROLE_DEFINITION` are populated with the appropriate metadata information from the composite.

Enabling Oracle BAM 11g in a Project

You can use Oracle BAM 11g to monitor the activity of the process in your project, leveraging the capabilities of Oracle BAM 11g while using Oracle BPM. You can use Oracle BAM 11g with predefined Oracle BAM 12c Process metrics, or you can choose to disable the latter.

For more information about Oracle BAM, see *Part X Using Oracle Business Activity Monitoring* in *Developing SOA Applications with Oracle SOA Suite*.

How to Enable Oracle BAM 11g in a Project

Before enabling Oracle BAM 11g in your project, you must configure Oracle BAM 11g correctly. See chapter "Configuring Oracle BPMN Process Service Components and Engines" in *Administering Oracle SOA Suite and Oracle Business Process Management Suite*, for information on how to configure Oracle BAM 11g to work with Oracle BPM.

When you deploy a BPM project, Oracle BAM 11g automatically creates the custom and predefined BAM data objects for that BPM project.

To enable Oracle BAM 11g in a project:

1. In the Application Navigator, right-click the project.
2. Select **BPM > 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 11g**.
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**.

What Happens When You Enable Oracle BAM 11g

When you run a process that has Oracle BAM 11g enabled, the BPMN Service Engine populates Oracle BAM 11g Monitor Express data objects 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.

If you enable the data target for BAM 11g, then the BAM 11g data objects are automatically created when you deploy the process. To use Monitor Express dashboards you must import the automatically created data objects.

When you deploy the first process a custom business identifier and all the common data objects are created in the recommended/Samples/Monitor Express folder. In the next deployments the common data objects are shared by all process and only a new custom business identifier is created.

For more information, see *Developing SOA Applications with Oracle SOA Suite* and *Monitoring Business Activity with Oracle BAM*.

Part IV

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:

- Using the Business Catalog
- Sharing BPM Projects Using the Process Asset Manager
- Modeling Business Objects
- Working with Human Tasks
- Working with Services and References
- Using Business Rules
- Sending Notifications
- Using SOA Composites with BPM Projects

8

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. It also describes how the artifacts in your BPM project are represented in the business catalog.

This chapter includes the following sections:

- Introduction to the Business Catalog
- Adding a New Module
- Deleting a Module
- Customizing Synthesized Types
- Creating an Enumeration

Introduction to the Business Catalog

The business catalog is a directory within a BPM project that stores the components you use to implement some flow objects in BPMN processes. You can share and reuse the assets in the business catalog across the processes in a BPM project.

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
- Enumerations

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 8-1 shows which flow objects use each of the components in the business catalog for their implementation.

Table 8-1 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

Table 8-1 (Cont.) Flow Object Implementation

Component	Flow Objects
Enumerations	You can use the attribute of an enumeration 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

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
- Enumerations

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

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 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 Working with 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 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 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 Introduction to Oracle Mediator in Oracle BPM and 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 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 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 Modeling Business Objects for more information.
- **Enumerations**: You can add enumerations in any custom module in the Business Catalog. See Creating an Enumeration for more information on how to create enumerations.

Note:

If you add a component that references resources that are missing or corrupted, the business catalog shows this missing dependencies with an error node. The label of the node displays the path of the resource. For more information on the missing resource place the mouse over the node to display the tooltip, or build the BPM project. Building the BPM project enables you to find the component that requires the missing resources.

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.

Predefined Modules

The business catalog contains the following predefined modules:

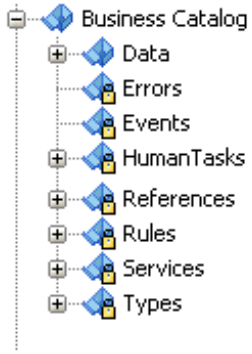
- **Errors**: Stores the errors that the operations in the services and references in your project define.
- **Events**: Stores the Events that you add to the SOA Composite.
- **HumanTasks**: Stores the Human Tasks that you add to your project.
- **References**: Stores the interfaces you can use to define 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 8-1 Custom and Predefined Modules



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.

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 Applications window, right-click the **Components** node or a user-defined existing module.
2. Select **New**.
3. Select **Module**.
4. Enter a name to identify the module.
5. Click **OK**.

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

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.

How to Delete a Module

You can delete those modules that you do not use or need.

To delete a module:

1. In the **Applications window**, right-click the module you want to remove.
2. Select **Delete**.

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

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.

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.

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 Applications window, expand the **Types** predefined module.
2. Right-click the type you want to customize.
3. Select **Customize Type**.

   The Create Business Object dialog box 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**.

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.

Creating an Enumeration

You can use enumerations to model a collection of values of the same type. Enumerations enable you to check and restrict the values of a certain argument in compile time, avoiding bugs and errors in runtime.

When you type the name of the enumeration the Simple Expression Editor autocomplete shows you the list of available attributes and their value. From this list you can select the attribute you want to use.

How to Create an Enumeration

You can create an enumeration to model a collection of values.

To create an enumeration:

1. In the **Project Navigator Tree**, right click the **Business Catalog** node.
2. Select **New**.
3. Select **Enumeration**.

   The Create Enumeration dialog box appears.
4. Enter a name to identify the enumeration.
5. Click the **Search** button next to the Destination Module field to select the module where to store the enumeration.

   To create a new module you can type the name in the **Destination Module** field or create it from the **Search** dialog box.
6. From the **Type** list, select the type of the enumeration.

   Available types are:

   - string
   - int
   - Double
   - Decimal
   - Boolean

   You can only add one enumeration of each type.

7. Click **OK**.

   The enumeration editor appears.

8. Add attributes to the enumeration.

   For more information, see How to Add Attributes to an Enumeration.

How to Add Attributes to an Enumeration

After you create the enumeration you must add the attributes that make part of the collection you are modeling, and assign them values.

To add an attribute to an Enumeration:

1. Edit the enumeration.
2. Expand the **Items** module in the **Enumeration** editor.
3. Click the **Add** button.

   The Create Item dialog box appears.

4. Enter a name to identify the attribute.

   The name is unique for each attribute and must not be a Java reserved words, start with a number, contains spaces or special characters.

5. Enter a value for the attribute.
6. Click **OK**.

   The new attribute appears in the Enumeration Editor.

7. Optionally, click the **Edit** button next to the **Documentation** label to add documentation that describes the attribute.

Using an Enumeration in a Simple Expression

The following sample code shows you how to use an enumeration in a simple expression. The enumeration in the example is a collection of colors. The code in the example is accessing the value of the red attribute, a string that represents the color value in the hexadecimal RGB format:

Example 8-1 Enumeration Use in a Simple Expression

```
Module.Color.red
```

9

Sharing BPM Projects Using the Process Asset Manager

This chapter describes how to store business assets in a process asset manager. The process asset manager runs in an application server that you configure.

This chapter includes the following sections:

- Introduction to the Process Asset Manager
- Working with BPM Projects Stored in the Process Asset Manager
- Working with the Process Asset Manager

Introduction to the Process Asset Manager

The process asset manager provides a uniform way to manage the different business assets that are part of a BPM project and Oracle Business Process Architect projects.

You can use the Process Asset Manager to share BPM projects between the different persons working on the project. Process developers working with Oracle BPM Studio can share their BPM projects with other process developers, or with process analysts using Oracle Business Process Composer. For more information about the development life cycle, see Overview of the Application Development Life Cycle.

After you checkout a project from the process asset manager, the JDeveloper SVN features are enabled. You can choose between using the process asset manager versioning functionality of the SVN. For example, you can publish a project using the process asset manager save action, or the SVN commit action. A project that you checked out from the process asset manager is an SVN project for JDeveloper.

The process asset manager supports the following:

- Collaboration

  It enables multiple users to work on the same project simultaneously.

- Security and Access Control

  It provides fine grained security and access control of the business assets. The application accessing the catalog determines the correct access rights granted to the principal of the application.

- Versioning

  It stores multiple versions of the same business asset.

- Conflict Resolution, Diff and Merge

  When a business asset is modified simultaneously by different users, the process asset manager enables you view the differences between the different versions, resolve the conflicts between them and merge the changes.

- Life Cycle

  It supports a flexible life cycle model that enables a business asset to mature from initial brainstorming to development and testing, to integration test and finally deployment in production.

- Reporting

  It provides a detailed reporting of the business assets in the catalog and their history.

- Backup and Recovery

  In the event of hardware failure, software bugs, and human error, you can revert the changes to a stable version of the project.

Working with BPM Projects Stored in the Process Asset Manager

This section covers the typical use cases of the Process Asset Manager, including setting up an environment and adding and exporting BPM projects.

- Setting up an environment to work with the Process Asset Manager

  See How to Set Up an Environment to Work with Projects Stored in the Process Asset Manager.

- Working with BPM projects stored in the Process Asset Manager

  See How to Modify a BPM Project Stored in the Process Asset Manager.

- Adding a BPM project to the Process Asset Manager

  How to add a BPM Project to the Process Asset Manager

- Exporting a BPM project from the Process Asset Manager to a zip file

  How to Export a BPM Project Stored in the Business Process Manager

How to Set Up an Environment to Work with Projects Stored in the Process Asset Manager

Before you start working with a BPM project stored in the Process Asset Manager, you must set up your environment by configuring a connection and checking out the BPM project.

To set up your environment:

1. Create a Process Asset Manager connection.

   See How to Create a Process Asset Manager Connection.

2. Checkout the BPM project from the Process Asset Manager.

   See How to Check Out a BPM Project from the Process Asset Manager

3. Modify the BPM project.

4. Save the changes to the Process Asset Manager.

   See How to Save a BPM Project to the Process Asset Manager.

How to Modify a BPM Project Stored in the Process Asset Manager

You can modify a BPM project that you checked out from the Process Asset Manager and then save the changes to the Process Asset Manager, to share them with other developers.

To modify a BPM project stored in the Process Asset Manager:

1. Update your local copy with the changes from the Process Asset Manager.

   See How to Update Local BPM Projects

2. Modify the BPM project.
3. Save the changes to the Process Asset Manager.

   See How to Save a BPM Project to the Process Asset Manager.

How to add a BPM Project to the Process Asset Manager

You can add a BPM project that you created and stored locally, to the Process Asset Manager.

To add a BPM project to the Process Asset Manager

1. If you do not have a Process Asset Manager connection, create one.

   See How to Create a Process Asset Manager Connection.

2. Save the BPM project to the Process Asset Manager.

   See How to Save a BPM Project to the Process Asset Manager

How to Export a BPM Project Stored in the Business Process Manager

You can export a BPM project in the process asset manager to a zip file that you can store locally.

To export a BPM project:

1. Open the **Process Asset Manager Navigator**.
2. Right-click the *BPM project* that you want to export.
3. Select **Export**.

   The Export Project from PAM wizard appears.

4. Click **Next**.

   If you want to avoid viewing the welcome page the next time you export a project, select **Skip This Page Next Time**.

5. Enter a file name to identify the exported file.
6. Click the **Search** button next to the **Location** field, to select a destination where to store the exported file.

7. Click **Next**.

   The Included Files page appears. This page shows all the files that are included in the export file.

8. Click **Next**.

   The Finished page appears. This page shows if the export operation was successful.

9. Click **Finish**.

Working with the Process Asset Manager

The Process Asset Manager supports creating a connection, checking out and saving a BPM project, updating a BPM project locally, removing projects, and viewing the change history.

- Creating a connection

  See How to Create a Process Asset Manager Connection

- Checking out a BPM project

  See How to Check Out a BPM Project from the Process Asset Manager

- Saving a BPM project

  See How to Save a BPM Project to the Process Asset Manager

- Updating locally stored BPM projects

  See How to Update Local BPM Projects

- Removing BPM projects

  See How to Delete a BPM Project from the Process Asset Manager

- Viewing the change history

  See How to View the Change History

How to Create a Process Asset Manager Connection

Before you start to work with a process asset manager, you must configure a process asset manager connection to locate the server where the process asset manager is stored.

To create a process asset manager connection:

1. Open the **Process Asset Manager Navigator** window:

   a. From the JDeveloper menu, select Window.

   b. Select Process Asset Manager Navigator.

   The Process Asset Manager Navigator opens on the left of the screen.

2. Right-click the PAM node, and select Create Connection.

   The **Create Process Asset Manager Connection** wizard appears.

3. Click **Next**.

The **Connection Name** page appears.

If you want to avoid viewing the welcome page the next time you export a project, select **Skip This Page Next Time**.

4. Enter a name to identify the connection.
5. Enter the user name to log in to the process asset manager server.
6. Enter the password to log in to the process asset manager server.
7. Click Next.
8. In the WebLogic Hostname field, enter the URL to locate the process asset manager server.
9. Optionally, configure the port, SSL port and the Always Use SSL option
10. Click **Test Connection** to test if the provided information is correct.

    The Status field shows if the connection to the process asset manager server is successful.

11. Click **Finish**.

    The new connection appears in the Process Asset Manager Navigator.

How to Check Out a BPM Project from the Process Asset Manager

You can checkout a specific version of a BPM project from the process asset manager and store it locally. You can make local modifications to this local project and then check them in to the process asset manager.

To check out a BPM Project from the process asset manager:

1. In the **Process Asset Manager Navigator**, expand the connection node and the expand the space node.
2. Right-click the BPM project to check out.
3. Select **Check Out**.

   The Checkout Project from PAM dialog box appears.

4. Click the **Search** button next to the **Destination** field to select a local destination where to store the checked out project.
5. Click **OK**.

   The Checking Out Project dialog box appears while the Process Asset Manager checks out the project.

   The BPM project that you selected is stored to a local destination and appears in the Applications window. Note that the BPM project checked out from the Process Asset Manager shows an icon with the status of the project, to view the status place the cursor over the node and wait for the tool tip to appear.

How to Save a BPM Project to the Process Asset Manager

You can store a BPM project in the process asset manager to share it with other users and work together on the same project. You can use this procedure to store a new

BPM project or to save the changes you made to a BPM project already stored in the Process Asset Manager.

To save a BPM project to the process asset manager:

1. Open the BPM project you want to store in the process asset manager.
2. In the **Applications** window, right click the *BPM project.*
3. Select **Save to PAM**.

   You might need to do an update and resolve any possible conflicts before you can save your changes.

   The Select Connection dialog box appears.
4. From the PAM Connections list, select a connection, or click the Add button to create a new connection.
5. From the Space list, select a space, or click the Add button to create a new space.
6. Click OK.

   The Save Project to PAM dialog box appears. This dialog box displays the source of the BPM project, and the connection and space where to store the BPM project.
7. Enter a comment.
8. Click **OK**.

   The BPM project appears in the Process Asset Manager and is available to other users using the Process Asset Manager.

Note:

If the project is locked from Business Process Composer you cannot save your changes. To save your changes, first release the lock from Business Process Composer.

How to Update Local BPM Projects

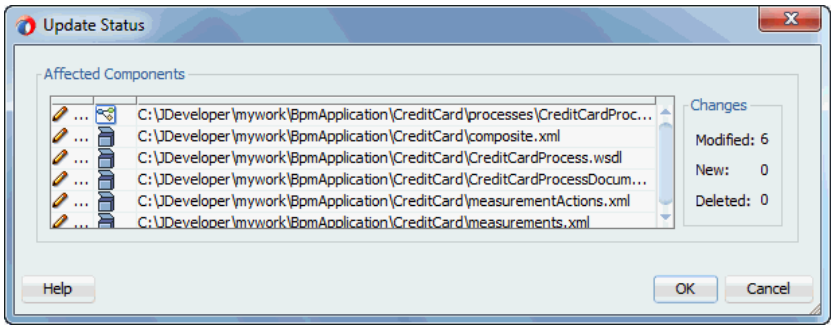
You can update your local BPM project with the changes made to the BPM Project in the Process Asset Manager.

To update local BPM projects:

1. In the Applications window, right-click the *BPM project* that you want to refresh.
2. Select Update. This item is located under the Save to PAM item.
3. If there are changes in the BPM Project stored in the Process Asset Manager, the Update Status dialog box appears. Click OK to accept the changes.

   The Updating Local Source Dialog appears while the Process Asset Manager updates the local BPM Project.

Figure 9-1 Update Status Dialog



How to Delete a BPM Project from the Process Asset Manager

You can delete a project from the Process Asset Manager.

To delete a BPM Project from the process asset manager:

1. Open the **Process Asset Manager Navigator**.
2. Right-click the *BPM project* that you want to delete.
3. Select **Delete**.

   The Delete Project from BAC dialog box appears.
4. Click Yes.

   The Deleting Project dialog box appears while the Process Asset Manager deletes the project.

How to View the Change History

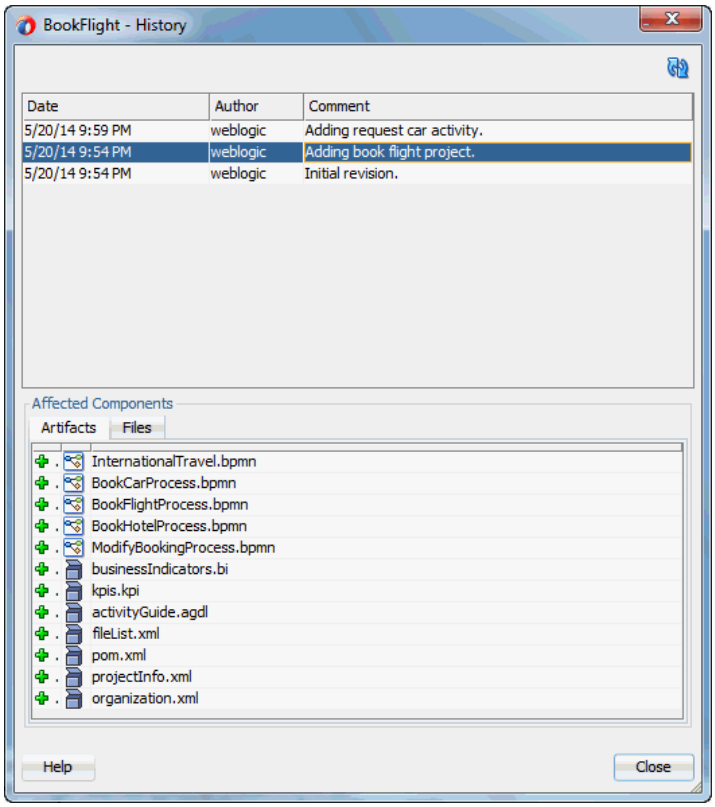
You can view the history of the changes made to a BPM project since it was added to the Process Asset Manager, and you can browse the details of each of those changes.

To view the change history:

1. Open the **Process Asset Manager Navigator**.
2. Right click the *BPM project* from which you want to see the change history.
3. Select History and Changes.

   The History dialog box appears. The history table shows the date, author and comments for each of the changes in the history of the BPM project. When you select a change the Affected Components table shows the modified artifacts and files, depending on which tab you select. Figure 9-2 shows the History dialog box for the BookFlight project.

Figure 9-2 History Dialog



10

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:

- Introduction to Business Objects
- Working with Business Objects
- Using a Business Object in a Process
- Adding Business Objects Based on a XML Schema Element or Type
- Introduction to Business Object Attributes
- Working with Business Object Attributes
- Working with Business Object Methods
- Sharing Business Objects
- Introduction to Business Object Inheritance
- Working with Business Object Inheritance

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 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.

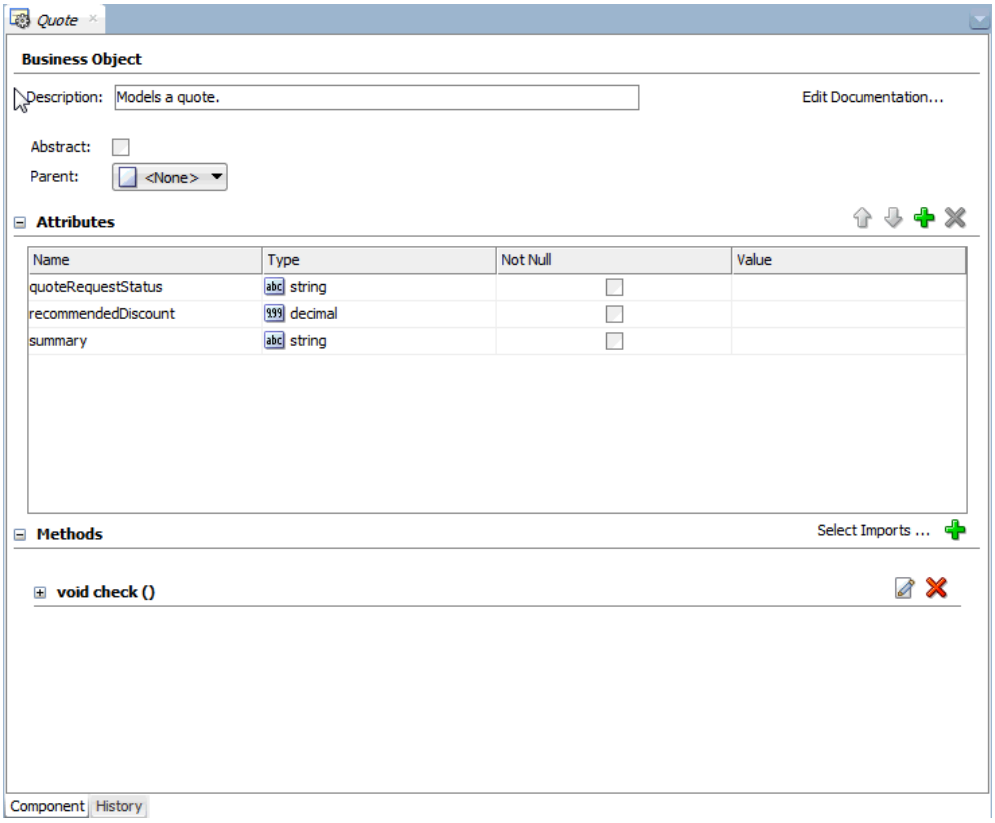
Business objects support inheritance enabling you to reuse the data and behavior they define. For more information about inheritance, see Introduction to Business Object Inheritance and Working with Business Object Inheritance.

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 10-1 shows the Business Object Editor editing a Quote object created manually.

Figure 10-1 Business Object Editor



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 module**: 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.

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 objects 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. You can also reuse the data and behavior defined in a business object within the same project, using business object inheritance. For more information about inheritance, see Introduction to Business Object Inheritance and Working with Business Object Inheritance.
- **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.

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:

Oracle BPM Studio forces the first letter of the name of a business object to uppercase.

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 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.

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**.

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

How to Modify a Business Object

You can modify an existing business object by:

- Adding attributes

  See How to Add a Business Object Attribute
- Removing attributes

  See How to Remove a Business Object Attribute
- Adding methods

  See How to Add a Business Object Method
- Removing methods

  See How to Remove a Business Object Method
- Adding documentation

  See How to Document a Business Object Attribute

  See How to Document a Business Object Method

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 Applications window, right-click the business object you want to delete.
2. Select **Delete**.

A confirmation message appears.

3. Click **OK**.

What Happens When You Delete a Business Object

Oracle BPM Studio removes 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.

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 Introduction to the Documentation Editor, for details on how to create and edit documentation.
4. Click **Close**.

What Happens When You Document a Business Object

The documentation is available for other process developers to read and modify.

Using a Business Object in a Process

Business objects store data related to your process. You can update the information in this data object from any of the activities in the 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.

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 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.

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.

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.

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 object.

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 How to add an XML Schema to Your BPM Project.
6. Select the external type from which to create the new business object.

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.

How to add an XML Schema to Your BPM Project

From the Create Business Object dialog box 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, select the Based on External Schema option.
2. Click the **Schema Browser** button.

   The Type Chooser dialog box appears.
3. In the upper right corner, click the **Import Schema Files** button.

   The Import Schema File dialog box appears.
4. Click the **Browse Resources** button next to the URL field.

   The SOA Resource Browser appears.
5. Browse your file system and select a schema file.
6. Select **Copy to Project**.
7. Click **OK**.

   If the XML schema contains references to other types a dialog box to confirm their import appears.

   The Browse Resources dialog box closes and the Type Chooser dialog box appears.
8. Select an element to use as the base of your business object.

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.

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 How to Document a Business Object Attribute.
- Custom default value
- Not null constraint.

Supported Data Types for Business Object Attributes

The following table describes the supported data types for an attribute in a business object:

Table 10-1 Supported Data Types

Data Type	Description
string	Alphanumeric values
int	Integer numbers
boolean	True or false values
double	Double numbers
decimal	Decimal numbers with defined precision
dateTime	Unit of time, stores the date and the time
long	Long numbers
duration	Intervals of time
base64Binary	Binary values (For example: images, files)
float	Float numbers

Table 10-1 (Cont.) Supported Data Types

Data Type	Description
byte	An 8-bit signed two's complement integer
short	A 16-bit signed two's complement integer
date	Unit of time, stores the date only
time	Unit of time, stores the time only
Array	A collection of elements of a specified data type
Complex Types	Other business objects

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.

Working with Business Object Attributes

To model a business object you must add its attributes. These attributes store the data related to your process.

You can add, modify and delete attributes as necessary. You can also add them to documentation that describes the data they store and provides any necessary information to the user of the business object.

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 Applications window 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.

Note:

To use the array type, click the Browse More Types button, select the type of the array and select the Array check box.

5. Click **OK**.

How to Remove a Business Object Attribute

You can delete a business object that is no longer useful to your project.

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, select the attribute you want to remove.
3. Click the Remove button at the top of the table.

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

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 Documentation** button next to the **Description** field.

   The Documentation Dialog appears.
4. Add the text to document the functionality of the selected attribute.

   See Introduction to the Documentation Editor, for details on how to create and edit documentation.

What Happens When You Document a Business Object Attribute

The documentation is available for other process developers to read and modify

Working with Business Object Methods

A business object can contain methods that manipulate and perform operations with the data it contains. These methods represent the behavior of the business object.

You can add, modify and delete attributes as necessary. You can also add them todocumentation that describes the operations they perform and provides any necessary information to the user of the business object.

How to Add a Business Object Method

You can define business objects methods to manipulate and perform operations based on the data of the business object.

To add a business object method:

1. Open the business object.
2. In the Methods section, click the Add Method button.

   The Method dialog box appears.
3. Enter a name to identify the method.
4. Click OK.

   The new method appears in the Methods section.
5. Expand the method and add the script code in the text area that appears.

   For more information on BPM Scripting, see Writing BPM Scripts.
6. Optionally you can change the signature, see How to Change the Signature of Business Object Method.

How to Change the Signature of Business Object Method

You can change the signature of a business object method by changing the name or the return type, or by modifying the input arguments.

To change the signature of a business object method:

1. Open the business object that contains the method in the Business Object editor.
2. Expand the Methods section.
3. Expand the method that you want to modify.
4. Click the Change Signature button located next to the Remove button.

   The Change Signature dialog box appears.
5. You can edit the following elements of the method signature:

- the name of the method
- the type of the return parameter
- add, remove or modify input arguments

How to Remove a Business Object Method

You can remove a business object method that you no longer use.

To remove a business object method:

1. Open the business object that contains the method you want to remove, in the Business Object editor.
2. Expand the Methods section.
3. Click the Remove button for the method you want to remove.

How to Document a Business Object Method

You can add documentation to a business object method for other process developers to understand its functionality.

To document a business object Attribute:

1. Edit the business object that contains the method you want document.
2. In the Methods section, expand the business object method you want to document.
3. Click the **Edit Documentation** button next to the **Description** field.

   The Documentation Dialog appears.
4. Add the text to document the functionality of the selected method.

   See Introduction to the Documentation Editor, for details on how to create and edit documentation.

Sharing Business Objects

You can share business objects between different projects by exporting them to a file and then importing them.

You can choose to export a single business objects or multiple business objects. When exporting multiple business objects you can also export exceptions. The file that contains the exported business object has the extension .bob. If the business object depends on other business objects, then those dependencies are also included in the export file.

You can import the business objects from the export file in any other project. When you import a business object, Studio also imports the module where it was stored if the module does not exist already.

How to Export a Business Object

You can export a business object to share it with other developers.

To export a business object:

1. In the **Applications window**, right-click the business object.
2. Select **Export**.

   The Select Object File dialog box appears.
3. Select a directory where to store the exported business object.
4. In the **File Name** field, enter a name for the exported business object.
5. Click **Save**.

   The exported business object file is stored to the selected directory.

How to Import Business Objects from a File

You can import a business object that was exported from another project.

To import business objects from a file:

1. In the **Applications window**, right-click the **Business Catalog** node.
2. Select **Import Business Objects**.

   The **Select Object File** dialog box appears.
3. Select the file that contains the exported business objects.
4. Click **Open**.

   The business objects contained in the selected file appear in the business catalog.

Introduction to Business Object Inheritance

Inheritance enables you to reuse the data and behavior defined in a business object. This avoids duplicating data structures and scripts, making maintenance easier and less error-prone.

To use inheritance you create a business object as the child of another business object that defines the attributes and methods that you want to reuse. The latter is called the parent object.

The child object has access to the attributes and methods of the parent objects as if it defined them.

In a business process that manages the payroll of a company, you can define a business object Employee and then a child business object Manager that inherits the data and methods of Employee and re-defines some of them.

Method Overloading

The child business object can re-define the methods defined in the parent object, assigning them a different behavior specific to that child business object. The input arguments and return parameters of the re-defined method must match the ones in the method from the parent object.

Polymorphism

The child business object is a sub-type of the parent business object. This means that you can define a data object using the type of the parent business object and then assign it an instance of the child business object. When you do this, you only have access to the attributes and methods defined in the parent business object. However, those methods that were re-defined in the child business object, behave in the way they were defined in the child object.

Method Overriding

A business object can have multiple methods with the same name but different sets of arguments. The runtime engine decides which method to run based on the arguments passed when invoking the method.

Attribute Shadowing

The child business object can define an attribute of the same name and type of an attribute defined in the parent object.

Abstract Business Objects

You can mark a business object as abstract to indicate that it only defines data and methods to reuse in child business objects, but it does not make sense to create an instance of this business object. Marking it as abstract ensures that you cannot create an instance of this business object.

Working with Business Object Inheritance

You can use business object inheritance to reuse data and methods, avoiding code duplication and making maintenance easier.

This section describes how to create a child business object and how to mark an object as abstract.

How to Create a Child Business Object

You can create a child business object to reuse the attributes and methods defined in a business object.

To create a child business object:

1. In the Applications window, right-click the business object you want to use as the parent business object.
2. Select Create Child Object.

   The Create Business Object dialog box appears.
3. Enter a name and select a destination module.
4. Click OK.

   The Business Object editor with the child business object opens. Note that there is a parent field that indicates which is the parent object.

Note:

You can also define a child object by selecting its parent object for an existing business object.

What Happens When You Create a Child Object

The child object can use the attributes and methods defined in the parent business object.

How to Mark a Business Object as Abstract

You can mark a business object as abstract to indicate that it only defines behavior and data.

To create an abstract business object:

1. Open the business object you want to mark as abstract in the Business Object editor.
2. Select the Abstract check box.

What Happens When You Mark a Business Object as Abstract

You can mark a business object as abstract to indicate that it only defines behavior and data for child objects to reuse. You cannot instantiate an abstract business object, you can only use it to define the type of a data object and then assign it an instance of a child business object.

11

Working with Human Tasks

This chapter describes how the business catalog displays and handles human tasks. It also describes how to update a user task using the update task.

This chapter includes the following sections:

- Introduction to Human Tasks in BPM
- Using Human Task Patterns in Oracle BPM
- Updating User Tasks Using Update Tasks

For detailed information on how to create, edit, and configure human tasks, see Designing Human Tasks in Oracle BPM.

Introduction to Human Tasks in BPM

Human Tasks are interactive steps performed by the end user and are used to implement user tasks. Human Tasks contain a payload (data associated to the Human Task), forms, and policies to define escalation, notifications and reminders.

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.

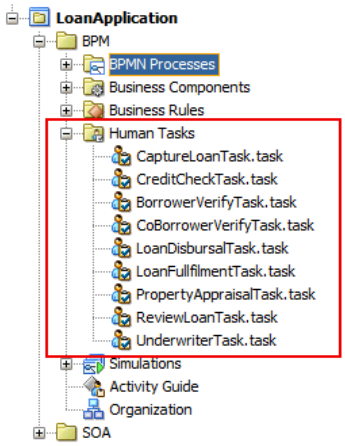
You can add new Human tasks to your project in the following ways:

- 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 11-1 shows a Human Task component in the Sales Quote example.

Figure 11-1 Human Task components in the Business Catalog



You can define the participants in a Human Task using swimlanes in the BPMN process or dynamically by evaluation and expression based on process data. The workflow pattern used for the human tasks defines the number of participants. For more information about workflow patterns, see Using Human Task Patterns in Oracle BPM.

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 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.

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: combines different routing patterns, each pattern represents a different stage.

- FYI: notifies the participants, the process does not wait for this task to complete before running the following tasks in the process flow.
- Group
- Initiator: the process starts when the end user fills in a form.
- Management: assignees perform the work in a serial sequence based on a management chain.
- User: a single assignee performs the work.
- Parallel: assignees perform work in parallel.
- Manual: these are tasks performed by humans that are not managed by the BPM runtime.

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 Using Approval Management.

Updating User Tasks Using Update Tasks

Update tasks enable you to update certain properties of specific user tasks in your process. You can choose to update a specific user task, all user tasks or to dynamically generate the ID of the user task to update using an expression. Update tasks do not require you to specify the taskId or the context.

You can use user tasks to perform the update operations based on the status of your process or the different paths the process instance takes. Update tasks enable you to model the updating sequence in your business process making the process flow easier to understand.

You can only update active user tasks. If the user task is completed or did not start yet, then you cannot update it using the update task.

Update Task Operations

Update tasks enable you to change the value of some of the properties of the human task used to implement the user task. For a description of the properties involved in the different operations, see Creating a Human Task from Oracle BPM Studio.

Update tasks support the following operations:

- **Update Outcome**: updates the outcome of the human task used to implement the specified user task. You must specify the new outcome using an expression. You can choose to write the expression using literals, simple expressions or XPath expressions.
- **Update Priority**: updates the priority of the human task used to implement the specified user task. You must specify the new priority using an expression. You can choose to write the expression using literals, simple expressions or XPath expressions.
- **Withdraw**: withdraws the task from the task list shown to other assignees.
- **Suspend**: suspends the task.
- **Resume**: resumes the task.

- **Escalate**: escalates the task using the escalation hierarchy or escalation callbacks specified in the human task that implements the specified user task.
- **Reassign**: reassigns the human task used to implement the specified user task, to a new user. You must specify the new user using an expression. You can choose to write the expression using literals, simple expressions or XPath expressions
- **Suspend Timers**: suspends the timers that are running over the human tasks that implements the specified user task. Suspending the timers results in the suspension of the expiration policies defined for the human task.

How to Update a User Task Using Update Tasks

You can update different parameters of user tasks from your business process using an update task.

To update user tasks using update tasks:

1. Open the business process that contains the user task.
2. Add an update task.
3. Configure the update task.

How to Configure Update Tasks

You can configure an update task to perform different operations over the user tasks you specify.

To configure an update task:

1. Right-click the update task.
2. Select Properties.
3. Click the **Implementation** tab.
4. From the target list select how to specify the target user task.

   The available options are:

   - **Select a specific user task**: The user tasks in your process appear in this list, you can select any of them and update it using this update task.
   - **All User Tasks**: This option enables you to update all the user tasks in the process.
   - **Task Id**: This option enables you to specify a task Id that identifies the user task, using an expression. When you select this option the Task Id field appears.
5. From the Operation list, select an operation.

   For more information about the available operations, see Update Task Operations.

12

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:

- Introduction to Services and References
- Introduction to Service Adapters in Oracle BPM
- Introduction to Oracle Mediator in Oracle BPM
- Introduction to BPEL Processes in Oracle BPM
- Using Services in Oracle BPM
- Using References in Oracle BPM
- Customizing Services and References

Introduction to Services and References

Some flow objects, such as Service Task, Send Task and Receive task, require you to define a service or a reference to implement them. You can define service interfaces using the BPMN Editor or use existing interfaces from the Business Catalog.

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:

- 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.

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

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

Introduction to Callbacks

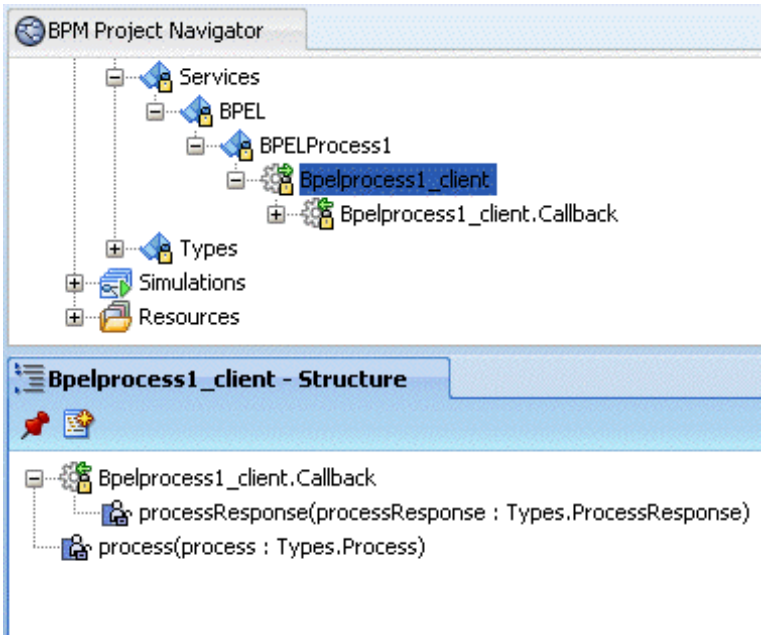
If a service is asynchronous and contains a callback interface, then the component in the business catalog contains a callback inner component. The callback inner component groups all the callback operations in the service.

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 12-1 shows a service with a callback interface in the business catalog.

Figure 12-1 Service with Callback Interface



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
- Coherence
- A variety of databases
- Direct binding
- EJB service
- Files
- FTP
- Healthcare
- HTTP
- JMS
- LDAP

- MFT Binding
- MQSeries
- MSMQ
- Oracle applications
- REST Binding
- Sockets
- Third-party adapters
- User Messaging Service
- Web services

For a detailed description of service adapters see:

- "Getting Started with Binding Components" in *Developing SOA Applications with Oracle SOA Suite*
- *Understanding Technology Adapters*

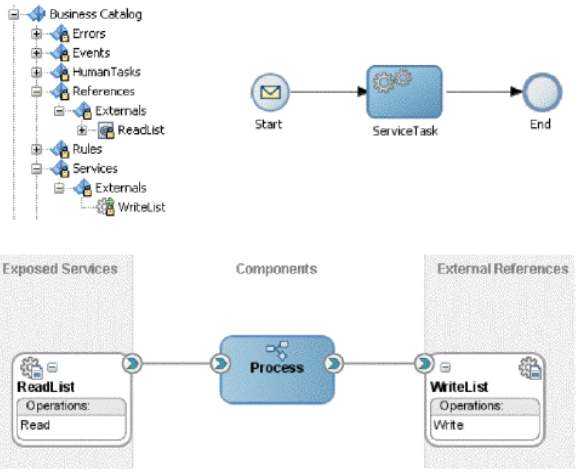
When you add a SOA service adapter to the SOA composite of a BPM project, the service adapter automatically appears in the business catalog. The business catalog stores the SOA service adapters in different modules depending on the swimlane of the SOA composite where you added the SOA service adapter:

- 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 12-2 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 12-2 Adapter Services in Oracle BPM



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 Applications window, 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.

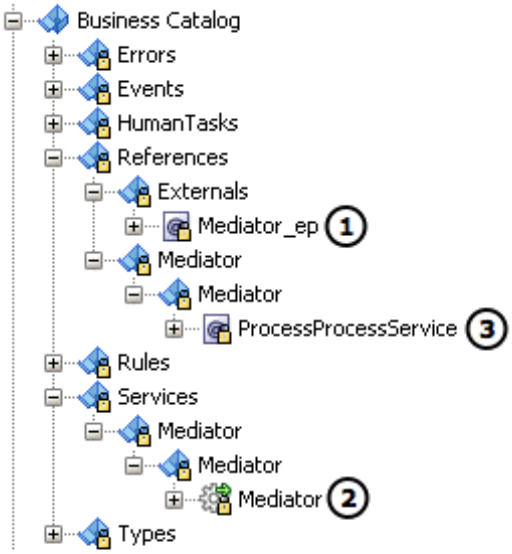
Introduction to Oracle Mediator in Oracle BPM

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 SOA Composite 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 12-3 Mediator Components in the Business Catalog



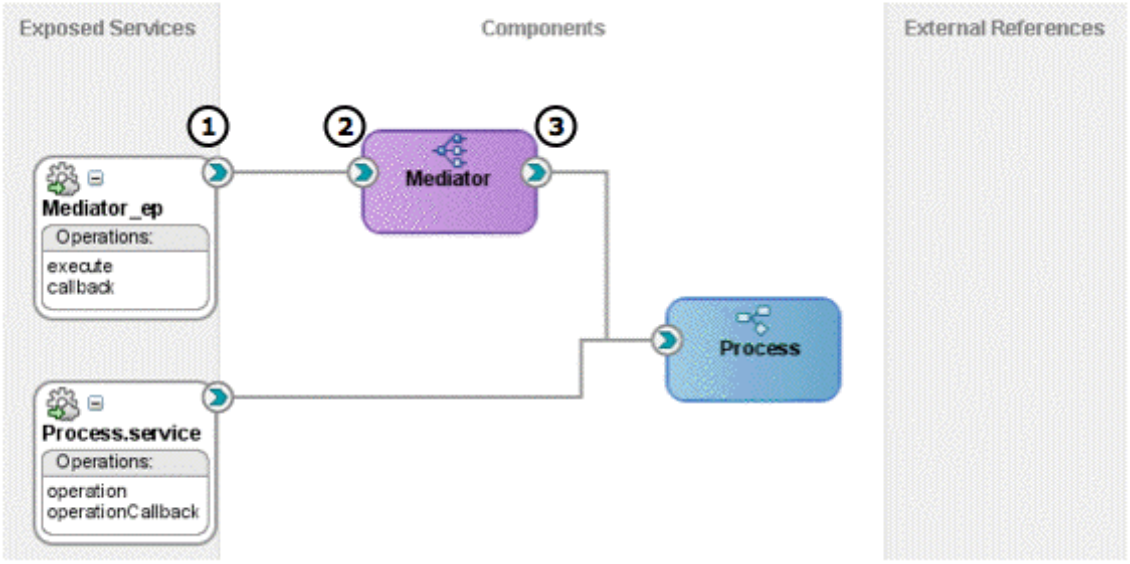
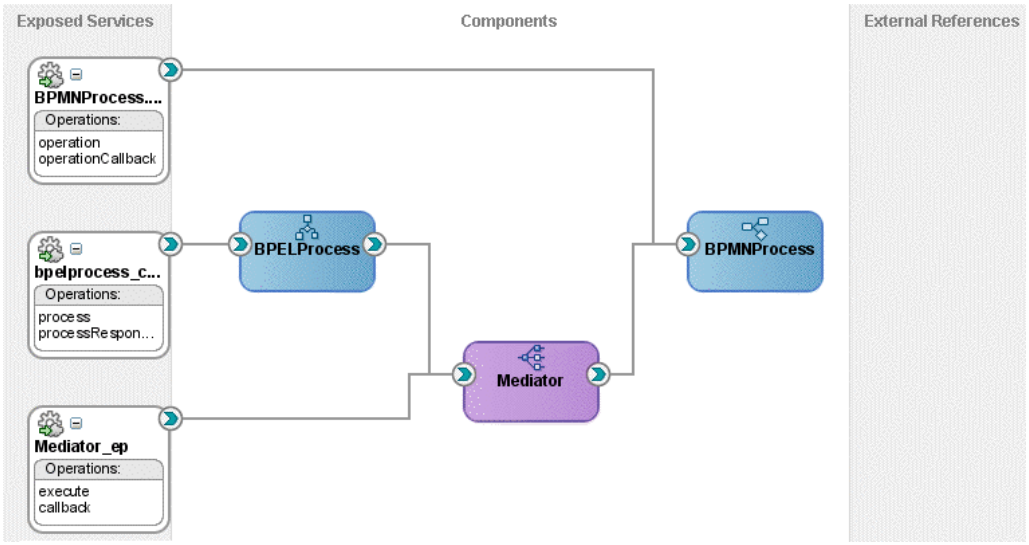


Figure 12-4 shows a BPEL process that invokes a BPMN process through a mediator. Note that the service handle of the mediator connects to the BPEL process and the reference handle connects to the BPMN process.

Figure 12-4 BPEL Process Using a Mediator to Invoke a BPMN Process



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 1 in Figure 12-3 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 12-3 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 Using Message Events with an Interface from the Business Catalog to Define Your Process Interface and 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 3 in Figure 12-3 shows the reference interface component ProcessService for the mediator component shown in the SOA composite.

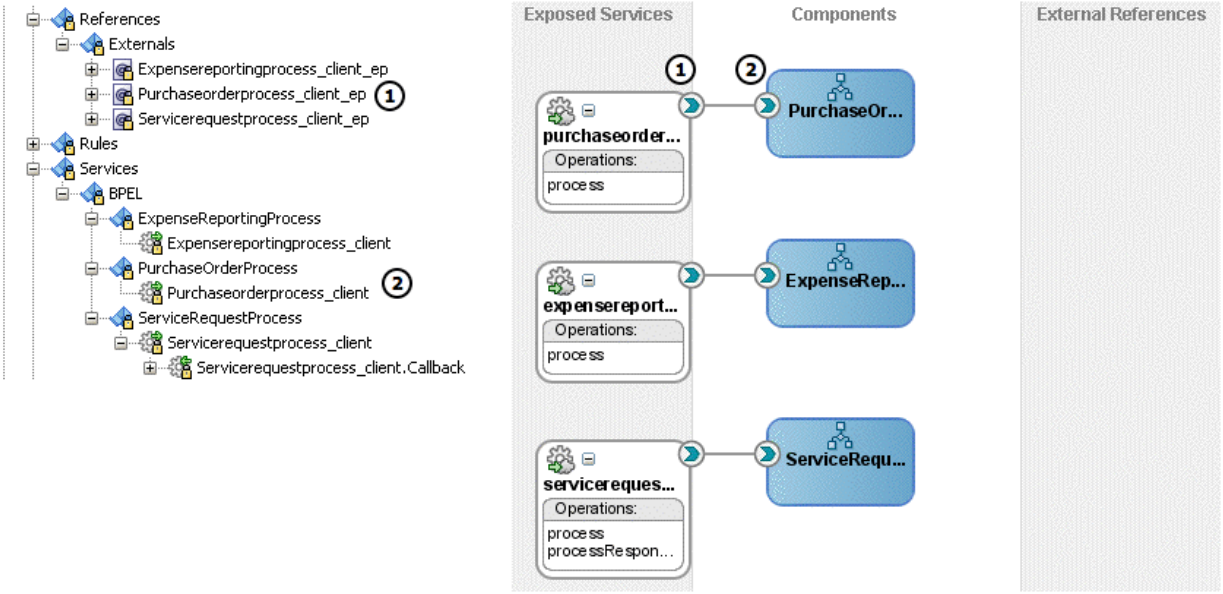
For more information on SOA mediators, see part "Using the Oracle Mediator Service Component" in *Developing SOA Applications with Oracle SOA Suite*.

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 12-5 shows an SOA composite that contains multiple BPEL processes and how the business catalog displays them.

Figure 12-5 BPEL Process Components in Business Catalog



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 12-5 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 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 Using Message Events with an Interface from the Business Catalog to Define Your Process Interface and 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 *Developing SOA Applications with Oracle SOA Suite*.

Using Services in Oracle BPM

The service tasks, message events, and send and receive tasks require you to define a service to implement them.

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 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 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 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.

Using References in Oracle BPM

The message events and receive tasks flow objects enable you to define an interface using a reference component from the business catalog.

- Message events

  See 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 Using Send and Receive Tasks 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 receive tasks.

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.

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 Applications window, 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 How to Customize an Operation for information on how to customize an operation.
8. Click **OK**.

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 box 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**.

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.

13

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:

- Introduction to Business Rules in Oracle BPM
- Assigning an Existing Business Rule to a Business Rule Task
- Creating a Business Rule from Oracle BPM Studio

For detailed information about Oracle Business Rules, see *Designing Business Rules with Oracle Business Process Management*.

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

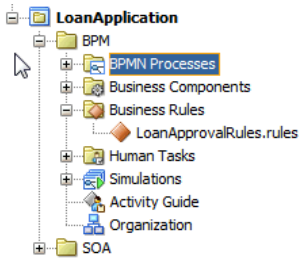
The business catalog displays the business rules in your project in the predefined module Rules. It stores each rule in a module named as the package of the business rule dictionary.

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 *Designing Business Rules with Oracle Business Process Management*.

Figure 13-1 shows a business rule component in the Sales Quote example.

Figure 13-1 Business Rules Components in the Business Catalog



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.

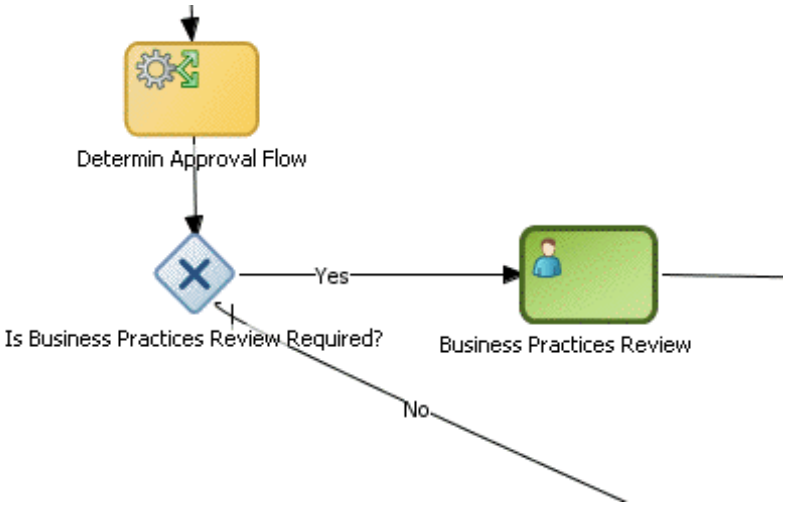
After a business rule task, you can add an exclusive gateway that determines the flow of the process based on the value of a data object that contains the result of running the business rule task.

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 13-2 shows a business rule task in the Sales Quote example.

Figure 13-2 Business Rule Task in the Sales Quote Example



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.

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.

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.

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.

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 simple types or 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.

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 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 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.

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.

   The data associations are automatically created mapping the existing data objects to arguments.

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.

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.

14

Sending Notifications

This chapter describes how to use the notification task to communicate with end users of the business process. It describes the different types of notifications tasks and how to configure each of them.

This chapter includes the following sections:

- Introduction to Notifications
- Sending Email Notifications
- Sending a User Notification
- Sending an SMS Notification
- Sending a Voice Notification
- Sending an IM Notification

Introduction to Notifications

The notification task allows you to send different types of notifications to the users of the application.

It supports the following types of notifications:

- Email
- User
- SMS
- Voice
- IM

This tasks uses the Oracle Notification Service.

To configure this task you must provide expressions for the different fields of the notification and in some cases you can use the Identity Lookup browser to select one or more users. When you write the expressions you can use any variables accessible from the notification task context such as process data objects or predefined data objects.

Sending Email Notifications

Using the notification service, you can send an email message to users at a certain point in the business process.

You can use the Identity Lookup browser to select one or more users.

How to Send an Email Notification

To send an email notification:

1. Open your BPMN business process.
2. Add a **Mail Notification** task.
3. Right-click the Mail Notification task.
4. Select **Properties**.
5. Click the **Implementation** tab.
6. In the **General** tab configure the properties described in How to Configure Email Notification General Properties
7. Click the **Content** tab to configure the properties described in How to Configure Email Notification Content Properties.
8. Click the **Attachments** tab to configure the properties described in How to Configure Email Notification Attachment Properties.
9. Click the **Headers** tab to configure the properties described in How to Configure Email Notification Header Properties.
10. Click OK.

How to Configure Email Notification General Properties

Use the following table to configure the general properties of an email notification. Note that some of the properties are optional.

Table 14-1 Email Notification General Properties

Property	Data Type	Optional	Description
From	string	No	Specifies the email address used to send the email notification.
To	string	No	Specifies the email addresses of the receivers of the email notification. You can specify these email addresses using expressions or using the Identity Lookup browser.
Bc	string	Yes	Specifies the email addresses of the additional receivers of the email notification. You can specify these email addresses using expressions or using the Identity Lookup browser.
Bcc	string	Yes	Specifies the email addresses of the hidden additional receivers of the email notification.ou can specify these email addresses using expressions or using the Identity Lookup browser
Reply	string	Yes	Specifies the email addresses to use when replying the email notification. You can specify this email address using expressions or using the Identity Lookup browser

How to Configure Email Notification Content Properties

Use the following table to configure the content properties of an email notification.

Table 14-2 Email Notification Content Properties

Property	Data Type	Optional	Description
Subject	string	No	Specifies the subject of the email notification. You must specify this value using expressions. If you are creating an actionable email, avoid using '&' in the subject. It can cause truncated subject lines in some email clients including some versions of Outlook.
Body	string	No	Specifies the body of the email notification. You must specify this value using expressions.

How to Configure Email Notification Attachment Properties

Use the following table to configure the attachment properties of an email notification. Note that some of the properties are optional.

Table 14-3 Email Notification Attachment Properties

Property	Data Type	Optional	Description
Name	string	No	Specifies the name of the attachment. You must specify this value using expressions. The default value is "attachment" followed by a number.
Mime Type	string	No	Specifies the attachment content type. You must specify this value using expressions. The default value is "text/html".
Encoding	string	Yes	Specifies the encoding of the email notification. You must specify this value using expressions.
Value	Any	No	Specifies the attachment file. You must specify this value using expressions.

How to Configure Email Notification Header Properties

To configure the header properties you can add one or more headers. Note that configuring header properties is optional.

To add a header:

1. Click the **Add** button.

   The Create Header dialog box appears.

2. Provide the name and value properties.

Table 14-4 Header Properties

Table 14-4 (Cont.) Header Properties

Property	Data Type	Description
Name	string	Specifies the name of the header. You must specify this value using expressions.
Value	string	Specifies the value of the header. You must specify this value using expressions.

3. Click **OK**.

Sending a User Notification

User notification allows you to send a message to the users in a certain point of the process.

You must use the communication media defined for that user.

How to Send a User Notification

To send an email notification:

1. Open your BPMN business process.
2. Add a **User Notification** task.
3. Right-click the User Notification task.
4. Select **Properties**.
5. Click the **Implementation** tab.
6. In the **General** tab configure the properties described in How to Configure User Notification General Properties.
7. Click the **Properties** tab to configure the properties described in How to Configure User Notification Properties.
8. Click **OK**.

How to Configure User Notification General Properties

Use the following table to configure the general properties of a user notification. Note that some of the properties are optional.

Table 14-5 Email Notification General Properties

Property	Data Type	Optional	Description
To	string	No	Specifies the email addresses of the receivers of the user notification. You can specify these email addresses using expressions or using the Identity Lookup browser.
Subject	string	Yes	Specifies the subject of the user notification. You must specify this using expressions.

Table 14-5 (Cont.) Email Notification General Properties

Property	Data Type	Optional	Description
Message	string	Yes	Specifies the message to send using the user notification. You must specify this using expressions.

How to Configure User Notification Properties

You can add one or more properties. Note that configuring properties is optional.

To add a header:

1. Click the **Add** button.

   The Create Property dialog box appears.

2. Provide the name and value properties.

Table 14-6 Properties

Property	Data Type	Description
Name	string	Specifies the name of the property. You must specify this value using expressions.
Value	string	Specifies the value of the property. You must specify this value using expressions.

3. Click **OK**.

Sending an SMS Notification

Using the notification service, you can send an SMS a message to users in a certain point of the business process.

You can use the Identity Lookup browser to select one or more users.

How to Send an SMS Notification

To send an email notification:

1. Open your BPMN business process.
2. Add a **SMS Notification** task.
3. Right-click the Mail Notification task.
4. Select **Properties**.
5. Click the **Implementation** tab.
6. Configure the General Properties described in How to Configure SMS Notification General Properties.

7. Click **OK**.

How to Configure SMS Notification General Properties

Use the following table to configure the general properties of an SMS notification. Note that some of the properties are optional.

Table 14-7 Email Notification General Properties

Property	Data Type	Optional	Description
From #	string	No	Specifies the cellphone number used to send the SMS notification.
To #	string	No	Specifies the cellphone numbers of the receivers of the SMS notification. You can specify these cellphone numbers using expressions or using the Identity Lookup browser.
Subject	string	Yes	Specifies the subject of the SMS notification. You must specify this using expressions.
Body	string	Yes	Specifies the body of the SMS notification. You must specify this using expressions.

Sending a Voice Notification

Using the notification service, you can send a voice notification to users in a certain point of the business process.

You can use the Identity Lookup browser to select one or more users.

How to Send an Voice Notification

To send a voice notification:

1. Open your BPMN business process.
2. Add a **Voice Notification** task.
3. Right-click the Mail Notification task.
4. Select **Properties**.
5. Click the **Implementation** tab.
6. Configure the General Properties described in How to Configure Voice Notification General Properties.
7. Click **OK**.

How to Configure Voice Notification General Properties

Use the following table to configure the general properties of a voice notification. Note that some of the properties are optional.

Table 14-8 Email Notification General Properties

Table 14-8 (Cont.) Email Notification General Properties

Property	Data Type	Optional	Description
To #	string	No	Specifies the telephone numbers of the receivers of the voice notification. You can specify these telephone numbers using expressions or using the Identity Lookup browser.
Mime Type	string	Yes	Specifies the content type. You must specify this value using expressions. The default value is "text/html"
Body	string	Yes	Specifies the body of the voice notification. You must specify this using expressions.

Sending an IM Notification

Using the notification service, you can send an Instant Message to users in a certain point of the business process.

You can use the Identity Lookup browser to select one or more users.

How to Send an IM Notification

To send an email notification:

1. Open your BPMN business process.
2. Add a **IM Notification** task.
3. Right-click the Mail Notification task.
4. Select **Properties**.
5. Click the **Implementation** tab.
6. Configure the General Properties described in How to Configure IM Notification General Properties.
7. Click **OK**.

How to Configure IM Notification General Properties

Use the following table to configure the general properties of an IM notification. Note that some of the properties are optional.

Table 14-9 Email Notification General Properties

Property	Data Type	Optional	Description
To	string	No	Specifies the user IDs of the receivers of the IM notification. You can specify these user IDs using expressions or using the Identity Lookup browser.
Body	string	Yes	Specifies the body of the IM notification. You must specify this using expressions.

15

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:

- Introduction to SOA Composites
- Opening the SOA Composite in a BPM Project
- Opening BPMN Processes from the SOA Composite in a BPM Project
- Adding a BPMN Process from the SOA Composite Editor
- Integrating with BPEL Processes Using the SOA Composite
- Adding a BPMN Process as a Partner Link in a BPEL Process
- Connecting to a BPMN Process Using Web Services
- Building a BPM Project

For detailed information about SOA Composites, see *Developing SOA Applications with Oracle SOA Suite*.

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. They are a SOA composite project that also includes BPMN component types and the configuration related to BPMN components such as calendars and organizational units. 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 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.

Reusable processes do not appear in the SOA Composite. When you modify a business process and transform it into a reusable process, it disappears from the SOA Composite. For more information about reusable subprocesses, see Introduction to Invoking a Process Using Call Activities.

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.

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 instances of the SOA components. However the references components are not created automatically. The instances linked to services are created when the service is invoked and consequently a composite is created. In the case of a human task with a wire to a BPMN process, the BPMN process instance is created when the BPMN process is triggered.

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 *Developing SOA Applications with Oracle SOA Suite*.

Some of the activities in your BPMN process require you to assign them an SOA component to implement them. For most of these components you can choose to add them from the Oracle BPM Studio user interface, or you can 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

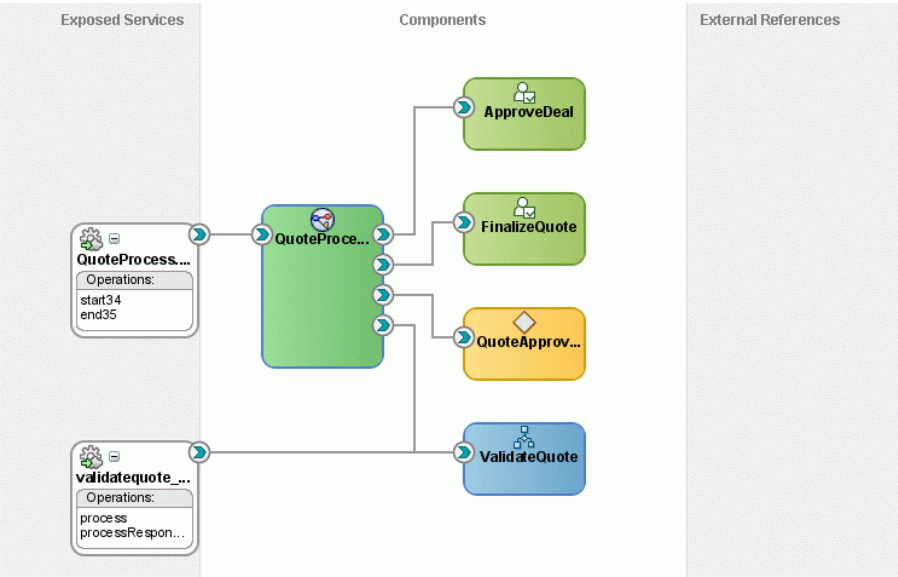
- BPEL Processes

Mediators and BPEL Processes are only available from the SOA Composite editor.

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:

- Working with Business Components
- Controlling the Process Flow

Figure 15-1 BPMN process in an SOA Composite



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.

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.

Opening the SOA Composite in a BPM Project

BPM projects are layered on top of a SOA project. The SOA project contains a 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.

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**s window.
2. Double-click the composite file located in the SOA directory of your project.

   The composite file name matches the name of your project.

   The SOA Composite editor opens.

Opening BPMN Processes from the SOA Composite in a BPM Project

BPM projects use the SOA technology, therefore they contain a SOA Composite. You can use the SOA Composite editor to view the dependencies of your BPM processes with other components in your BPM project.

You can also add new components to your BPM project.

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 Applications window.

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.

Adding a BPMN Process from the SOA Composite Editor

You can add new BPMN processes directly from the SOA Composite editor

You do not have to switch to the Applications window.

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 Components section in the Components window.
3. Drag the selected component to the Components area in the SOA Composite editor.

   The BPMN 2.0 Process wizard appears.

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 Applications window.

To edit the BPMN process right-click it and select edit, or double click the BPMN process.

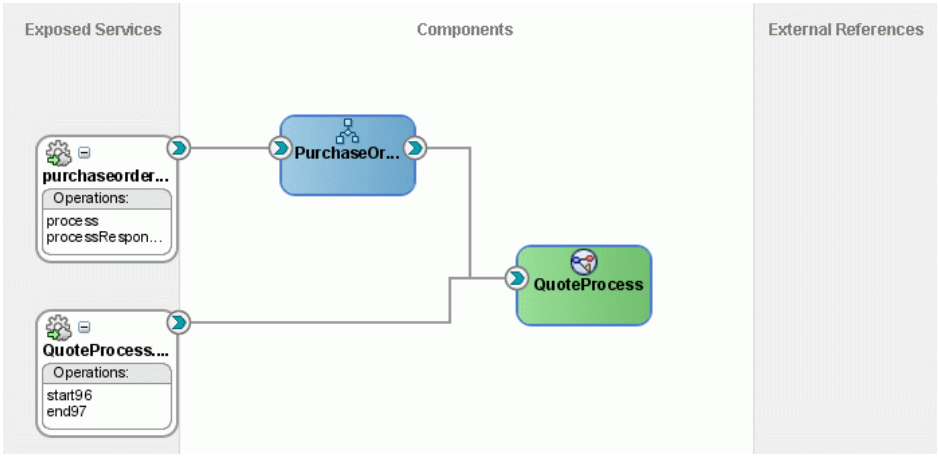
Integrating with BPEL Processes Using the SOA Composite

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 *Developing SOA Applications with 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 15-2 BPMN Process as a Partner Link in a BPEL Process



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.

You can then use the BPEL editor to invoke the BPMN process from the activities in the BPEL process.

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.

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.

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 Defining the Process Interface.

To connect to a BPMN process using a custom web service client you must log in to Oracle Enterprise Manager and vie the dashboard page for the composite. This page contains a link to the WSDL URL. The WSDL contains the service location information needed to connect to the BPMN process.

Building a BPM Project

When you deploy a BPM project to the SOA runtime, it automatically builds your project. If the build fails then the deployment fails too.

You can build your BPM project from Oracle JDeveloper to view the errors in the build and fix them.

After you build the BPM project, the Compiler Log window displays the results.

If there are any errors, you can select the Compiler tab and click the errors to open the corresponding editor and correct them.

How to Build a BPM Project

To build a BPM project:

1. Open the BPM project.
2. Select the **Applications** 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.

Part V

Controlling the Process Flow

This part describes how to implement the different BPMN flow objects 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:

- Controlling the Process Flow
- Adding Delays_ Deadlines_ and Time Based Cycles to Your Process
- Handling Errors
- Using Fault Handling in BPM
- Communicating With Other BPMN Processes and Services
- Defining the Process Interface
- Communicating Business Processes Using Correlations
- Defining Conversations
- Writing Expressions
- Writing BPM Scripts
- Debugging a BPM Project

16

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:

- Introduction to Controlling the Process Flow
- Introduction to Loop and Multi-Instance Markers in Subprocesses
- Suspending the Current Process Flow to Run an Alternative Process Flow

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

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 "BPMN Flow Object Reference" in the *Developing Business Processes with Oracle Business Process Composer*.

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 Adding Delays_ Deadlines_ and Time Based Cycles to Your Process.

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 Handling Errors.

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 a BPMN process (asynchronous inter-process communication) or an external service such as a BPEL process. For more information about message events, see Communicating With Other BPMN Processes and Services.

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 Communicating With Other BPMN Processes and Services.

Loop Markers

Loop markers enable you to run a subprocess multiple times based on a certain condition. For more information about loop markers, see Introduction to Loop and Multi-Instance Markers in Subprocesses.

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 Introduction to Loop and Multi-Instance Markers in Subprocesses

Suspending the Current Process Flow

You can suspend a flow object, a subprocess or a process. For more information about suspending the current process flow, see Suspending the Current Process Flow to Run an Alternative Process Flow.

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.

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:
   a. Select the expression language.

      Possible options are Simple or XPath.

   b. 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:
   a. Select **Loop Maximum**.
   b. 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**.

How to Configure Multi-Instance Markers

You can configure a multi-instance marker to run a 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. Select the mode:
   - Sequential: specifies that the each token must complete the subprocess before the next token starts to run the subprocess.
   - Parallel: specifies that tokens run in parallel

   Note:

   When using parallel mode consider that using process data objects to store information that results from running the subprocess may result in instances overwriting the information. To avoid this use subprocess data objects.
6. Specify the Loop Cardinality:
   a. Select the expression language.

      Possible options are Simple or XPath.

   b. 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.
7. Optionally, you can specify the Completion Condition:
   a. Select the expression language.

      Possible options are Simple or XPath.

   b. 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.
8. 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.

9. 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.

10. Click **OK**.

Suspending the Current Process Flow to Run an Alternative Process Flow

To suspend a process you must configure an existing start event of a subprocess or add a new one. When the process instance arrives to the start event of the event subprocess, the main process flow is suspended and the BPM runtime runs the process flow in the even subprocess.

To suspend a flow object or a sub-process you must configure an existing boundary event or add a new one. The boundary event can be a message event, a timer event or a signal event. When the process instance reaches the flow object with the boundary event, the main process flow is suspended and the BPM runtime runs the event handler sequence flow, the alternative sequence flow.

How to Configure a Flow Object to Suspend the Current Process Flow

You can configure boundary events and event subprocesses to suspend the current process flow.

To configure a flow object to suspend the current process flow:

1. Open the BPMN process.
2. Right-click the boundary event or subprocess that you want to use to suspend the process flow.
3. Select Properties.
4. Select the Implementation tab.
5. Select the Suspending Event option.

How to Resume the Suspended Process Flow

To resume the process flow you must set the value of the predefined variable action to one of the following values:

- RESUME

  Resumes the suspended process flow.

- SEND

  Moves the process instance to the next flow object in the process flow that caught the suspension. The suspended scope is canceled.

You can set the value of the predefined variable action in following ways:

- Using data associations

  For more information, see Introduction to Data Associations.

- Using BPM scripts

For more information, see Writing BPM Scripts.

After running a task in an alternative sequence flow, the BPM runtime checks the value of the predefined variable action. If the value of the predefined variable action is RESUME or SEND, it resumes the main process flow and cancels the event handler sequence flow.

17

Adding Delays, Deadlines, and Time-Based Cycles to a 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:

- Introduction to Timer Events
- Adding a Delay to the Process Flow
- Designing a Process to Start Based on a Time Condition
- Configuring a Deadline for an Activity
- Configuring a Deadline for a BPMN Process
- Running Additional Activities
- Configuring Timer Events

Introduction to Timer Events

Timer events enable you to control the flow of your process using a time condition. Timer events are not based on the business calendar definitions.

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

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.

Note: Make changes to calendar only at design time as runtime changes to calendar are not supported. The technical reason behind this is as there is no a calendar cache accessing the database for calendars causes performance issues.

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 17-1 Delaying the Process Flow

How to Add a Delay to the Process Flow

You can add a delay between two 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. See How to Configure a Timer Event To Use a Specific Date and Time.

   If you want to delay the process for a certain period, then you must configure the timer start event as cycle. See How to Configure a Timer Event to Use an Interval.

   If you want the timer to run using a specific schedule, then you must configure the timer as schedule. See How to Configure a Timer Event to Run Periodically.

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.

Designing a Process to Start Based on a Time Condition

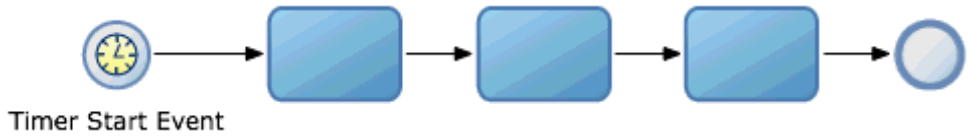
You can add a timer start event to your process to configure it to be triggered based on a time condition. When the time condition specified in the timer start event evaluates to true, the BPMN Service Engine creates a new instance in the process.

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 17-2 Starting a Process Based on a Time Condition



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 event as time date. See How to Configure a Timer Event To Use a Specific Date and Time.

   If you want the process to start after a certain period, then you must configure the timer start event as cycle. See How to Configure a Timer Event to Use an Interval.

   If you want the process to start based on a specific schedule, then you must configure the timer as schedule. See How to Configure a Timer Event to Run Periodically.

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.

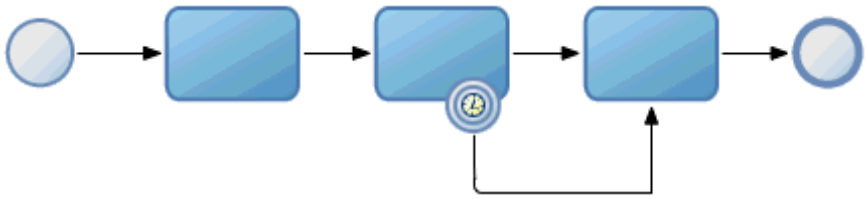
Configuring a Deadline for an Activity

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 17-3 Activity Deadline



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. See How to Configure a Timer Event To Use a Specific Date and Time.

   If you want the deadline to happen after a certain period, then you must configure the boundary timer catch event as cycle. See How to Configure a Timer Event to Use an Interval.

   If you want to set the deadline using a specific schedule, then you must configure the timer as schedule. See How to Configure a Timer Event to Run Periodically
6. In the Implementation tab, in the Timer Properties dialog box, select **Interrupting Event**.

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.

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.

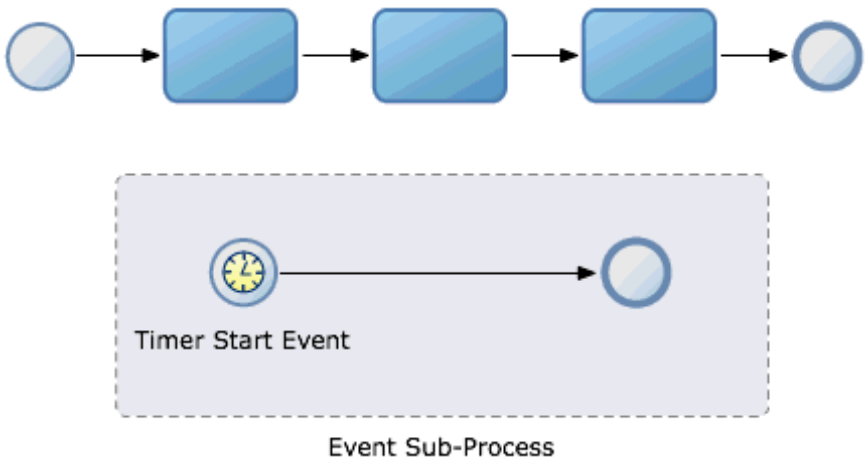
The timer event is only active while the token remains in the process.

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 17-4 Process Deadline



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:
   a. Right-click the start event in the event subprocess.
   b. Select **Properties**.
   c. Click the **Implementation** tab.
   d. From the Implementation Type list, select **Timer**.
   e. Select **Interrupting Event**.
   f. If you want the deadline to happen on a specific date, then you must configure the timer event as time date. See How to Configure a Timer Event To Use a Specific Date and Time.

      If you want the deadline to happen after a certain period, then you must configure the timer event as cycle. See How to Configure a Timer Event to Use an Interval.

      If you want to set the deadline using a specific schedule, then you must configure the timer as schedule. See How to Configure a Timer Event to Run Periodically

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.

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 email 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 17-5 Running Additional Activities While an Activity is Running

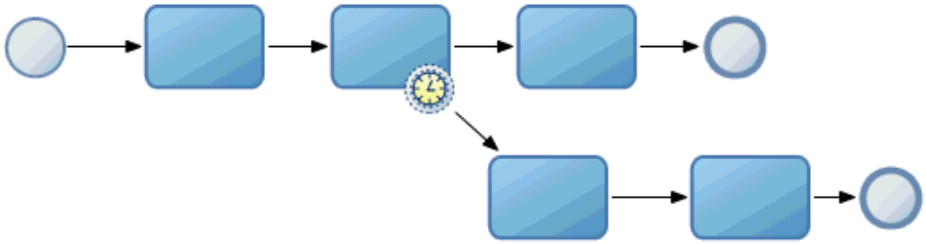
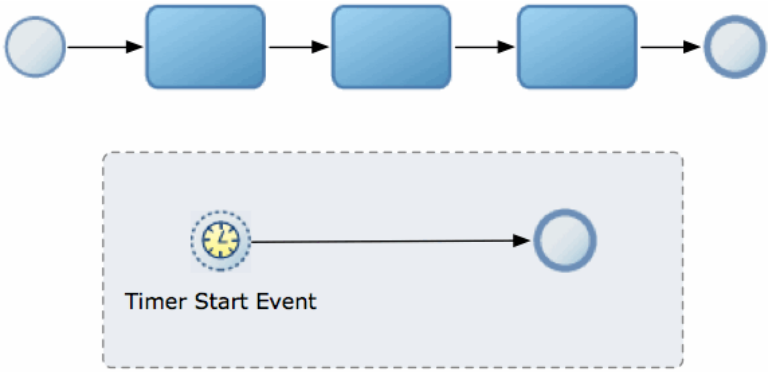


Figure 17-6



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.

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.

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 a long time 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. See How to Configure a Timer Event To Use a Specific Date and Time

   If you want the additional activities to run periodically, then you must configure the timer event as cycle. See How to Configure a Timer Event to Use an Interval.

   If you want to run additional activities using a specific schedule, then you must configure the timer as schedule. See How to Configure a Timer Event to Run Periodically
8. Add the additional activities to the subprocess event.

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.

Configuring Timer Events

You can configure timer events 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.

You can also configure a timer event to run periodically. The different supported granularities are daily, weekly and monthly. In all these cases you can configure the timer event to calculate the dates using the calendar rules defined for that project and to reschedule any event that matches a holiday.

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 box.
   - 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 Writing Expressions in Timer Events in 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**.

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.

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 **Time Cycle**.
5. Provide a time interval or select Use Expression and write an expression that returns an Interval.

   See Writing Expressions in Timer Events for more information

6. Click **OK**.

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.

How to Configure a Timer Event to Run Periodically

You can configure a timer event to run daily, weekly or monthly at one or more specified times. You can also configure it to schedule the event using the calendar rules defined for the organization and to use a specified policy for rescheduling the event in case it falls on a holiday our outside the work schedule.

To configure a timer event to run periodically:

1. Right-click the timer event.
2. Select **Properties**.
3. Click the **Implementation** tab.
4. Select **Time Schedule**.
5. Select the tab that corresponds to the period you want to define.

   The available options are:

   - Daily
   - Weekly
   - Monthly
6. Click the **Add** button to specify the exact time the event happens.

   You may configure the event to run multiple times during a day, a week or a month.

   Table 17-1 shows how to specify the time according to the period you selected.

Table 17-1 Time Definition According to Selected Period

Selected Period	Time Definition
Daily	You must define the hour of the day.
Weekly	You must specify the days of the week and the hour.
Monthly	You must specify the month, the week, the day and the hour. To configure the event to run on a specific day of the month, select Day of Month in the Week column and the Day column allows you to specify the day of the month. To configure an event to run every month, select All in the Month column.

7. Additionally you can configure the following optional settings:

   - **Run from**: enables you to define the date and time for the event to start running.
   - **Run to**: enables you to define the date and time for the event to stop running.
   - **Repetitions**: enables you to define the number of times the event runs. After the event runs the number of times you defined, it stops running.

- **Use Calendar Rules**: specifies that the event runs based on the calendar rules defined for the organization.

  If you select this option, then you must define how to reschedule the event in case it is scheduled to run on a holiday or outside the work schedule. The available rescheduling options are:

Rescheduling Option	Description
No Reschedule	The event does not run and it is not rescheduled.
As soon as possible	The event is rescheduled for the next available working hour.
As soon as possible but at the same hour	The event is rescheduled for the next available working day at the same time

8. Click OK.

18

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:

- Introduction to Error Handling
- Using Business Exceptions
- Using System Exceptions
- Typical Flow of an Exception
- Handling Exceptions in a Business Process
- Configuring Catch Events to Recover from an Exception
- Throwing Exceptions in Subprocesses or Reusable Processes
- Handling Exceptions in Subprocesses
- Handling Errors in a Peer Process Using Message Events

Introduction to Error Handling

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 on which the BPMN Service Engine is running, while process errors are unexpected situations within the process flow itself.

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 *Administering 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 How Do BPMN Errors Affect the SOA Composite Status.

You can configure activities and events to force the BPM runtime to add a checkpoint after completing them. To do this you must select the Force Commit After Execution option. This is similar to the Dehydrate action in BPEL. You can use this option to avoid re-running non-idempotent activities on recovery, which in case of an error rolls back the transaction.

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.

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.

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 18-1 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 18-1 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

Table 18-1 (Cont.) System Exceptions

System Exception	Module	Description	Error Event
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

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

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 *Administering 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.

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 is 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 *Administering Oracle SOA Suite and Oracle Business Process Management Suite*, for more information about the Enterprise Manager fault recovery system.

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 is 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 *Administering Oracle SOA Suite and Oracle Business Process Management Suite*, for more information about the Enterprise Manager fault recovery system.

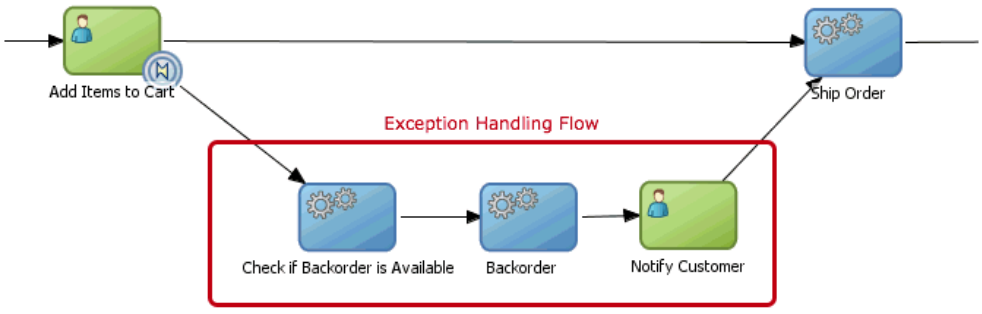
Handling Exceptions in a Business Process

You can handle the exceptions that occur in an activity using a boundary error catch event or 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 18-1 shows a process that handles an error using a boundary error catch event.

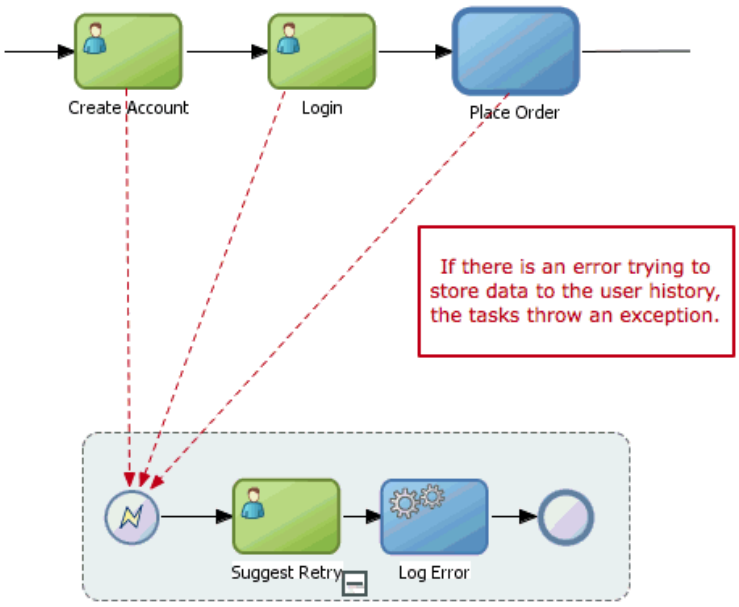
Figure 18-1 Boundary Error Catch Event



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 18-2 shows a process that handles an error using a an event subprocess.

Figure 18-2 Event Subprocess with a Start Error Event



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 window, 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 How to Configure an Error Event to Catch Business Exceptions.

For information on how to configure the implementation properties to catch system exceptions, see How to Configure a Catch Event to Catch System Exceptions.

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.

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 window, 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 How to Configure an Error Event to Catch Business Exceptions.

   For information on how to configure the implementation properties to catch system exceptions, see How to Configure a Catch Event to Catch System Exceptions.

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.

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. In the Type section, select By Error.

b. From the By Error list, select Browse.

   The Browse Types dialog box appears.

c. Deselect the Show System Faults option.

d. Enter the name of the exception or select it from the tree.

e. Optionally, you can configure the error event to recover from the exception, to do this select the Recoverable Error option.

   To recover from an error, you must set a value to the predefined variable action. For more information, see Configuring Catch Events to Recover from an Exception.

f. Click **OK**.

   The Browse Types dialog box closes and the selected exception appears in the **Exception** field.

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. In the Type section, select By Error.

   b. From the By Error list, select Browse.

      The Browse Types dialog box appears.

   c. 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 18-1.

   d. Enter the name of the exception or select it from the tree.

   e. Optionally, you can configure the error event to recover from the exception, to do this select the Recoverable Error option.

      To recover from an error, you must set a value to the predefined variable action. For more information, see Configuring Catch Events to Recover from an Exception.

   f. Click **OK**.

      The Browse Types dialog box closes and the selected exception appears in the **Exception** field.

Configuring Catch Events to Recover from an Exception

When an exception occurs while running a process flow, you can choose to retry running the flow object that caused that process flow or to move the process instance to the next flow object in the main process flow.

You can define this by setting a specific value to the predefined variable action. You can set the value of the predefined variable action in the following ways:

- Using data associations

  For more information, see Introduction to Data Associations.

- Using BPM scripts

  For more information, see Writing BPM Scripts.

The available values for the action predefined variable are:

- OK

  This is the default value for the action variable. When the action variable has this value, if an exception occurs then the process instance moves to the next flow object in the exception handling flow. The process flow that was running when the exception occurred is canceled. The faulted scope will be canceled. Using this action can be thought as a try-catch where the activity is in the try block and the handler in the catch block.

  To assign this value to the action predefined variable use the string “send".

- BACK

  When the action variable has this value, if an exception occurs then the process instance moves back to the flow object where the exception occurred to retry running the failed flow object. In some cases the circumstances that caused the exception might change so when you retry running the flow object it succeeds. For example, you try establish a phone call and the line is busy so you try again after a few minutes.

  To assign this value to the action predefined variable use the string “back".

- SKIP:

  When the action variable has this value, if an exception occurs then the process instance moves to the next activity in the main process flow.

  To assign this value to the action predefined variable use the string “skip".

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.

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 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 Components window, 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 Browse 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 18-1.

11. Enter the name of the exception or select it from the tree.
12. Click **OK**.

    The Browse Types dialog box closes and the selected exception appears in the Exception field.

13. Optionally, select the Force Commit After Execution option if you want the BPM runtime to add a checkpoint with the information from running the error end event.
14. Click **OK**.

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.

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. In the Applications window, right-click a module in the Business Components node.

   If the Business Component node does not contain a module, then you must create one.

2. Select **New**.

3. Select **Exception**.

   The Create Business Exception dialog box appears.

4. Enter a name to identify the exception.

5. Click the Browse button next to the Destination Module text field.

   The Browse Module dialog box appears.

6. Enter the name of the exception or select it from the tree.

7. Click **OK**.

   The Business Exception Editor opens.

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.

Handling Exceptions in Subprocesses

Event subprocesses enable you to define a cleaner process with less effort because the catch error event is located within the event subprocess.

You can handle exceptions that occur in a subprocess in the same way you handle the exceptions in any other BPMN activity.

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 inter process 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.

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 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 Type list, select **Error**.
8. Click the Exception list and select **Browse**, to browse the available business exceptions.

   The Browse Types dialog box appears.
9. Select an exception.

10. Click **OK**.

    The select exception appears in the Exception list.

11. Click OK.

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.

19

Using Fault Handling in BPM

This chapter describes how to use the SOA fault management framework with Oracle BPM.

This chapter includes the following sections:

- Handling Faults with the Fault Management Framework
- Designing Fault Policies for Oracle BPM Suite

Handling Faults with the Fault Management Framework

Oracle SOA Suite provides a generic fault management framework for handling faults. If a fault occurs during runtime, the framework catches the fault and performs a user-specified action defined in a fault policy file. If a fault results in a condition in which human intervention is the prescribed action, you perform recovery actions from Oracle Enterprise Manager Fusion Middleware Control. You can also use the fault management framework with Oracle BPM Suite.

For more information about the fault management framework, see Section "Handling Faults with the Fault Management Framework" of *Developing SOA Applications with Oracle SOA Suite*.

Designing Fault Policies for Oracle BPM Suite

You can design and execute fault policies for Oracle BPM Suite. The fault policies file defines fault conditions and their corresponding fault recovery actions.

The fault policy bindings file associates the policies defined in the fault policies file with one of the following:

- Composite with a BPMN process
- Oracle BPMN process service component
- Reference binding component (for example, another BPMN process or a JCA adapter)

The following fault recovery actions are supported in the fault policies file for Oracle BPM Suite:

- Retry
- Human intervention
- Terminate
- Java code
- Replay scope

- Rethrow fault

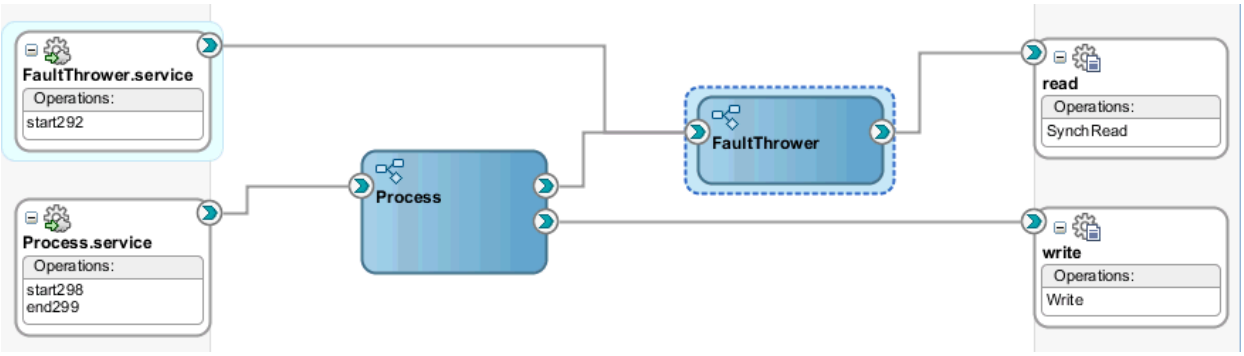
Note:

For more information about the supported fault recovery actions, see "Creating a Fault Policy File for Automated Fault Recovery" in *Developing SOA Applications with Oracle SOA Suite*.

Designing Composite Level Fault Policies

Figure 19-1 shows a composite that includes BPMN process service components named **Process** and **FaultThrower**.

Figure 19-1 Composite with Process and FaultThrower BPMN Process Service Components



Example 19-1 shows the fault conditions and their corresponding fault recovery actions defined in the `fault-policies.xml` file. A condition catches faults and transfers them to actions for recovering from the faults during runtime.

Example 19-2 shows the fault policy bindings file that associates the fault recovery policies defined in the `fault-policies.xml` file with the entire composite.

Example 19-1 Fault Policies File

```
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicies xmlns="http://schemas.oracle.com/bpel/faultpolicy"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <faultPolicy version="0.0.1" id="ravi_faultPolicy"
                 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
                 xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 xmlns="http://schemas.oracle.com/bpel/faultpolicy"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Conditions>
            <!-- catches all -->
            <faultName>
                <condition>
                   <action ref="Action-Retry"/>
                </condition>
            </faultName>
        </Conditions>
        <Actions>
            <Action id="Action-Abort">
                <abort/>
            </Action>
            <Action id="Action-Retry">
                  <retry>
                    <retryCount>3</retryCount>
```

```
                    <retryInterval>10</retryInterval>
                    <exponentialBackoff/>
                    <retryFailureAction ref="Action-human-intervention"/>
                </retry>
            </Action>
            <Action id="Action-human-intervention">
                <humanIntervention/>
            </Action>
        </Actions>
    </faultPolicy>
</faultPolicies>
```

Example 19-2 Fault Policy Bindings File

```
<?xml version="1.0" encoding="UTF-8" ?>
<faultPolicyBindings version="0.0.1"
 xmlns="http://schemas.oracle.com/bpel/faultpolicy"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <composite faultPolicy="ravi_faultPolicy"/>
</faultPolicyBindings>
```

Designing Service Component Level Fault Policies

Figure 19-2 shows a composite in which a BPMN service component named **BPMNThatFails** invokes a reference binding component (for this example, a file adapter).

Figure 19-2 Composite with BPMN Process Service Component

Example 19-3 shows the fault conditions and their corresponding fault recovery actions defined in the `fault-policies.xml` file. A condition catches faults and transfers them to actions for recovering from the faults during runtime.

Example 19-4 shows the fault policy bindings file that associates the fault recovery policies defined in the `fault-policies.xml` file with the `BPMNThatFails` BPMN service component.

Example 19-3 Fault Policies File

```
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicies xmlns="http://schemas.oracle.com/bpel/faultpolicy"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <faultPolicy version="0.0.1" id="FailsTrulyPolicy"
                 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
                 xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 xmlns="http://schemas.oracle.com/bpel/faultpolicy"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Conditions>
        <!-- catches all -->
        <faultName>
            <condition>
               <action ref="Action-human-intervention"/>
            </condition>
        </faultName>
```

```
    </Conditions>
    <Actions>
        <Action id="Action-Abort">
            <abort/>
        </Action>
        <Action id="Action-Retry">
              <retry>
                <retryCount>3</retryCount>
                <retryInterval>2</retryInterval>
                <exponentialBackoff/>
            </retry>
        </Action>
        <Action id="Action-human-intervention">
            <humanIntervention/>
        </Action>
    </Actions>
  </faultPolicy>
</faultPolicies>
```

Example 19-4 Fault Policy Bindings File

```
<?xml version="1.0" encoding="UTF-8" ?>
<faultPolicyBindings version="0.0.1"
 xmlns="http://schemas.oracle.com/bpel/faultpolicy"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <component faultPolicy="FailsTrulyPolicy">
        <name>BPMNThatFails</name>
    </component>
</faultPolicyBindings>
```

Designing Reference Level Fault Policies (Calling a BPM Process)

Figure 19-3 shows a composite in which a BPMN process named **Caller** invokes another BPMN process named **ProcessService1** as a reference.

Figure 19-3 Composite with Caller BPM Process

Example 19-5 shows the fault conditions and their corresponding fault recovery actions defined in the `fault-policies.xml` file. A condition catches all faults and transfers them to an action that requires human intervention to recover from the fault.

Example 19-6 shows the `fault-bindings.xml` file that associates the fault policies defined in `fault-policies.xml` with the reference.

When a process is called using a service reference, the reference used is not the BPMN process reference, but rather the reference created to call the BPMN process (`ProcessService1` in Figure 19-3) named `Services.Externals.ProcessService1.reference`. The reference name is created as follows:

- `Services.Externals.` is prefixed to the reference name of `ProcessService1`.
- `.reference` is appended to the reference name of `ProcessService1`.

You can obtain the reference name to specify in the `fault-bindings.xml` file from either of the following files:

- From the `reference` section of the *`process_name`*`.componentType` file, as shown in Example 19-7.
- From the `wire` section of the `composite.xml` file, as shown in Example 19-8.

This reference calls a BPMN process, but it can also invoke an FTP server, an EJB component, or something else.

Example 19-5 Fault Policies File

```
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicies xmlns="http://schemas.oracle.com/bpel/faultpolicy"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <faultPolicy version="0.0.1" id="Policy0"
                 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
                 xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 xmlns="http://schemas.oracle.com/bpel/faultpolicy"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Conditions>
            <!-- catches all -->
            <faultName>
                <condition>
                  <action ref="Action-human-intervention"/>
                </condition>
            </faultName>
        </Conditions>
        <Actions>
            <Action id="Action-human-intervention">
                <humanIntervention/>
            </Action>
        </Actions>
     </faultPolicy>
</faultPolicies>
```

Example 19-6 Fault Policy Bindings File

```
<faultPolicyBindings version="0.0.1"
 xmlns="http://schemas.oracle.com/bpel/faultpolicy"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <!-- reference ProcessService1 calls BPMN component Process -->
    <reference faultPolicy="Policy0">
        <name>Services.Externals.ProcessService1.reference</name>
    </reference>
</faultPolicyBindings>
```

Example 19-7 Reference Name in componentType File

```
<componentType
. . .
. . .
  <reference name="Services.Externals.ProcessService1.reference"
  . . .
  . . .
  </reference>
</componentType>
```

Example 19-8 Reference Name in the composite.xml File

```
<wire>
<source.uri>Caller/Services.Externals.ProcessService1.reference
 </source.uri>
. . .
</wire>
```

Designing Reference Level Fault Policies (Calling a File Adapter)

Figure 19-4 shows a composite in which a BPMN process named **Caller** invokes a file adapter named **FolderListing** as a reference.

Figure 19-4 Composite with Caller BPM Process

Example 19-9 shows the fault conditions and their corresponding fault recovery actions defined in the `fault-policies.xml` file. A condition catches all faults and transfers them to an action that requires human intervention to recover from the fault.

Example 19-10 shows the `fault-bindings.xml` file that associates the fault policies defined in `fault-policies.xml` with the reference.

Example 19-9 Fault Policies File

```
<?xml version="1.0" encoding="UTF-8"?>
<faultPolicies xmlns="http://schemas.oracle.com/bpel/faultpolicy"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <faultPolicy version="0.0.1" id="Policy0"
                 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
                 xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 xmlns="http://schemas.oracle.com/bpel/faultpolicy"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Conditions>
            <!-- catches all -->
            <faultName>
                <condition>
                  <action ref="Action-human-intervention"/>
                </condition>
            </faultName>
        </Conditions>
        <Actions>
            <Action id="Action-human-intervention">
                <humanIntervention/>
            </Action>
        </Actions>
     </faultPolicy>
</faultPolicies>
```

Example 19-10 Fault Policy Bindings File

```
<faultPolicyBindings version="0.0.1"
 xmlns="http://schemas.oracle.com/bpel/faultpolicy"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

   <!-- reference FolderListing is a reference to a file adapter -->
   <reference faultPolicy="Policy0">
```

```
        <name>Services.Externals.FolderListing.reference</name>
    </reference>
</faultPolicyBindings>
```

What You May Need to Know About the Difference Between Reference Naming Conventions in Oracle SOA Suite and Oracle BPM Suite

For external services, the reference name used in the fault policy bindings file is different between Oracle SOA Suite and Oracle BPM Suite. Table 19-1 provides details.

Table 19-1 Naming Conventions for References in the .componentType and Fault Policy Binding Files

Reference Naming Convention	Entry in .componentType File	Entry in Fault Policy Bindings File

For Oracle BPM Suite, the naming convention is as follows:
• `Services.Externals.` is prefixed to the reference name.
• `.reference` is appended to the reference name.
	The naming convention in the *`BPMN_process_name`*`.componentType` file for a reference named `Synch_read` is as follows:
	`<reference name="Services.Externals.Synch_read.reference"`
	
	`ui:wsdlLocation="synch_read.wsdl">`
	`<interface.wsdl interface="http://xmlns.oracle.com/pcbpel/adapter/file/UpdateTaskApps/FaultPolicyApp/synch_read#`
	
	`wsdl.interface(SynchRead_ptt)"/>`
	`</reference>`
		For Oracle BPM Suite, specify `Services.Externals.Synch_read.reference`:
		`<reference faultPolicy="Policy0">`
		`<name>Services.Externals.Synch_read.reference</name>`
		
		For more information, see Example 19-6 and Example 19-10.

For Oracle SOA Suite, no names are prefixed or appended to the reference name.	The naming convention in the *`BPEL_process_name`*`.componentType` file for a reference named `Synch_read` is as follows:
	`<reference name="Synch_read"`
	
	`ui:wsdlLocation="synch_read.wsdl">`
	`<interface.wsdl interface="http://xmlns.oracle.com/pcbpel/adapter/file/UpdateTaskApps/FaultPolicyApp/synch_read#`
	
	`wsdl.interface(SynchRead_ptt)"/>`
	`</reference>`
		For Oracle SOA Suite, specify `Synch_read`:
		`<reference faultPolicy="Policy0">`
		`<name>Synch_read</name>`

Note:

You can also define a single fault policy bindings file to catch faults from both Oracle BPM Suite and Oracle SOA Suite.

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:

- Introduction to Communication with Other BPMN Processes and Services
- Communicating With Other BPMN Processes and Services Using Message Events
- Using Message Events to Invoke Asynchronous Services and Asynchronous BPMN Processes
- Using Message Events Configured as Boundary Events
- Using Service Tasks to Invoke Synchronous Operations in Services and BPMN Processes
- Communicating With Other BPMN Processes and Services Using Send and Receive Tasks
- Using Send and Receive Tasks to Invoke Asynchronous Services and Asynchronous BPMN Processes
- Introduction to Invoking a Process Using Call Activities
- Invoking a Process Using Call Activities
- Introduction to Communication Between Processes Using Signal Events
- Communicating Between Processes Using Signal Events

Introduction to Communication with Other BPMN Processes and Services

Oracle BPM provides multiple ways for BPMN processes to communicate with other processes or services, such as messages,send and receive tasks, and signal events.

- **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 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 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 Communicating Between Processes Using Signal Events, for more information about signal events.

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 a 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.

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.

You can configure message start events in a subprocess to suspend the main process flow. For more information, see Suspending the Current Process Flow to Run an Alternative Process Flow.

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 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 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

Table 20-1 (Cont.) Message Event Implementation

Event	Initiates Conversation	Continues Conversation
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.

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

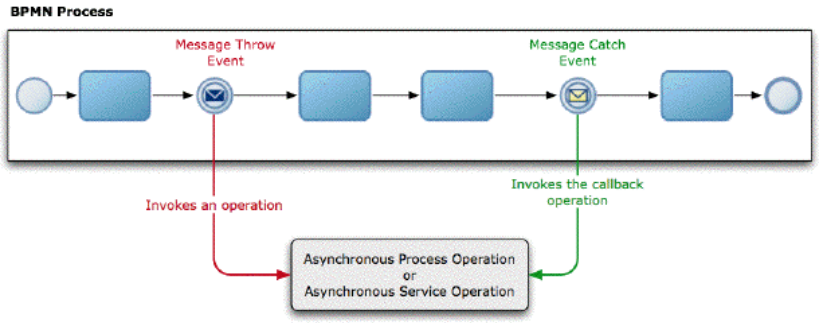
Then it sends the XML message to the service or BPMN process, and continues running the rest of the process flow. It does not wait for the asynchronous service or BPM process to answer.

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



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 Message Exchange section, click **Browse**.

   The Conversation dialog box appears.
8. Click the **New** button, next to the Search field.

   The Create Conversation dialog box appears.
9. From the **Type** list, select **Service Call**.

   You must ensure that the service you select is an asynchronous service.
10. In the Definition section, click the **Browse** button next to the Service field.

    The Service dialog box appears.
11. Select the asynchronous service you want to invoke.
12. Click **OK**.
13. From the Operation list, select the operation to invoke from the asynchronous service.
14. If the asynchronous service requires arguments, configure the message throw event data association.

See Introduction to Data Associations, for more information on how to configure data associations.

15. Click **OK**.
16. Follow the procedure described in How to Receive the Callback Operation of an Asynchronous Service Using Message Events to invoke the callback operation of the asynchronous process.

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 Message Exchange section, click **Browse**.

   The Conversation dialog box appears.
8. Select the conversation you want to use.
9. Click **OK**.

   The Message Exchange section changes.
10. From the Target Node list, select the callback operation.
11. If the asynchronous service requires arguments, configure the message throw event data association.

    See Introduction to Data Associations, for more information on how to configure data associations.
12. Click **OK**.

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.

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 Message Exchange section, select **New**.

   The Conversation dialog box appears.
8. Click the **New** button.
9. From the **Type** list, select **Process Call**.
10. Click the **Browse** button next to the Process field.

    The Type dialog box appears.
11. Select the asynchronous BPMN process you want to invoke.
12. Click **OK**.
13. From the **Target Conversation** list, select the conversation from the asynchronous BPMN process.
14. If the asynchronous BPMN process requires arguments, configure the message throw event data association.

    See Introduction to Data Associations, for more information on how to configure data associations.
15. Click **OK**.
16. Follow the procedure described in How to Invoke the Callback Operation of an Asynchronous BPMN Process Using Message Events to invoke the callback operation of the asynchronous process.

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. Click the **Browse** button, next to the Conversation field.

   The Conversation dialog box appears.
8. Select the conversation you want to use.
9. Click **OK**.

   The Message Exchange section changes.
10. From the Target Conversation list, select the conversation in your process to invoke the asynchronous process.
11. Click **OK**.
12. If the asynchronous BPMN process requires arguments, configure the message throw event data association.

    See Introduction to Data Associations, for more information on how to configure data associations.
13. Click **OK**.

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.

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.

You can configure boundary events to suspend the flow object or subprocess they are associated to. For more information, see Suspending the Current Process Flow to Run an Alternative Process Flow.

Using Service Tasks to Invoke Synchronous Operations in Services and BPMN Processes

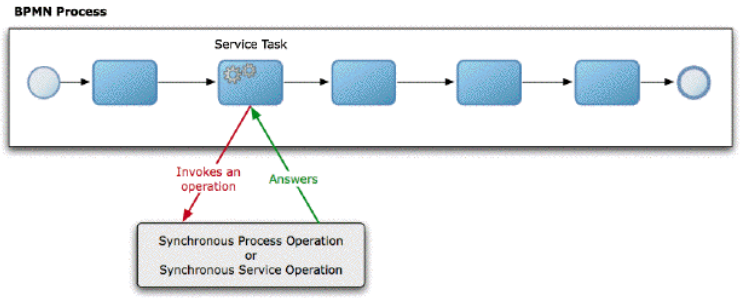
Service tasks enable you to invoke synchronous operations in services and BPMN processes.

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 Using Message Events to Define a Synchronous Operation in a BPMN Processes Interface or 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



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 Message Exchange section, click **Browse**.

   The Conversation dialog box appears.
8. Click the **New** button.

   The Create Conversation dialog box appears.
9. From the **Type** list, select **Service Call**.
10. Select the synchronous service you want to invoke.
11. Click **OK**.
12. Click the **Browse** button next to the Service field and select the service you want to use.
13. 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 Introduction to Data Associations, for more information on how to configure data associations.
14. Click **OK**.

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.

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 Message Exchange section, click **Browse**.

   The Conversation dialog box appears.
8. Click the **New** button.

   The Create Conversation dialog box appears.

   Note:

   Service tasks only support outbound conversations.

   ---
9. From the **Type** list, select **Process Call**.
10. Click the **Browse** button next to the Process field to select the process to invoke.
11. From the **Target Conversation** list, select the flow object 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 Introduction to Data Associations, for more information on how to configure data associations.
13. Click **OK**.

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.

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

Table 20-2 (Cont.) Send and Receive Tasks Implementation

Task	Initiates Conversation	Continues Conversation
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

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:

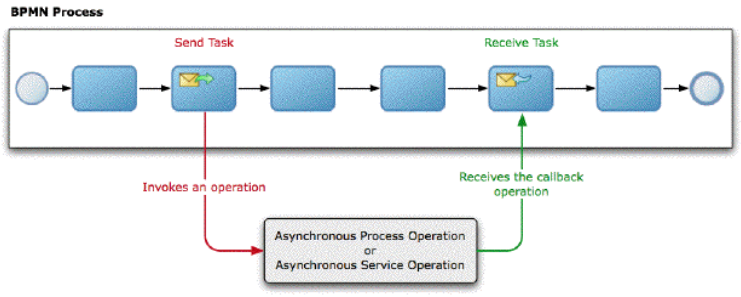
- the asynchronous operation
- the input required by the asynchronous operation
- the data association defined for the message throw event

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



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 Message Exchange 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 box 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 Introduction to Data Associations, for more information on how to configure data associations.
14. Click **OK**.
15. Follow the procedure described in How to Use the Receive Task to Invoke the Callback Operation of an Asynchronous Service to invoke the callback operation of the asynchronous process.

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 Message Exchange 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 box appears.
10. Select the asynchronous service whose callback you want to invoke.
11. Click **OK**.

    The Type dialog box disappears and the receive task properties dialog box 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 Introduction to Data Associations, for more information on how to configure data associations.
14. Click **OK**.

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.

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 Message Exchange 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 box 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 Introduction to Data Associations, for more information on how to configure data associations.
14. Click **OK**.

15. Follow the procedure described in 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.

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 Message Exchange 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 box appears.
10. Select the asynchronous BPMN process whose callback you want to invoke.
11. Click **OK**.

    The Type dialog box disappears and the receive task properties dialog box 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 Introduction to Data Associations, for more information on how to configure data associations.
14. Click **OK**.

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.

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.

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.

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 How to Create a New Business Process.

6. If necessary, configure data associations or transformations.

   For more information on data associations, see Introduction to Data Associations.

   For more information on transformations, see Introduction to Transformations.

7. Click **OK**.

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 *Developing SOA Applications with Oracle SOA Suite*.

For information on how to access Event in Oracle BPM, see 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.

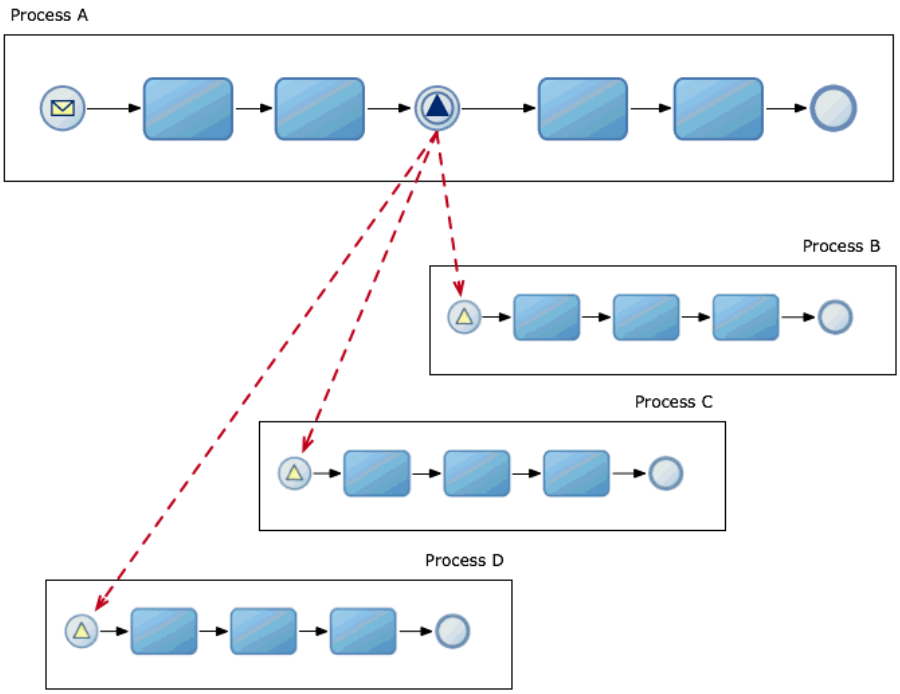
In a similar way, the process that reacts to a specific message has no information about the processes that broadcast that message. If you add a process that broadcast a message to your project, all the process waiting for that specific message react to it without you having to modify them.

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.

Note:

Use signal events to communicate with other processes. The process that broadcasts the signal event does not include itself in the broadcast list.

Figure 20-4 Signal Broadcast



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.

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.

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 box appears.
8. Select an event.
9. Click **OK**.

   The Type dialog box disappears and the type name appears in the type field.
10. Click **OK**.

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.

How to Configure Your Process to 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 box appears.
6. Select an event.
7. Click **OK**.

   The Type dialog box disappears and the type name appears in the type field.
8. Click **OK**.

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:

- Defining the Process Interface
- Using Message Events to Define the BPMN Process Interface
- Using Message Events to Define Asynchronous Operations in a BPMN Processes
- Using Message Events to Define a Synchronous Operation in a BPMN Processes Interface
- Using Message Events with an Interface from the Business Catalog to Define Your Process Interface
- Defining the BPMN Process Interface Using Send and Receive Tasks
- Defining Asynchronous Processes Operations Using Send and Receive Tasks
- Using Send and Receive Tasks to Define a Synchronous Operation in a BPMN Process
- Using Send and Receive Tasks with an Interface from the Business Catalog to Define Your Process Interface
- Defining the Process Input and Output

This chapter assumes that you are familiar with SOA Composites. For more information about SOA Composites see *Developing SOA Applications with Oracle SOA Suite*.

Defining the Process Interface

The process interface is a group of operations that 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.

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.

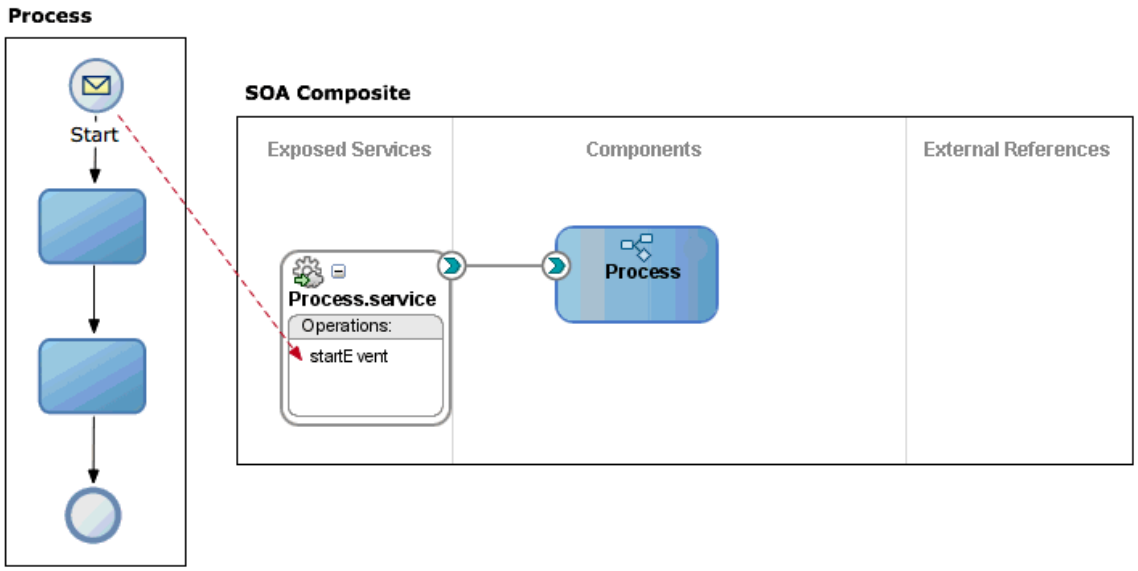
The process interface must always contain an operation that exposes the start event of a BPMN process. A process or service that invokes this BPMN process must always invoke the operation that corresponds to the start event before invoking any of the operations in the process.

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 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

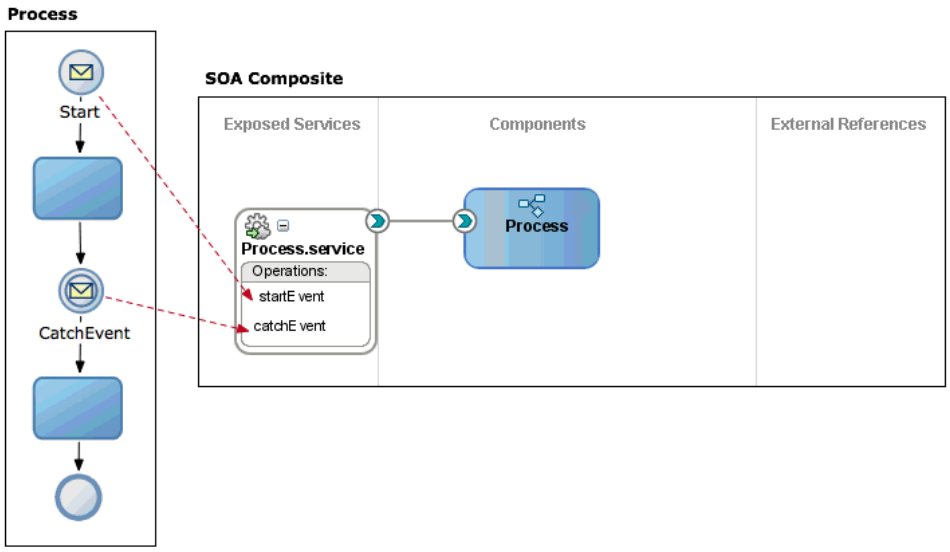


In addition, the process interface may contain the operations exposed by the catch message events in the process. Before invoking an operation that corresponds to a

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



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.

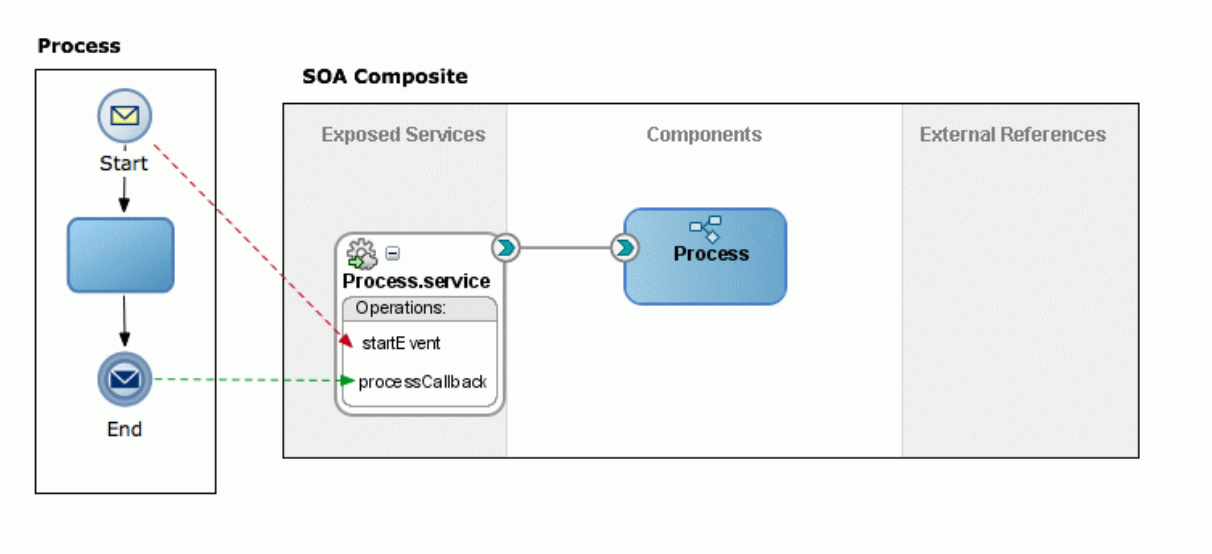
See Using Message Events to Invoke Asynchronous Services and Asynchronous BPMN Processes, for information on how invoke an asynchronous BPMN process from another BPMN process.

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 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



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.

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 Message Exchange 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 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 How to Define a Callback Operation Using Message Events, to define the callback operation of the asynchronous BPMN process.

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 Message Exchange 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 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**.

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 Using Message Events to Invoke Asynchronous Services and Asynchronous BPMN Processes and 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.

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 Message Exchange 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 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 How to Define a Callback Operation Using Message Events, to define the callback operation for this asynchronous operation.

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.

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 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 Handling Errors in a Peer Process Using Message Events.

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 Message Exchange 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 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 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.

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 Message Exchange 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 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 Introduction to Data Associations.

8. Click **OK**.

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 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.

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.

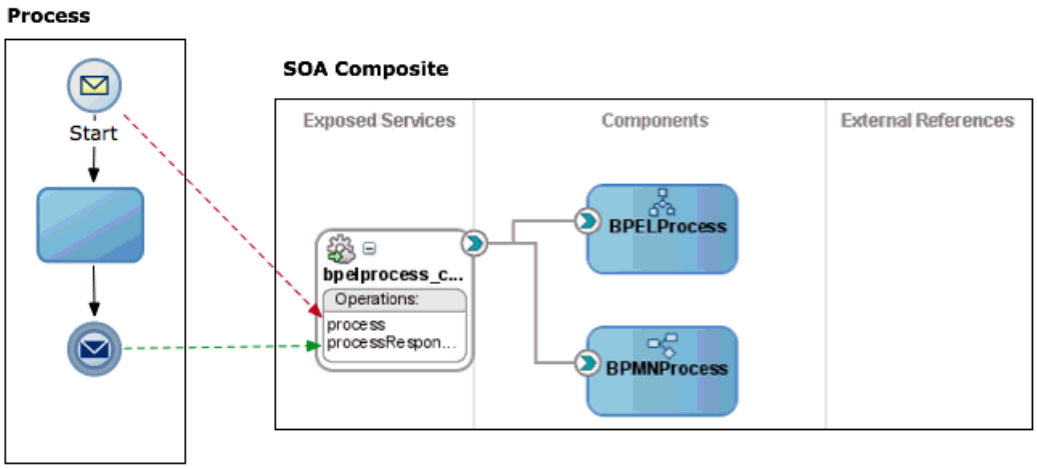
If you define a message start or a message catch event using an interface from the business catalog, then the associated message throw or message end event must also use an interface from the business catalog. If the operation you are defining is asynchronous, then the message throw or message end events 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



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 Message Exchange 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 box 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 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 How to Configure a Message End or a Message Throw Event to Use an Interface from the Business Catalog Using Message Events.

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 Message Exchange 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 box 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 Introduction to Data Associations, for more information on how to configure data associations.

12. Click OK.

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.

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 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.

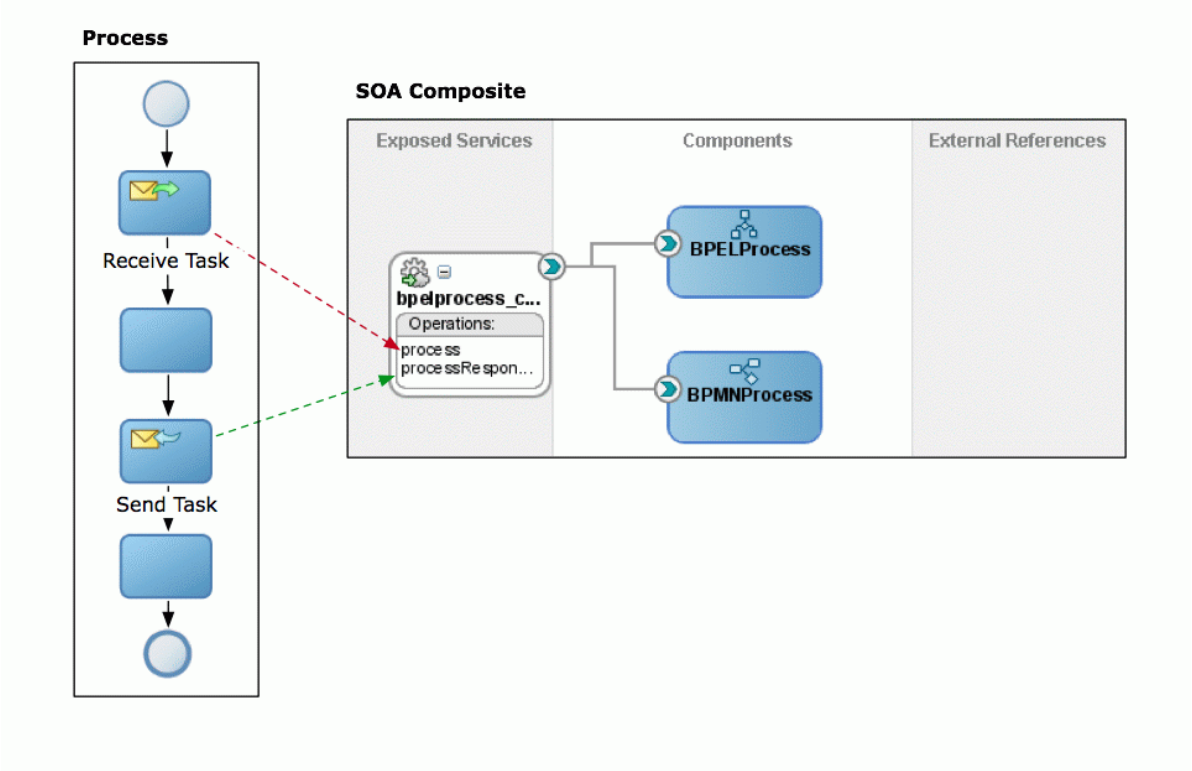
To define the process output, you must configure the send task that continues the receive that defines the operation. If the operation is asynchronous, then the send task also defines the callback operation.

If an interface contains an asynchronous operation, then it must also define the callback operation that returns the result of this operation. See 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



If you used a send task to expose an operation, then you must use a receive task to define the callback operation. See Defining the Callback Interface for BPMN Processes Using a Send Task for more information on how to define a callback operation using send events.

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.

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.

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 Message Exchange 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 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 How to Define a Callback Process Operation Using a Send Task, to define the callback operation of the asynchronous BPMN process.

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 Message Exchange 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 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**.

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 Message Exchange 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 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**.

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 Using Message Events to Invoke Asynchronous Services and Asynchronous BPMN Processes and Using Send and Receive Tasks to Invoke Asynchronous Services and Asynchronous BPMN Processes, for more information on how to invoke an asynchronous BPMN process.

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 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.

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 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**.

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 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.

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.

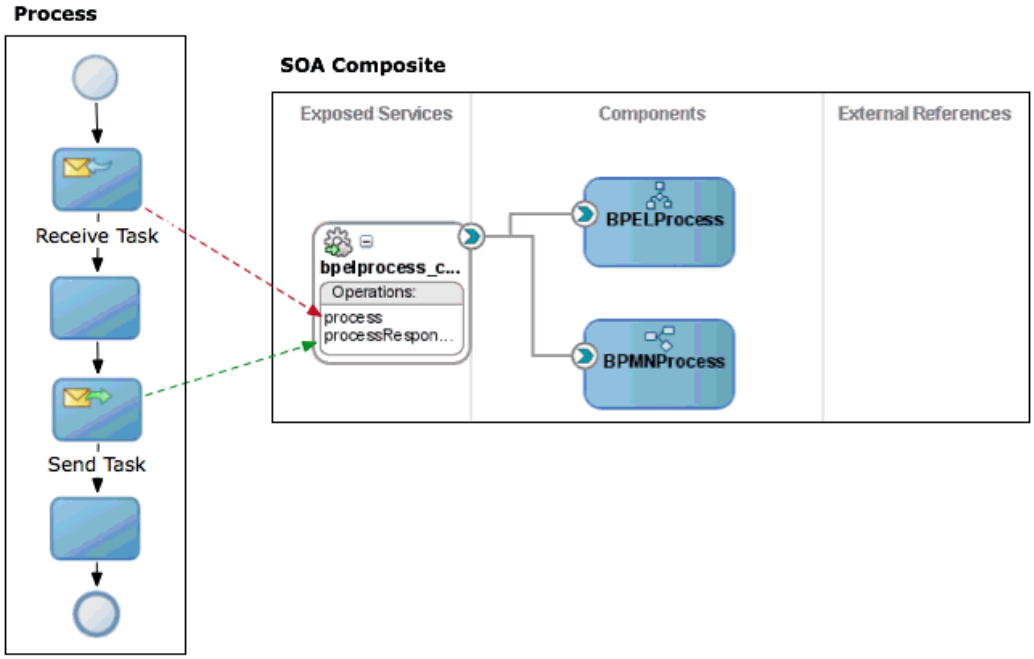
You can choose any of the operations from the components in the business catalog and use it as the interface for your process operations.

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



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 Message Exchange 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 box 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 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 How to Configure a Message End or a Message Throw Event to Use an Interface from the Business Catalog Using Message Events.

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 Message Exchange 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 box 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 Introduction to Data Associations, for more information on how to configure data associations.
12. Click **OK**.

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 from 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.

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.

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 box appears.

2. Enter a name to identify the argument.
3. Click the **Browse More Types** Button.

   The Browse Type dialog box 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 box 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.

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 box appears.

3. Change the name of the type.
4. Click **OK**.

The argument in the Argument Definition table shows the updated name and type.

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

Communicating Business Processes Using Correlations

This chapter describes how to develop a BPMN process that communicates with other BPMN processes and services using correlations. Correlations are used to identify the instance that receives the message in the peer process.

This chapter includes the following sections:

- Introduction to Correlations
- Understanding the Components of a Correlation
- Typical Design Workflow
- Defining Correlations for a BPMN Element
- Creating Correlations Keys

Introduction to Correlations

Correlations enable business processes to communicate with each other based on the state of an instance. The state of all the process data objects in a process defines the state of the instance.

Defining a correlation for a business process enables you to identify an instance in another process through the instance state and send a message to that specific instance.

For example you can use correlations to communicate a sales process with the corresponding shipping and mailing processes. When the customer confirms an order, the shipping process sends a message to the shipping and mailing processes using a correlation that defines that it uses the order ID to locate the instances in both processes.

After you initialize a correlation you cannot change its value because the Service Engine uses this value to locate the instance. If you try to assign a new value to the correlation this produces a Correlation ViolationError.

You can define and initialize multiple correlations for a flow object. The flow object that sends the message can use just one correlation or all the correlations defined for that flow object. If it uses all the existing correlations, then all of the values it sends together with the message must lead to the identification of the same instance.

Some flow objects, like the service task, define two types of correlations: input and output. In those cases you can initialize and use a correlation in the same activity.

The scope of the correlation is the instance of the process or subprocess where it is defined. In the case of subprocess with multi-instance loop conditions the scope of the correlation is each instance of the multi-instance subprocess.

Note:

Use correlations to communicate with a subprocess in a single flow. If the flow is parallel, then you must use conversations. For more information about conversations, see Defining Conversations.

Understanding the Components of a Correlation

Components of a correlation include a definition, keys, property, and property alias.

The following list describes the different components of a correlation:

- Correlation Definition

  Contains the set of correlation keys defined for a flow object.

- Correlation Keys

  Define the properties to use in the correlation. When you define a correlation key you provide a name to identify it. The scope of the correlation key is the project, which means that after you define a correlation key you can use it for the correlation definition of any flow object in that project.

  If your BPM project contains BPEL processes, then the correlation keys defined in that BPEL process automatically appear for you to reuse them in your BPM Projects.

- Correlation Property

  Properties are abstractions for very representative attributes in the process, like the order ID, the customer name or the social security number. Properties contain a name to identify the attribute and a data type. Properties only support basic data types.

- Correlation Property Alias

  Enable you to define how to assign a value to the correlation property using expressions. You can use the arguments and predefined variables of the activity to assign values to the correlation property alias.

Typical Design Workflow

This workflow describes the typical procedures you perform when you design a project that contains business processes that communicate with each other using correlations.

1. Design the processes that communicate with each other.
2. In the calling process, add the flow object that sends a message to the other process.
3. Define a correlation for the flow object that sends the message and configure it to initiate the property aliases.

   For more information on how to define the correlation, see Defining Correlations for a BPMN Element.
4. In the invoked process, add the flow object that receives the message.
5. Configure the flow object that receives the message to use the correlation you defined and assign a value to the property aliases.

For more information on how to define the correlation, see Defining Correlations for a BPMN Element.

Defining Correlations for a BPMN Element

You can define multiple correlations for a single flow object. The flow object that sends the message can choose to use just one correlation or use all of them. In the latter it the values it uses to invoke the correlation lead to the identification of the same instance.

You can define correlations for the following BPMN flow objects that you use to communicate business processes:

- Message Events
- Send and Receive Tasks
- Signal Events

Note:

The use of a multicast subscription impacts all message based correlations. If you define a multi-cast subscription for Oracle EDN, then all message based correlations are multi-cast.

How to Define a Correlation for a Flow Object

You can define a correlation while you are defining the properties of a flow object. Studio provides two modes for defining correlations: simple and advanced. To define a correlation that contains just one property use the simple mode. If the correlation you define contains more that one correlation key each with multiple properties, then use advanced mode.

To define a correlation for a flow object:

1. Right-click the BPMN element.

   The Properties dialog box appears.

2. Click the Implementation tab.
3. Click the Correlations link.

   The Correlation Definition dialog box appears.

4. Define the correlation using one of the following modes:

   - Simple Mode: Follow the procedure described in How to Define a Correlation Using Simple Mode.
   - Advanced Mode: Follow the procedure described in How to Define a Correlation Using Advanced Mode.

How to Define a Correlation Using Simple Mode

Simple mode allows you to define a correlation that contains just one property. This mode simplifies the definition of the correlation by creating parts of the correlation automatically based on the information you define for that property.

To define a correlation using simple mode:

1. From the Property list, select a property.

   If the Property list is empty, create a new property:

   a. Click the New button next to the Property list.

   b. Enter a name.

   c. Select a type.

      Available types are: string, int, double, decimal, boolean, time.

2. If the BPMN element initiates the value of the correlation, select Initiates.

3. In the Correlation Property Alias text box, define an expression to assign values to the correlation property.

   To define a complex expression, click the Expression Builder button next to the Correlation Property Alias text box. For more information see Writing Expressions.

4. Click OK.

How to Define a Correlation Using Advanced Mode

Advanced mode enables you to define multiple correlation keys with multiple correlation properties.

To define a correlation using advanced mode:

1. In the Correlation Definition dialog box, click **Switch to Advanced Mode**.

   The Correlation Keys table appears.

2. For each of the correlation keys you want to add to the correlation:

   a. Click the **Add** button.

      The Create CorrelationKey dialog box appears.

   b. Select an existing correlation key or click the **New Correlation Key** button to create a new correlation key.

      For more information on how to define a new correlation key, see How to Configure a Correlation Key.

   c. If the BPMN element initiates the value of the correlation, select **Initiates**.

   d. Click **OK**.

3. In the Correlation Property Aliases section, for each of the listed properties, define an expression to assign a value to the correlation property.

   To define a complex expression, click the Expression Builder button next to the property text box. For more information see Writing Expressions.

4. Click **OK**.

   The Correlation Definition dialog box closes and the Correlations icon appear in colors now.

Creating Correlations Keys

Since you define correlation keys at project level you can reuse correlation keys across the processes in your project. You can also decide to create correlation keys before adding the flow objects that use them. For both of these cases you must create correlation keys from the Structure view.

Note:

Updates to a correlation key made using the runtime interface do not change the existing subscription, even though the GUI may reflect the change. The change is applicable to subsequent events in the flow.

How to Create a Correlation Key

You can create a correlation key at a project level and later use that correlation key to define the correlations of your flow objects.

To create a correlation key:

1. Select a business process.
2. Open the **Structure** view.
3. Expand the **Correlations** node in the **Structure** view.
4. Right-click the **Correlations Keys** node.
5. Select **New**.

   The Create Correlation Key dialog box appears.

How to Configure a Correlation Key

You can configure the properties that compose the correlation keys you define.

To configure a correlation key:

1. Enter the correlation name.
2. If the Correlation Properties list is empty or the property that you want to use for your correlation is not defined, you must define a new correlation property. To define a correlation property:

   a. Click the **New** button.

      The Create Correlation Property dialog box appears.

   b. Enter a name for the correlation property.

   c. Select a type.

      Available types are: string, int, double, decimal, boolean, time.

   d. Click **OK**.

3. Select the correlation properties that define the correlation:

a. From the **Correlation Properties** list, select the property you want to include in the correlation.

b. Click the **Select** button.

   The selected property appear in the Selected list.

4. Click OK.

23

Defining Conversations

This chapter describes how to create and configure conversations in your BPM project and how to view and use a collaboration diagram.

This chapter includes the following sections:

- Introduction to Conversations
- Understanding the Different Types of Conversations
- Creating Conversations
- Defining Conversations for a BPMN Element
- Viewing the Collaboration Diagram

Introduction to Conversations

Conversations define the state of a collaboration between two participants. You must use them when a process needs to have multiple parallel conversations with different instances of the same process or service, for example within a multi-instance subprocess. Conversations group the message exchange between two or more processes. The message exchange between processes is called collaboration.

A process instance may need to communicate with different instances in another process. For example, a procurement process may need to interact with two different instance in a supplier process, each representing a different item. You can model this message exchange using conversations.

Within a process you can define multiple conversations that you can reuse among the flow objects in that process.

The members of the collaboration are called *participants*. The participants in a collaboration can be one of the following:

- BPMN processes
- BPEL processes
- Human Tasks
- Business Rules
- External References

Collaboration diagrams allows you to view the process flow together with the interactions your process has with other participants in the conversation.

Defining the Default Conversation

Your BPM project defines a conversation by default. If you do not want to define multiple conversations you must use this default conversation to gather all the message exchange among the processes in your project.

You can only define one default conversation per project. However you can modify your project to use a different default conversation than the one it uses by default. For more information on how to do this, see How to Create a Conversation.

Understanding the Different Types of Conversations

The different types of conversations allow you to specify the different types of interaction your process can establish with other processes or services.

The following list describes the different types of conversations:

- Define Interface: use this type to define the operations that other services and processes can invoke to interact with a BPMN process.
- Use Interface: use this type to configure your process to use an interface from a component in the Business Catalog.
- Process Call: use this type to invoke another BPMN process.
- Service Call: use this type to invoke a service defined in your BPM project.

For more information on how to communicate your process with other processes or services, see the following chapters:

- Communicating With Other BPMN Processes and Services
- Defining the Process Interface

Creating Conversations

You can create a conversation to model the message exchange between a process instance and the instances in another process or service.

Conversations enable you to group the message exchange between the processes in your BPM project.

How to Create a Conversation

To create a conversation:

1. In the **Applications** window, select a process from the project in which you want to define the correlation.
2. In the **Structure** window, right-click the **Conversations** node.
3. Select **New**.

   The Create Conversation dialog box appears.
4. Enter a name to identify the conversation.
5. If you want this conversation to be the default conversation for this project, select **Default Conversation**.

6. From the Type list select the type for this conversation.

   Available types are:

   - Define Interface
   - Use Interface
   - Process Call
   - Service Call

7. If you want to expose this conversation as a SOAP service, select **Expose as a SOAP Service**.

8. Click **OK**.

Defining Conversations for a BPMN Element

The BPMN elements you use to communicate a process with other processes or services require you to define a conversation. The defined conversation groups the messages exchanged between the processes and services within a BPM project.

The BPMN elements that require you to define a conversation are:

- Message Events
- Send and Receive Tasks
- Signal Events

The main reason to define a conversation is to have a process with a multi-instance activity with an external receive.

How to Define a Conversation for a BPMN Element

You can define a conversation for a BPMN element.

To define a conversation for a BPMN element:

1. Right-click the BPMN element.

   The Properties dialog box appears.

2. Click the **Implementation** tab.

3. Click the **Browse** button next to the Conversation field.

   The Conversation dialog box appears.

4. Select a conversation from the list.

   To search for a conversation, enter part of the name or the complete name in the **Search** field.

   To create a new conversation, click the **New** button. For more information on how to create a new conversation, see Creating Conversations.

5. Click **OK**.

   The **Message Exchange** section displays different fields according to the type of conversation you selected.

6. Configure the **Message Exchange** section.
7. Click **OK**.

What Happens When You Define a Conversation for a BPMN Element

You can define conversations at process or subprocess level, this determines the visibility scope of the conversation from a specific instance.

Viewing the Collaboration Diagram

The collaboration diagram shows the flow of your process and how that process interacts with other processes or services in the same diagram.

You can view the collaborations in your process using the collaboration diagram.

How to View the Collaboration Diagram

To view the collaboration diagram:

1. Open the BPMN process.
2. In the BPMN process editor, click the Collaboration tab located at the bottom next to the Designer tab.

   The collaboration diagram for the selected process appears.

How to Hide a Collaboration

You can hide a collaboration so that you are able to focus on the rest of the collaborations in the diagram.

To hide a collaboration:

1. In the **Collaboration Diagram**, right click over a participant of the conversation.
2. Select **Hide Conversation**.

   The selected participant disappears from the Collaboration Diagram.

How to Show a Collaboration

You can show a conversation that was previously hidden.

To show a collaboration:

1. In the **Collaboration Diagram**, right-click a participant of the conversation.
2. Select **Show Conversation**.

   The previously hidden conversation appears from the Collaboration Diagram.

24

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 expression 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:

- Introduction to Expressions in Oracle BPM
- Writing Conditions in Conditional Sequence Flows
- Writing Expressions in Complex Gateways
- Writing Expressions in Timer Events
- Writing Expressions in Data Associations
- Writing Conditions in Loop and Multi-Instance Markers in Subprocesses
- Writing Expressions and Conditions Using the Simple Expression Builder
- Simple Expression Builder Supported Operators
- Simple Expression Builder Supported Functions
- Writing Expressions Using the XPath Expression Builder
- Using Arrays
- Using Literals
- XPath BPM Extension Functions

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
- Correlations

The results of the expression vary according to the type of element you are configuring. Table 24-1 describes the expression required by each of the BPM elements.

Table 24-1 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 duration 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.

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.

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 **Properties** 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 choose to write it in the provided text area or launch the expression builder by clicking the Launch Expression Builder button next to the 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**.

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 24-1 shows a condition that configures the gateway to release the tokens that arrived to it after two tokens arrive to the merge gateway.

Example 24-1 Condition in a Complex Gateway

```
activationCount >= 2
```

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**.

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)`

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**.

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.

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. Click the **Data Associations** link.
5. Click the Expression button.

   The Expression Builder Dialog appears.

6. Type the expression and click OK.

   The expression appears in the middle column.

7. Locate the input or output argument you want to modify using an expression and drag it over the middle column over the expression you created.
8. Click **OK**.

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.

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:
   a. Select the expression language.

      Possible options are Simple or XPath.

   b. 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:
   a. Select **Loop Maximum**.
   b. 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**.

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:
   a. Select the expression language.

      Possible options are Simple or XPath.

   b. 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:
   a. Select the expression language.

      Possible options are Simple or XPath.

   b. 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**.

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

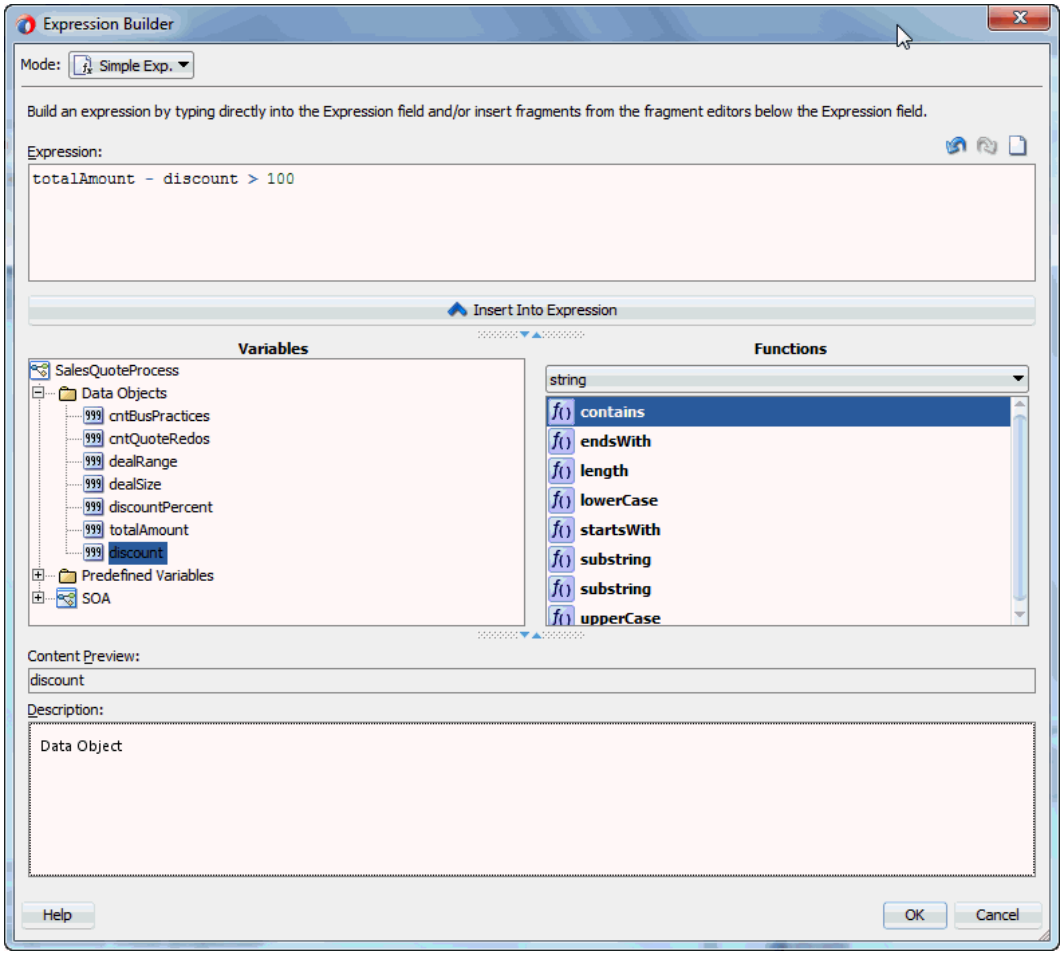
  If you wait a few seconds after you type the dot to invoke a method, then the Simple Expression Builder shows a list with the available functions that you can invoke over that data object. If you want the Simple Expression Builder to complete the expression for you, then you can press `Ctrl + Space`.

- 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 24-1 show the Simple Expression Builder dialog box.

Figure 24-1 Simple Expression Builder



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 **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.

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 **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.

Simple Expression Builder Supported Operators

The Simple Expression Builder enables you to create expressions using several 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 24-1, Table 24-2, Table 24-3, Table 24-4 and Table 24-5 describe the supported operators in the Simple Expression Builder.

Table 24-2 Arithmetic Operators

Operator	Name	Description
+	Addition	Adds numeric data types. Concatenates Strings. Add an interval value to a dateTime value.

Table 24-2 (Cont.) Arithmetic Operators

Operator	Name	Description
-	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 24-3 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 24-4 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 24-5 Conditional Operators

Table 24-5 (Cont.) Conditional Operators

Operator	Name	Description
and	Conditional And	Returns true if both operands evaluate to true.
or	Conditional Or	Returns true if one operand evaluates to true.

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

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:

- String Functions
- Numeric Functions
- DateTime and Duration Functions

String Functions

These functions enable you to manipulate string variables and literals, and perform calculations based on them.

length

Returns the number of characters in this string.

Signature:

```
int length(string stringToMeasure)
```

Arguments:

string - The string to measure.

Examples:

```
name.length()
length(name)
name.length
```

concatenation

Concatenates one or more Strings.

Examples:

```
name + " " + lastName
"Oracle " + "BPM"
```

contains

Returns true if the string contains the specified string.

Signature:

```
boolean contains(string mainString, string subString)
```

Arguments:

`subString` - The string to find.

Examples:

```
productName.contains("book")
contains(productName, "book")
```

startsWith

Returns true if the string starts with the specified string.

Signature:

```
boolean startsWith(string mainString, string subString)
```

Arguments:

`subString` - The string to find at the beginning of the string.

Example:

```
productId.startsWith("ABC")
startsWith(productId, "ABC")
```

Numeric Functions

These functions enable you to perform calculations using numeric data types. The available numeric data types are double, decimal, and int.

floor

Returns the largest int value that is smaller than the numeric value used for invoking this function. You can use this function with double and decimal data types.

Signature:

```
int floor(double number)
int floor(decimal number)
```

Arguments:

double / decimal - The number to use as a base for this function.

Examples:

```
number.floor()
floor(number)
number.floor
floor(totalAmount/3)
temperature.floor()
```

ceil

Returns the smallest int value that is greater than the numeric value used for invoking this function. You can use this function with double and decimal data types.

Signature:

```
int ceil(double number)
int ceil(decimal number)
```

Arguments:

double / decimal - The number to use as a base for this function.

Examples:

```
number.ceil()
ceil(number)
number.ceil
```

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 double and decimal data types.

Signature:

```
int round(double number)
int round(decimal number)
```

Arguments:

double / decimal - The number to use as a base for this function.

Examples:

```
number.round()
round(number)
number.round
```

abs

Returns the absolute value of this number. You can use this function with int, double, and decimal data types.

Signature:

```
int abs(int number)
double abs(double number)
decimal abs(decimal number)
```

Arguments:

int / double / decimal - The number to use as a base for this function.

Examples:

```
number.abs()
abs(number)
number.abs
```

DateTime and Duration 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 duration.

now

Special notation for the system current date and time.

Examples:

```
setReceivedDate('now')
```

addition

Adds an interval to a dateTime variable or value.

Examples:

```
today + '1d3h'
now + 3d
vacationStartingDate + '1M'
```

subtraction

Subtracts an interval to a dateTime variable or value.

Examples:

```
today - '2d3h25m'
now - age
expirationDate - '7d'
```

year

Returns the year of this dateTime variable.

Signature:

```
int year(dateTime date)
```

Arguments:

dateTime - The date to obtain the year from.

Examples:

```
today.year()
year(today)
today.year
```

month

Returns the month of this dateTime variable.

Signature:

```
int month(dateTime date)
```

Arguments:

dateTime - The date to obtain the month from.

Examples:

```
today.month()
month(today)
today.month
```

day

Returns the day of this dateTime variable.

Signature:

```
int day(dateTime date)
```

Arguments:

dateTime - The date to obtain the day from.

Examples:

```
today.day()
day(today)
today.day
```

hours

Returns the hour of this dateTime variable.

Signature:

```
int hours(dateTime date)
```

Arguments:

dateTime - The date to obtain the hours from.

Examples:

```
today.hours()
hours(today)
today.hours
```

minutes

Returns the minutes of this dateTime variable.

Signature:

```
int minutes(dateTime date)
```

Arguments:

dateTime - The date to obtain the minutes from.

Examples:

```
today.minutes()
minutes(today)
today.minutes
```

seconds

Returns the seconds of this dateTime variable.

Signature:

```
int seconds(dateTime date)
```

Arguments:

dateTime - The date to obtain the seconds from.

Examples:

```
today.seconds()
seconds(today)
today.seconds
```

timezone

Returns an duration value that represents the offset from UTC.

Signature:

```
duration 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'.

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.

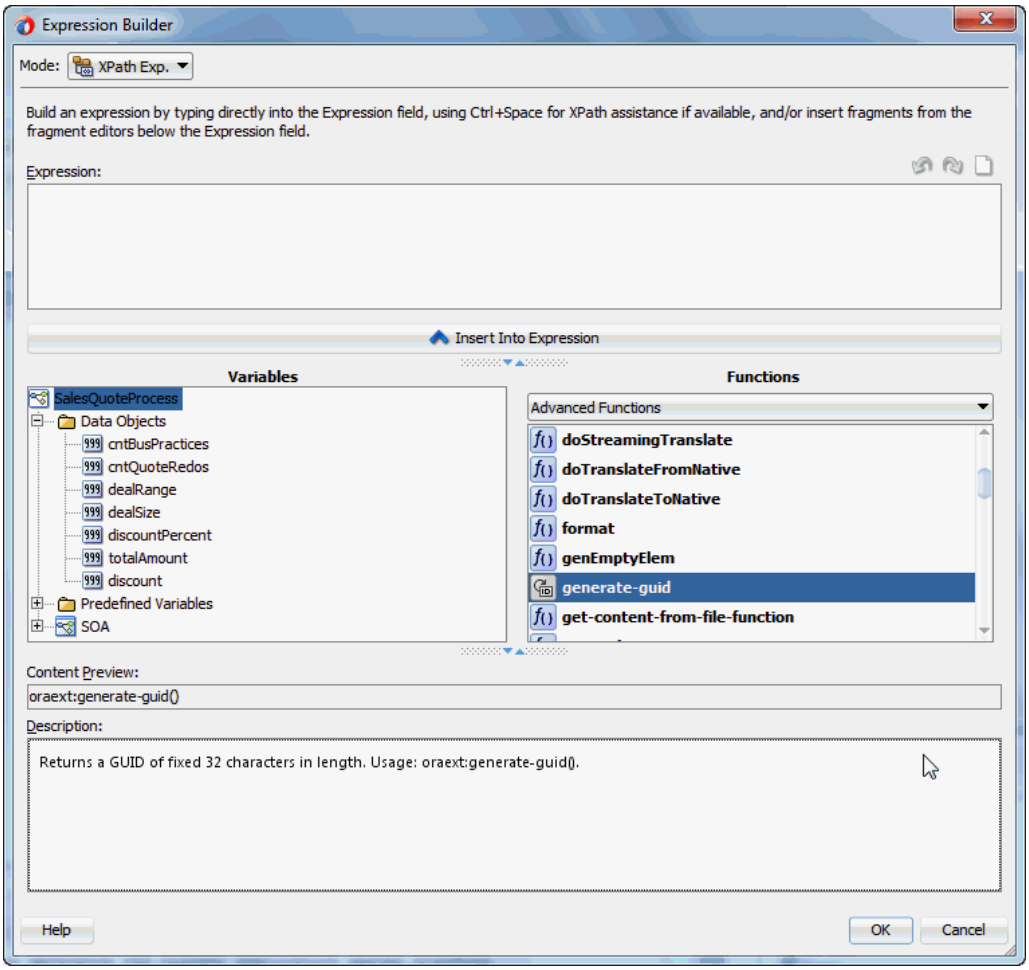
The XPath Expression Builder displays a list of the available variables that you can use in your expression. It also displays a list with the functions you can use in your expressions. When you select a function you can preview its syntax and description before adding it to your expression.

For more information about the functions supported for XPath see "Appendix B XPath Extension Functions" in *Developing SOA Applications with Oracle SOA Suite*.

Additionally Oracle BPM supports a group of BPM Extension Functions.

Figure 24-2 shows XPath Expression Builder.

Figure 24-2 XPath Expression Builder



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.

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.

Using Arrays

Some service operations may require or return arguments of type array. These arrays can be collections ofsimple 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 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 Introduction to the Data Association Editor.

Example 24-2 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.

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].

Example 24-2 Expression accessing an element of an array

```
persons[3]
```

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 24-3 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 24-3 Expression accessing an attribute of an element of an array

```
persons[1].name
```

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 Writing Conditions in Loop and Multi-Instance Markers in Subprocesses.

To obtain the length of an array you must invoke the length function.

Example 24-3 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 24-4 Expression obtaining the length of an array

```
persons.length()
```

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
- duration literals
- array literals

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"

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'`

Using Duration Literals

You can assign a value to a data object of type *duration interval* literals. You can also use interval literals to provide a parameter of type duration to a method.

Duration 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 24-6 shows the available time unit fields.

Table 24-6 Time Unit Suffixes

Time Unit Suffix	Description
Y	Year
M	Month

Table 24-6 (Cont.) Time Unit Suffixes

Time Unit Suffix	Description
d or D	Day
h or H	Hour
m	Minutes
s or S	Seconds
x	Microseconds

Table 24-7 shows examples of interval literals:

Table 24-7 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

Using Array Literals

You can assign a value to a data object of type *array* using literals. You can also use array literals to provide a parameter of type array 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]

XPath BPM Extension Functions

The BPM Extension Functions enable you to access various elements using XPath. In XPath this is the only way of accessing the value of the described elements in your BPMN process.

- Process and Project Data Objects
- Arguments
- Activity Instance Attributes

You can also use the XPath extension functions that Oracle SOA Suite provides. For more information see appendix XPath Extension Functions, in *Developing SOA Applications with Oracle SOA Suite*.

getActivityInstanceAttribute

Returns the value of a specific activity instance attribute. See 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)
```

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.

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)
```

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.

getGatewayInstanceAttribute

Returns the value of a specific activity instance attribute in a gateway. See 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.

getProcessInstanceAttribute

Returns the value that corresponds to a process activity instance attribute. See 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)
```

getBusinessParameter

Returns the value that corresponds to a business parameter.

Signature:

```
bpmn:getBusinessParameter(businessParametername)
```

Arguments:

`businessParameterName` - string that contains the name of the business parameter whose value you want to find out.

Examples:

```
bpmn:getBusinessParameter(APPROVED_DISCOUNT)
```

bpmn:getBusinessParameter(businessParameterName)

25

Writing BPM Scripts

This chapter describes how to use BPM scripting to access and modify the data objects in a BPM Project

This chapter includes the following sections:

- Introduction to BPM Scripting
- Introduction to the BPM Code Editor
- Introduction to the Scripting Catalog
- Importing Custom Libraries
- Working with the Elements of a BPM Project
- Importing Business Objects from the Business Catalog
- Predefined Variables
- Implementing Script Tasks
- Type Description Mapping for XML Schema Types

Introduction to BPM Scripting

Oracle BPM supports Groovy 2.1 compiler and runtime (with static compilation enabled) as a scripting language.

Scripting is available in the following contexts:

- Script task

  See How to Implement a Script Task

- Business object methods

  See Working with Business Object Methods

Within a script block, you can create and use types from the Business Catalog, Java 6 SE API Library, and from any JAR files included in the BPM project.

Introduction to the BPM Code Editor

The BPM Code editor provides basic syntax highlighting, error and warnings highlighting, and code completion.

- Basic Syntax Highlighting

  It displays the code in different colors and fonts according to the category of terms. Syntax highlighting also helps developers find mistakes in their code.

- Error and Warnings Highlighting

  It highlights compiling errors and warnings while you are writing the code. It also provides a description of the problem.

- Code Completion

  It predicts a word or phrase that the user wants to type before they finish typing it.

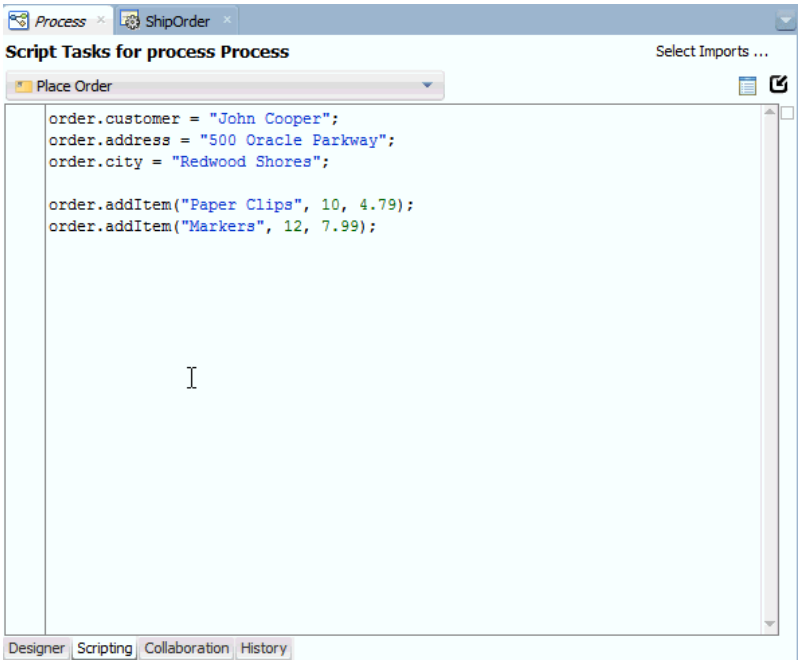
The Code editor supports the following types of code completion:

- Inherited members (extends/implements)
- Instance members (fields/methods)
- Class members (fields/methods)
- Local variables
- Method parameters
- Primitive types
- Class Names (import will be automatically added if not exists)

The editor displays the options ordered by their relevance. If multiple options have the same relevance, the editor orders them alphabetically.

Figure 25-1 shows the Script editor with the implementation of a script task.

Figure 25-1 Script Editor



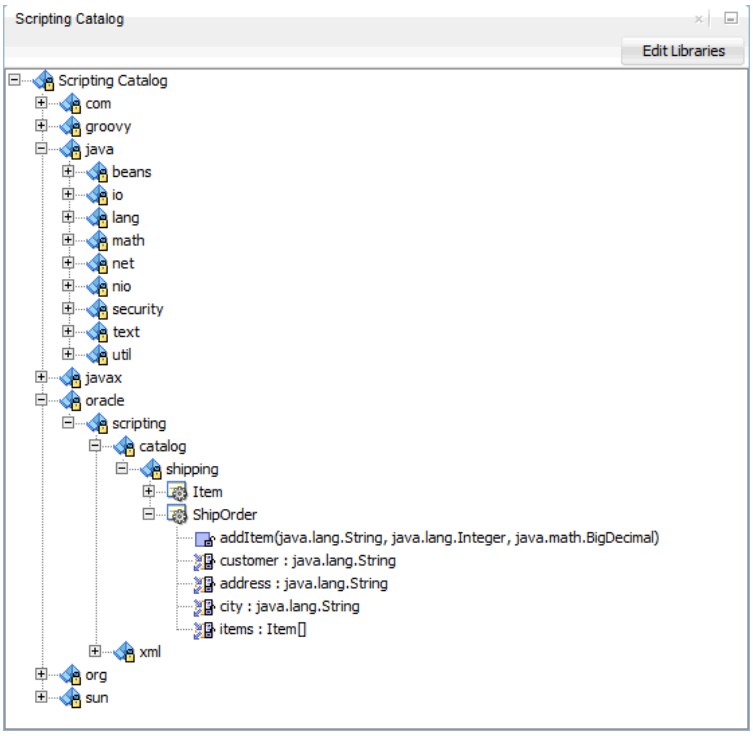
Introduction to the Scripting Catalog

The Scripting Catalog contains all the libraries available for you to use in your scripts. These libraries contain components that you can access by instantiating them and invoking their method. These components are stored in modules.

The Scripting Catalog contains the following components:

- The default Groovy libraries
- The default Java libraries
- The business catalog components
- The imported custom libraries

Figure 25-2 Scripting Catalog Window



Importing Custom Libraries

You can use external Java libraries that contain functions that are useful for your project. This enables you to benefit from the vast amount of libraries that solve different problems and are available for Java developers to use.

You can import these Java libraries and invoke them from the Groovy scripts.

After you import a Java library, it becomes part of the Scripting Catalog for that BPM project and you can use the from the scripts you write.

How to Import a Custom Library

To use an external Java library you must first import it to the BPM project's Scripting Catalog.

To import a custom library:

1. Open the business object or the process where you want to import the library.
2. In the Scripting Catalog window, click Edit Libraries.

   The Libraries dialog box opens.
3. Click the Add button.

The Open dialog box opens.

4. Select a the jar file containing the external library.
5. Click Open.

   The select library appears in the Libraries table.
6. Select the library.
7. Click OK.

   The selected library appears in the Scripting Catalog and is available for you to use its classes in your scripts. Note that some libraries may appear in already existing modules, like org or com.

Working with the Elements of a BPM Project

You can access the value of some of the elements of a BPM project and you can manipulate their values or use them to perform operations.

You can access the following elements of a BPM project from a script:

- project data objects
- process data objects
- business parameters

Project and process data objects are available as variables in the script.

How to Work with Business Objects

You can access the attributes and methods of the data objects defined as a business object, in the same way that you access the attributes and methods of a java object.

For example: You created a business data object that models a person. This business object contains a name attribute and a promote method.

You also defined a process data object of type Person.

You can access the name attribute using the following code:

```
person.name;
```

You can access the promote method using the following code:

```
person.promote;
```

To create an instance of a Business Object and store it in a local variable.:

1. Import the business object.

   For information on how to import a business object, see Importing Business Objects from the Business Catalog.
2. Create a variable with the type of the business object and assign it a new instance of the business object.

The following example imports the Person type, creates a local variable of type person, assigns a value to its attributes and then assigns the local instance to the Person process data object:

```
def localPerson = new Person();
localPerson.firstName = "John";
localPerson.lastName = "Cooper"
this.person = localPerson;
```

How to Work with Business Parameters

After you define a business parameter in the organization you can access it using its getter method.

For example, if you define a business parameter TAX_CODE, then you can access it using the following code:

```
this.getTAXCODE()
```

Importing Business Objects from the Business Catalog

You can use the business objects defined in the business catalog in your scripts. To do this you must first import the business objects.

To import a business object from the business catalog:

1. In the Script Editor, click Select Imports.

   The Select Imports dialog box appears.

2. Click the Add button.

   A new row appears in the imports table.

3. Start typing the path to the business object.

   For example: oracle.scripting.catalog.shipping.ShipOrder

   The auto-complete list displays a list of options.

4. Select the business object you want to import from the auto-complete list.

5. Click OK.

   The selected import is now available for you to use it in your code.

Predefined Variables

The variable *predef* is available in every script task method context. This variable enables you to access the available predefined variables.

The following table shows the supported predefined variables and how to access them using Groovy expressions. If the access type of the predefined variable is read/write, then you can use the Groovy expression to view the value of the predefined variable and to modify it. If the access type is read-only, then you can only view the value of the predefined variable.

Note: BPMN only perform the necessity actions on the data and does not check the authenticity of the data. Predefined variable values must be validated, and right values must be set by client application.

Table 25-1 Groovy Expressions to Access Predefined Variables

Table 25-1 (Cont.) Groovy Expressions to Access Predefined Variables

Predefined Variable	Groovy Expression	Access Type
Organizational Unit	predef.organizationalUnit	Read/write
Title	predef.title	Read/write
Creation Date	predef.creationDate	Read-only
Modify Date	predef.modifyDate	Read-only
Instance Number	predef.instanceNumber	Read-only
Instance ID	predef.instanceId	Read-only
ECID	predef.ecid	Read-only
Process DN	predef.processDN	Read-only
Conversation ID	predef.conversationId	Read-only
Component Type	predef.component.type	Read-only
Component Name	predef.component.name	Read-only
State	predef.state	Read-only
Composite Name	predef.composite.name	Read-only
Composite DN	predef.composite.dn	Read-only
Composite Label	predef.composite.label	Read-only
Composite Revision	predef.composite.revision	Read-only
Composite Instance ID	predef.composite.instanceId	Read-only
Activity Name	predef.activity.name	Read-only
Priority	predef.priority	Read/write
Creator	predef.creator	Read/write
Owner	predef.owner	Read/write
Owner Type	predef.ownerType	Read/write
Action	predef.action	Read/write
Expiration	predef.expiration	Read/write
Reviewer	predef.reviewer	Read/write
ReviewerType	predef.reviewerType	Read/write
DueDate	predef.dueDate	Read/write

Implementing Script Tasks

A BPMN script task is an activity that runs a script.

This script is written in Groovy and can access and modify the value of the data objects defined in a BPM project.

How to Implement a Script Task

Script tasks require you to implement them using Groovy code.

To implement a script task:

1. Add the script task to your BPMN process.
2. Right-click the script task.
3. Select Go To Script.

   The Scripting tab appears. In this tab you can write the script that implements the script task. To go back to the process editor, select the Designer tab.

Type Description Mapping for XML Schema Types

This section describes the mapping between the XML schema data types and the type descriptions used in the scripts.

Table 25-2 Type Description Mapping for XML Schema Types

XML Schema Data Type	Type Description
xsd:string	java.lang.String
xsd:normalizedString	java.lang.String
xsd:token	java.lang.String
xsd:Name	java.lang.String
xsd:QName	java.lang.String
xsd:NCName	java.lang.String
xsd:anyURI	java.lang.String
xsd:language	java.lang.String
xsd:ID	java.lang.String
xsd:IDREF	java.lang.String
xsd:ENTITY	java.lang.String
xsd:NOTATION	java.lang.String
xsd:NMTOKEN	java.lang.String
xsd:uriReference	java.lang.String

Table 25-2 (Cont.) Type Description Mapping for XML Schema Types

XML Schema Data Type	Type Description
xsd:anySimpleType	java.lang.String
xsd:IDREFS	java.lang.String
xsd:ENTITIES	java.lang.String
xsd:NMTOKENS	java.lang.String
xsd:timeInstant	java.lang.String
xsd:timeDuration	java.lang.String
xsd:anyType	java.lang.String
xsd:byte	java.lang.Byte
xsd:unsignedByte	java.lang.Byte
xsd:short	java.lang.Short
xsd:unsignedShort	java.lang.Short
xsd:int	java.lang.Integer
xsd:unsignedInt	java.lang.Integer
xsd:gDay	java.lang.Integer
xsd:gMonth	java.lang.Integer
xsd:gYear	java.lang.Integer
xsd:integer	java.math.BigInteger
xsd:positiveInteger	java.math.BigInteger
xsd:negativeInteger	java.math.BigInteger
xsd:nonPositiveInteger	java.math.BigInteger
xsd:nonNegativeIntege	java.math.BigInteger
xsd.long	java.lang.Long
xsd:unsignedLong	java.lang.Long
xsd:decimal	Decimal
xsd:float	java.lang.Float
xsd:double	java.lang.Double
xsd:boolean	java.lang.Boolean
xsd:dateTime	com.oracle.scripting.lib.xml.datatype.XmlCalendar

Table 25-2 (Cont.) Type Description Mapping for XML Schema Types

XML Schema Data Type	Type Description
xsd:time	com.oracle.scripting.lib.xml.datatype.XmlCalendar
xsd:date	com.oracle.scripting.lib.xml.datatype.XmlCalendar
xsd:gYearMonth	com.oracle.scripting.lib.xml.datatype.XmlCalendar
xsd:gMonthDate	com.oracle.scripting.lib.xml.datatype.XmlCalendar
xsd:duration	com.oracle.scripting.lib.xml.datatype.XmlDuration
xsd:base64Binary	byte[]
xsd:hexBinary	byte[]

26

Debugging a BPM Project

This chapter describes how to use Oracle JDeveloper debugger to debug any of the BPMN processes contained in a BPM project.

This chapter includes the following sections:

- Introduction to Debugging a BPM Project
- Adding a Breakpoint to a BPMN Flow Object
- Adding a Breakpoint to a BPMN Component
- Disabling a Breakpoint
- Debugging a BPM Project

Introduction to Debugging a BPM Project

You can debug any BPMN process contained in a BPM project using Oracle JDeveloper debugger. Debugging a BPMN process enables you to identify and fix any workflow or logical problems that prevent your process from running correctly.

The debugging process involves adding breakpoints to a SOA composite or a BPMN process so that the debugger stops running when it reaches one of those breakpoints and you can:

- Trace the SOA composite or the BPMN process workflow
- Inspect or watch the value of the process instance attributes
- Inspect or watch the value of data objects
- Inspect or watch correlation keys
- Inspect or watch conversations
- Step into calls to other components and view the normalized message values sent from and returned to the BPM component

When a process logical thread runs, it activates different frames. These frames can be nested, depending on the process flow. Each frame consists of a set of data values that represents the value of a data objects and a process instance attributes at the specific declaration level. The nesting of frames conforms the stack frame.

When a process enters a data declaration container, it activates a new frame. The different declaration containers that can appear when running a BPMN process are:

- The BPMN process itself
- An embedded subprocess

- A callable process
- An event subprocess

Each of these process containers support the declaration of data objects, correlations keys and conversations. When these process containers are activated the debugger creates a new frame.

The debugger builds the stack frame according to how these process containers are nested. The stack frame is a model that provides data visibility and enables you to access the data in a BPMN process. For example, a BPMN process that contains an embedded subprocess which in turn contains call activity to a reusable subprocess, defines three levels of nesting. The debugger thread running within the reusable subprocess contains a stack frame with three elements: the frame on the top corresponds to the reusable process, the last stack element corresponds to the BPMN process.

For more information about Oracle JDeveloper debugger, see Running and Debugging Java Programs in *User's Guide for Oracle JDeveloper*.

Adding a Breakpoint to a BPMN Flow Object

You can add a breakpoint to a BPMN flow object to stop the debugger when the process flow reaches that BPMN flow object.

Breakpoints define where the debugger stops while running your BPMN process. You can add breakpoints to multiple BPMN flow objects. When the debugger reaches one of these breakpoints, it stops allowing you to monitor the values of the different variables in a project. These values may help you to resolve any problems with the workflow or logic of your BPMN process.

How to Add a Breakpoint to a BPMN Flow Object

To add a breakpoint to a BPMN flow object:

1. Edit the *BPMN process* that contains the BPMN flow object where you want to add a breakpoint.
2. Right click the *BPMN flow object* where you want to add the breakpoint.
3. Select **Toggle Breakpoint**.

   A red dot appears in the upper right corner of the BPMN flow object to indicate there is a breakpoint.

   A breakpoint of the type BPM Breakpoint appears in the Breakpoints View.

Adding a Breakpoint to a BPMN Component

You can add a breakpoint to a BPMN component to stop the debugger when the debugger reaches it while running an SOA composite.

Breakpoints define where the debugger stops while running your BPMN process. You can add breakpoints to multiple BPMN flow objects. When the debugger reaches one of these breakpoints, it stops allowing you to monitor the values of the different variables in a project. These values may help you to resolve any problems with the workflow or logic of your BPMN process.

How to Add a Breakpoint to a BPMN Component

To add a breakpoint to a BPMN flow object:

1. Edit the *SOA composite* that contains the BPMN component where you want to add a breakpoint.
2. Right click the *BPMN component* where you want to add the breakpoint.
3. Select one of the following:
   - Create Breakpoint Pair

     To monitor messages entering and exiting the process.
   - Create Request Breakpoint

     To monitor messages entering the process.
   - Create Reply Breakpoint

     To monitor messages exiting the process.

   A red dot or a couple of red dots appear in the upper left corner of the BPMN component to indicate there is a breakpoint.

   A breakpoint of the type *BPM Breakpoint* appears in the Breakpoints View.

Disabling a Breakpoint

You can disable a breakpoint that you do not need at a certain point in your debugging.

As you progress in your debugging session you may need to remove some of the breakpoints you added and add breakpoints in other places.

How to Disable a Breakpoint

To disable a breakpoint:

1. Right click the *BPMN flow object* that contains the breakpoint that you want to disable.
2. Select **Disable Breakpoint**.

Debugging a BPM Project

You can debug any of the BPMN processes contained in a BPM project.

You use Oracle JDeveloper debugger to troubleshoot any of the processes contained in the BPM project.

How to Attach a BPM Project to the Debugger

To attach a BPM project to the debugger:

1. In the Applications window, right-click the BPM project you want to debug and select the **Debug** action.

2. Configure the server connection.

   This is a specific SOA debug connection that you must configure separately. For more information on how to configure the server connection, see How to Use a Project Configured for Remote Debugging in Developing Applications with Oracle JDeveloper.

   The Deploy Application dialog box appears.

3. Select Deploy to Application Server.

4. Click Next.

   The Deploy Configuration page appears.

5. Optionally modify the deploy configuration.

6. Click Next.

   The Select Server dialog box appears.

7. Select a server or add a new server connection.

8. Click Finish.

   The Composite editor and the Debugging window appears.

9. Run the BPMN process step by step.

   When the debugger reaches a breakpoint in the BPMN process, it highlights the BPMN flow object where the breakpoint is.

Part VI

Using Human Interaction Components

This part provides on overview on how to use human interaction components in BPM Projects.

This part contains the following chapters:

- Getting Started with Human Workflow
- Designing Human Tasks in Oracle BPM
- Configuring Human Tasks
- Working with Guided Business Processes
- Building a Guided Business Process Client Application
- Using Approval Management
- Working with Adaptive Case Management

27

Getting Started with Human Workflow

This chapter describes for developers the human workflow concepts, features, and architecture. Use cases for human workflow are provided. Instructions for designing your first workflow from start to finish are also provided.

This chapter includes the following sections:

- Introduction to Human Workflow
- Introduction to Human Workflow Concepts
- Introduction to Human Workflow Features
- Introduction to Human Workflow Architecture
- Human Workflow and Business Rule Differences Between Oracle SOA Suite and Oracle BPM Suite

Warning:

You must not modify SOA Human Task database tables directly. Oracle does not guarantee backward compatibility for the column names and data in these tables.

Introduction to Human Workflow

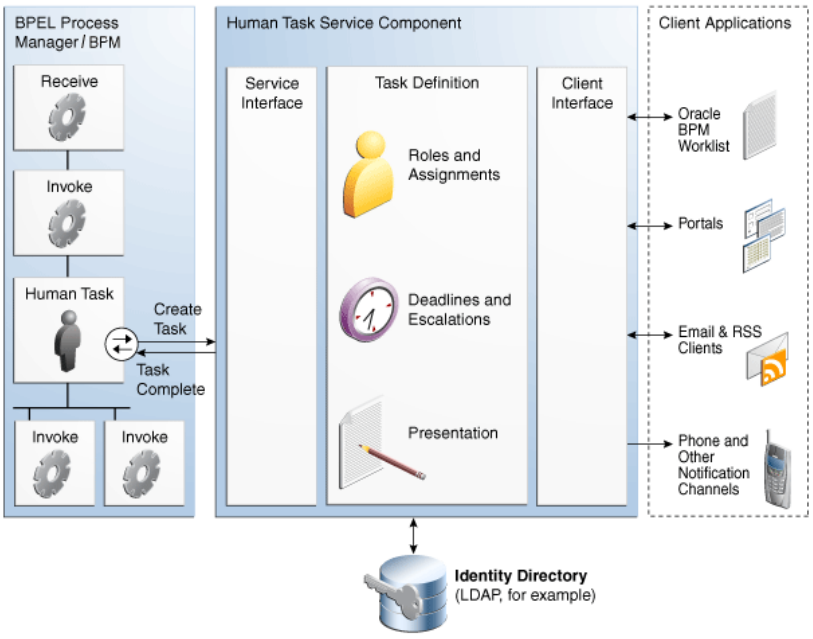
Many end-to-end business processes require human interactions with the process. For example, humans may be needed for approvals, exception management, or performing activities required to advance the business process.

The human workflow component provides 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)
- Presentation of tasks to end users through a variety of mechanisms, including a worklist application (Oracle BPM Worklist)
- Organization, filtering, prioritization, and other features required for end users to productively perform their tasks
- Reports, reassignments, load balancing, and other features required by supervisors and business owners to manage the performance of tasks

Figure 27-1 provides an overview of human workflow.

Figure 27-1 Human Workflow



In Figure 27-1, the following actions occur:

- A BPEL process invokes a special activity of the human task type when it needs a human to perform a task.
- This creates a task in the human task service component. The process waits for the task to complete. It is also possible for the process to watch for other callbacks from the task and react to them.
- There is metadata associated with the task that is used by the human task service component to manage the lifecycle of the task. This includes specification of the following:
  - Who performs the task. If multiple people are required to perform the task, what is the order?
  - Who are the other stakeholders?
  - When must the task be completed?
  - How do users perform the task, what information is presented to them, what are they expected to provide, and what actions can they take?
- The human task service component uses an identity directory to determine people's roles and privileges.

  You can configure the identity store to use the embedded WebLogic LDAP, Oracle Virtual Directory, third-party LDAPs and Active Directory RDBMS. For more information, see Securing Applications with Oracle Platform Security Services.
- The human task service component presents tasks to users through a variety of channels, including the following:

- Oracle BPM Worklist, a role-based application that supports the concept of supervisors and process owners, and provides functionality for finding, organizing, managing, and performing tasks.
- Worklist functionality is also available as portlets that can be exposed in an enterprise portal.
- Notifications can be sent by email, phone, SMS, and other channels. Email notifications can be actionable, enabling users to perform actions on the task from within the email client without connecting to Oracle BPM Worklist or Oracle WebLogic Server.

Introduction to Human Workflow Concepts

This section introduces you to key human workflow design time and runtime concepts.

This section also provides an overview of the three main stages of human workflow design.

Introduction to Design and Runtime Concepts

Before designing a human task, it is important to understand the design and runtime concepts. A typical task consists of a subject, priority, task participants, task parameters or data, deadlines, notifications or reminders, and task forms. This section provides an overview of key concepts.

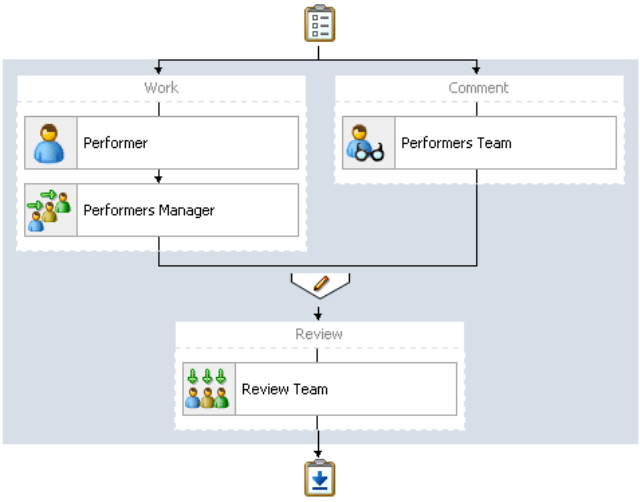
Note:

Human workflow design-time tasks are performed in a graphical editor known as the Human Task Editor. The tutorial in *Developing SOA Applications with Oracle SOA Suite* describes how to use this editor.

Task Assignment and Routing

Human workflow supports declarative assignment and routing of tasks. In the simplest case, a task is assigned to a single participant (user or group). However, there are many situations in which more detailed task assignment and routing is necessary (for example, when a task must be approved by a management chain or worked and voted on by a set of people in parallel, as shown in Figure 27-2). Human workflow provides declarative, pattern-based support for such scenarios.

Figure 27-2 Participants in a Task



Participant

A participant is a user or set of users in the assignment and routing policy definition. In Figure 27-2, each block with an icon representing people is a participant.

Participant Type

In simple cases, a participant maps to a user, group, or role. However, as discussed in Task Assignment and Routing, workflow supports declarative patterns for common routing scenarios such as management chain and group vote. The following participant types are available:

- Single approver

  This is the simple case where a participant maps to a user, group, or role.

  For example, a vacation request is assigned to a manager. The manager must act on the request task three days before the vacation starts. If the manager formally approves or rejects the request, the employee is notified with the decision. If the manager does not act on the task, the request is treated as rejected. Notification actions similar to the formal rejection are taken.

- Parallel

  This participant indicates that a set of people must work in parallel. This pattern is commonly used for voting.

  For example, multiple users in a hiring situation must vote to hire or reject an applicant. You specify the voting percentage that is needed for the outcome to take effect, such as a majority vote or a unanimous vote.

- Serial

  This participant indicates that a set of users must work in sequence. While working in sequence can be specified in the routing policy by using multiple participants in sequence, this pattern is useful when the set of people is dynamic. The most common scenario for this is management chain escalation, which is done by specifying that the list is based on a management chain within the specification of this pattern.

- FYI (For Your Information)

This participant also maps to a single user, group, or role, just as in single approver. However, this pattern indicates that the participant just receives a notification task and the business process does not wait for the participant's response. FYI participants cannot directly impact the outcome of a task, but in some cases can provide comments or add attachments.

For example, a regional sales office is notified that a candidate for employment has been approved for hire by the regional manager and their candidacy is being passed onto the state wide manager for approval or rejection. FYIs cannot directly impact the outcome of a task, but in some cases can provide comments or add attachments.

For more information, see Assigning Task Participants.

Participant Assignment

A task is work that must be done by a user. When you create a task, you assign humans to participate in and act upon the task. Participants can perform actions upon tasks during runtime from Oracle BPM Worklist, such as approving a vacation request, rejecting a purchase order, providing feedback on a help desk request, or some other action. There are three types of participants:

- Users

  You can assign individual users to act upon tasks. For example, you may assign users `jlondon` or `jstein` to a particular task. Users are defined in an identity store configured with the SOA Infrastructure. These users can be in the embedded LDAP of Oracle WebLogic Server, Oracle Internet Directory, or a third-party LDAP directory.

- Groups

  You can assign groups to act upon tasks. Groups contain individual users who can claim and act upon a task. For example, users `jcooper` and `fkafka` may be members of the group `LoanAgentGroup` that you assign to act upon the task.

  As with users, groups are defined in the identity store of the SOA Infrastructure.

- Application roles

  You can assign users who are members of application roles to claim and act upon tasks.

  Application roles consist of users or other roles grouped logically for application-level authorizations. These roles are application-specific and are defined in the application Java policy store rather than the identity store. These roles are used by the application directly and are not necessarily known to a Java EE container.

  Application roles define policy. Java permissions can be granted to application roles. Therefore, application roles define a set of permissions granted to them directly or indirectly through other roles (if a role is granted to a role). The policy can contain grants of application roles to enterprise groups or users. In the `jazn-data.xml` file of the file-based policy store, these roles are defined in `<app-role>` elements under `<policy-store>` and written to `system-jazn-data.xml` at the farm level during deployment. You can also define these roles after deployment using Oracle Enterprise Manager Fusion Middleware Control. You can set a task owner or approver to an application role at design time if the role has been previously deployed.

For more information about Oracle BPM Worklist, see Task Forms.

Ad Hoc Routing

In processes dealing with significant variance, you cannot always determine all participants. Human workflow enables you to specify that a participant can invite other participants as part of performing the task.

For more information, see Allow All Participants to Invite Other Participants or Edit New Participants.

Outcome-based Completion of Routing Flow

By default, a task goes from starting to final participant according to the flow defined in the routing policy (as shown in Figure 27-2). However, sometimes a certain outcome at a particular step within a task's routing flow makes it unnecessary or undesirable to continue presenting the task to the next participants. For example, if an approval is rejected by the first manager, it does not need to be routed to the second manager. Human workflow supports specifying that a task or subtask be completed when a certain outcome occurs.

For more information, see Stopping Routing of a Task to Further Participants.

Static, Dynamic, and Rule-Based Task Assignment

There are different methods for assigning users, groups, and application roles to tasks.

- Assign tasks statically

  You can assign users, groups, and application roles statically (or by browsing the identity service). The values can be either of the following:

  – A single user, group, or application role (for example, `jstein`, `CentralLoanRegion`, or `ApproverRole`).

  – A delimited string of users, groups, or application roles (for example, `jstein`, `wfaulk`, `cdickens`).

- Assign tasks dynamically

  You can assign users, groups, and application roles dynamically in the following ways:

  – By using a task-assignment pattern. This pattern enables you to do the following:

    ◆ Simply enable participants to claim the task manually. This is the default behavior. No task-assignment pattern is applied.

    ◆ If the participant type is either Single or FYI, then apply a task-assignment pattern to select a single assignee of a requested type from *all* potential assignees in the participant.

    For example, suppose that the potential assignees comprise the user `jcooper`, the group `LoanAgent`, and the application role `Developers`. Suppose further that the requested type is `user`. Applying this task-assignment pattern selects a single user from the user `jcooper`, and from all members of the group `LoanAgent`, and from all users with the application role `Developers`.

- If the particulates type is Parallel or Serial, then apply a task-assignment pattern to select a single assignee of a requested type from *each* of the potential assignees in the participant.

  For example, suppose that the potential assignees comprise the user `jcooper`, the group `LoanAgent`, and the application role `Developers`. Suppose further that the requested type is `user`. Applying this task-assignment pattern selects the user `jcooper`, and one user from the group `LoanAgent`, and one user with the application role `Developers`.

– By using XPath expressions. These expressions enable you to dynamically determine assignment to users not included in the participant type. Here you create a list of potential assignees, one of whom must then claim the task.

  For example, you may have a business requirement to create a dynamic list of task approvers specified in a payload variable. The XPath expression can resolve to zero or more XML nodes. Each node value can be either of the following:

  - A single user, group, or application role
  - A delimited string of users, groups, or application roles. The default delimiter for the assignee delimited string is a comma (`,`).

  For example, if the task has a payload message attribute named `po` within which the task approvers are stored, you can use the following XPath expression:

  - `/task:task/task:payload/po:purchaseOrder/po:approvers`
  - `ids:getManager('jstein', 'jazn.com')`

    This returns the manager of `jstein`.

  - `ids:getReportees('jstein', 2, 'jazn.com')`

    This returns all reportees of `jstein` up to two levels.

  - `ids:getUsersInGroup('LoanAgentGroup', false, 'jazn.com')`

    This returns all direct and indirect users in the group `LoanAgentGroup`.

  You can use both options simultaneously—for example, you can use an XPath expression to dynamically select a group, and then apply a task-assignment pattern to dynamically select a user from that group.

- Assign tasks with business rules

  You can create the list of task participants with complex expressions. The result of using business rules is the same as using XPath expressions. You can also apply the task-assignment pattern to a participant list created using business rules.

Task Stakeholders

A task has multiple stakeholders. Participants are the users defined in the assignment and routing section of the task definition. These users are the primary stakeholders that perform actions on the task.

In addition to the participants specified in the assignment and routing policy, human workflow supports additional stakeholders:

- Owner

This participant has business administration privileges on the task. This participant can be specified as part of the task definition or from the invoking process (and for a particular instance). The task owner can act upon tasks they own and also on behalf of any other participant. The task owner can change both the outcome of the task and the assignments.

For more information, see How to Specify a Task Owner to specify an owner in the Human Task Editor or *Developing SOA Applications with Oracle SOA Suite* to specify an owner in the **Advanced** tab of the Human Task dialog box.

- Initiator

  The person who initiates the process (for example, the initiator files an expense report for approval). This person can review the status of the task using initiated task filters. Also, a useful concept is for including the initiator as a potential candidate for request-for-information from other participants.

  For more information, see *Developing SOA Applications with Oracle SOA Suite*.

- Reviewer

  This participant can review the status of the task and add comments and attachments. You can grant the reviewer role to a participant at runtime using the process instance attributes reviewer and reviewerType. The reviewer process attribute stores the name of the reviewer, the default value is "ProcessReviewer" or the value assigned in the Human Task configuration. The reviewerType process attribute stores the type of reviewer which can be: user, role or group. You can set these attributes dynamically to modify the effective reviewer.

- Admin

  This participant can view all tasks and take certain actions such as reassigning a test, suspending a task to handle errors, and so on. The task admin cannot change the outcome of a task.

  While the task admin cannot perform the types of actions that a task participant can, such as approve, reject, and so on, this participant type is the most powerful because it can perform actions such as reassign, withdraw, and so on.

- Error Assignee

  When an error occurs, the task is assigned to this participant (for example, the task is assigned to a nonexistent user). The error assignee can perform task recovery actions from Oracle BPM Worklist, the task form in which you perform task actions during runtime.

  For more information, see How to Configure the Error Assignee and Reviewers.

Task Deadlines

Human workflow supports the specification of deadlines associated with a task. You can associate the following actions with deadlines:

- Reminders:

  The task can be reminded multiple times based on the time after the assignment or the time before the expiration.

- Escalation:

  The task is escalated up the management hierarchy.

- Expiration:

  The task has expired.

- Renewal:

  The task is automatically renewed.

For more information, see Escalating_ Renewing_ or Ending the Task.

Notifications

You can configure your human task to use notifications. Notifications enable you to alert interested users to changes in the state of a task during the task lifecycle. For example, a notification is sent to an assignee when a task has been approved or withdrawn.

You can specify for notifications to be sent to different types of participants for different actions. For example, you can specify the following:

- For the owner of a task to receive a notification message when a task is in error (for example, sent to a nonexistent user).
- For a task assignee to receive a notification message when a task has been escalated.

You can specify the contents of the notification message and the notification channel to use for sending the message.

- Email

  You can configure email notification messages to be actionable, meaning that a task assignee can act upon a task from within the email.

- Voice message
- Instant messaging (IM)
- Short message service (SMS)

For example, you may send the message shown in Example 27-1 by email when a task assignee requests additional information before they can act upon a task:

During runtime, you can mark a message sender's address as spam and also display a list of bad or invalid addresses. These addresses are automatically added to the bad address list.

For more information about notifications, see the following:

- *Developing SOA Applications with Oracle SOA Suite*
- Specifying Participant Notification Preferences
- Part XI, "Using Oracle User Messaging Service"

Example 27-1 Email Message

```
For me to approve this task, more information is required to justify the need
 for this business trip
```

Task Forms

Task forms provide you with a way to interact with a task. Oracle BPM Worklist displays all worklist tasks that are assigned to task assignees in the task form. When

you navigate into a specific task, the task form displays the contents of the task to the user's worklist. 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 integrated development environment of Oracle SOA Suite includes Oracle Application Development Framework (Oracle ADF) for this purpose. With Oracle ADF, you can design a task form that depicts the human task in the SOA composite application.

ADF-based task forms can be automatically generated. Advanced users can design their own task forms by using ADF data controls to lay out the content on the page and connect to the workflow service engine at execution time to retrieve task content and act on tasks.

You can create task forms in JSF, .NET, or any other client technologies using the APIs.

For more information, see:

- the chapter on designing task forms for human tasks in *Developing SOA Applications with Oracle SOA Suite*
- the chapter on using BPM Worklist in *Developing SOA Applications with Oracle SOA Suite*

Advanced Concepts

This section describes advanced human workflow concepts.

Rule-based Routing

You can use Oracle Business Rules to dynamically alter the routing flow. If used, each time a participant completes their step, the associated rules are invoked and the routing flow can be overridden from the rules.

For more information, see How to Specify Advanced Task Routing Using Business Rules.

Rule-based Participant Assignment

You can use Oracle Business Rules to dynamically build a list of users, groups, and roles to associate with a participant.

For more information, see Assigning Task Participants.

Stages

A stage is a way of organizing the approval process for blocks of participant types. You can have one or more stages in sequence or in parallel. Within each stage, you can have one or more participant type blocks in sequence or in parallel.

For more information, see Assigning Task Participants.

Access Rules

You can specify access rules that determine the parts of a task that assignees can view and update. For example, you can configure the task payload data to be read by assignees. This action enables only assignees (and nobody else) to have read permissions. No one, including assignees, has write permissions.

For more information, see How to Specify Access Policies on Task Content.

Callbacks

While human workflow supports detailed behavior that can be declaratively specified, in some advanced situations, more extensible behavior may be required. Task callbacks enable such extensibility; these callbacks can either be handled in the invoking BPEL process or a Java class.

For more information, see How to Specify Callback Classes on Task Status.

Reports and Audit Trails

Oracle BPM Worklist provides several out-of-the-box reports for task analysis:

- Unattended tasks

  Analysis of tasks assigned to users' groups or reportees' groups that have not yet been acquired.

- Tasks priority

  Analysis of tasks assigned to a user, reportees, or their groups, based on priority.

- Tasks cycle time

  Analysis of the time taken to complete tasks from assignment to completion based on users' groups or reportees' groups.

- Tasks productivity

  Analysis of assigned tasks and completed tasks in a given time period for a user, reportees, or their groups.

- Tasks time distribution

  The time an assignee takes to perform a task.

You can view an audit trail of actions performed by the participants in the task and a snapshot of the task payload and attachments at various points in the workflow. The short history for a task lists all versions created by the following tasks:

- Initiate task
- Reinitiate task
- Update outcome of task
- Completion of task
- Erring of task
- Expiration of task
- Withdrawal of task
- Alerting of task to the error assignee

For more information, see the chapter on the BPM Worklist in *Developing SOA Applications with Oracle SOA Suite*.

Introduction to the Stages of Human Workflow Design

Human workflow modeling consists of three stages of modeling, as described in Table 27-1.

Table 27-1 Stages of Human Workflow Modeling

Step	Description
1	You create and define contents of the human task in the Human Task Editor, including defining a participant type, routing policy, escalation and expiration policy, notification, and so on.
2	You associate the human task definition with a BPEL process. The BPEL process integrates a series of activities (including the human task activity) and services into an end-to-end process flow.
3	You create a task form. This form displays the task details on which you act at runtime in Oracle BPM Worklist.

For more information, see *Developing SOA Applications with Oracle SOA Suite*.

Introduction to Human Workflow Features

This section provides an introduction to use cases for human workflow.

A tutorial guides you through the design of a human task from start to finish.

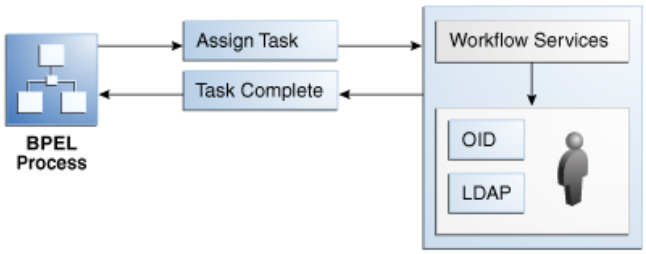
Human Workflow Use Cases

The following sections describe multiple use cases for workflow services.

Task Assignment to a User or Role

A vacation request process may start with getting the vacation details from a user and then routing the request to their manager for approval. User details and the organizational hierarchy can be looked up from a user directory or identity store. This scenario is shown in Figure 27-3.

Figure 27-3 Assigning Tasks to a User or Role from a Directory

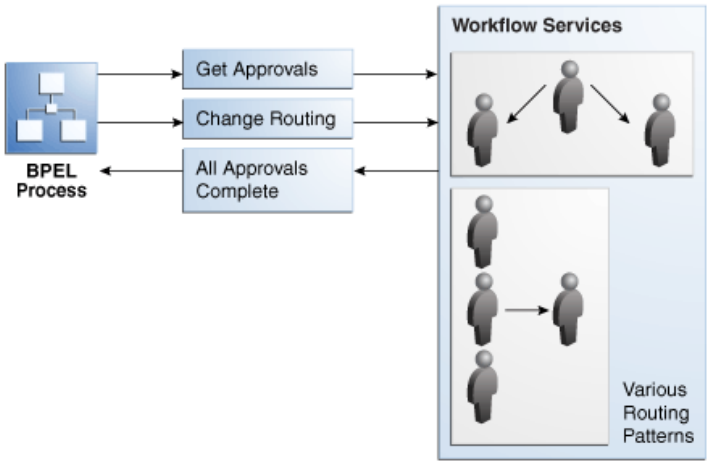


Use of the Various Participant Types

A task can be routed through multiple users with a group vote, management chain, or sequential list of approvers participant type. For example, consider a loan request that is part of the loan approval flow. The loan request may first be assigned to a loan agent role. After a specific loan agent acquires and accepts the loan, the loan may be

routed further through multiple levels of management if the loan amount is greater that $100,000. This scenario is shown in Figure 27-4.

Figure 27-4 Flow Patterns and Routing Policies

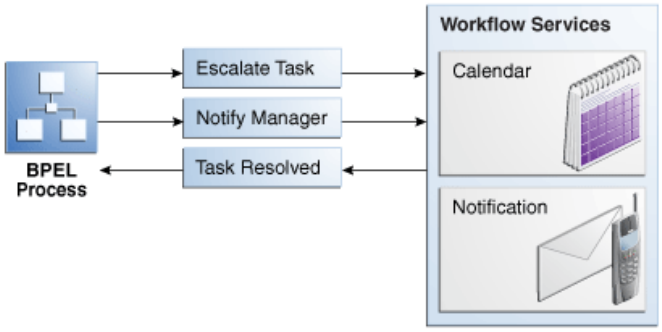


You can use these types as building blocks to create complex workflows.

Escalation, Expiration, and Delegation

A high-priority task can be assigned to a certain user or role based on the task type through use of custom escalation functions. However, if the user does not act on it in a certain time, the task may expire and in turn be escalated to the manager for further action. As part of the escalation, you may also notify the users by email, telephone voice message, or SMS. Similarly, a manager may delegate tasks from one reportee to another to balance the load between various task assignees. All tasks defined in BPEL have an associated expiration date. Additionally, you may specify escalation or renewal policies, as shown in Figure 27-5. For example, consider a support call, which is part of a help desk service request process. A high-priority task may be assigned to a certain user, and if the user does not respond in two days, the task is routed to the manager for further action.

Figure 27-5 Escalation and Notification



Automatic Assignment and Delegation

A user may decide to have another user perform tasks on their behalf. Tasks can be explicitly delegated from the Oracle BPM Worklist or can be automatically delegated. For example, a manager sets up a vacation rule saying that all their high priority tasks are automatically routed to one of their direct reports while the manager is on vacation. In some cases, tasks can be routed to different individuals based on the

content of the task. Another example of automatic routing is to allocate tasks among multiple individuals belonging to a group. For example, a help desk supervisor decides to allocate all tasks for the western region based on a round robin basis or assign tasks to the individual with the lowest number of outstanding tasks (the least busy).

Dynamic Assignment of Users Based on Task Content

An employee named James in the human resources department requests new hardware that costs $5000. The company may have a policy that all hardware expenses greater than $3000 must go through manager and vice president approval, and then review by the director of IT. In this scenario, the workflow can be configured to automatically determine the manager of James, the vice president of the human resources department, and the director of IT. The purchase order is routed through these three individuals for approval before the hardware is purchased.

Introduction to Human Workflow Architecture

This section provides an overview of human workflow architecture.

The following topics are discussed:

- The services that perform a variety of operations in the lifecycle of a task, such as querying tasks for a user, retrieving metadata information related to a task, and so on.
- The two ways to use a human task:
  - Associated with a BPEL process service component
  - Used in standalone mode
- The role of the service engine in the life of a human task

Human Workflow Services

Starting with release 11*g*, all human task metadata is stored and managed in the Metadata Service (MDS) repository. The workflow service consists of many services that handle various aspects of human interaction with a business process.

Figure 27-6 shows the following workflow service components:

- Task Service:

  The task service provides task state management and persistence of tasks. In addition to these services, the task service exposes operations to update a task, complete a task, escalate and reassign tasks, and so on. The task service is used by Oracle BPM Worklist to retrieve tasks assigned to users. This service also determines if notifications are to be sent to users and groups when the state of the task changes. The task service consists of the following services.

  - Task Routing Service

    The task routing service offers services to route, escalate, and reassign the task. The service makes these decisions by interpreting a declarative specification in the form of the routing slip.

  - Task Query Service

The task query service queries tasks for a user based on a variety of search criterion such as keyword, category, status, business process, attribute values, history information of a task, and so on.

– Task Metadata Service

The task metadata service exposes operations to retrieve metadata information related to a task.

- Identity Service

  The identity service is a thin web service layer on top of the Oracle Application Server 11*g* security infrastructure or any custom user repository. It enables authentication and authorization of users and the lookup of user properties, roles, group memberships, and privileges.

- Notification Service

  The notification service delivers notifications with the specified content to the specified user through any of the following channels: email, telephone voice message, IM, and SMS. See *Developing SOA Applications with Oracle SOA Suite* for more information.

- User Metadata Service

  The user metadata service manages metadata related to workflow users, such as user work queues, preferences, vacations, and delegation rules.

- Runtime Config Service

  The runtime config service provides methods for managing metadata used in the task service runtime environment. It principally supports management of task payload mapped attribute mappings.

- Evidence service

  The evidence service supports storage and nonrepudiation of digitally-signed workflow tasks.

Figure 27-6 Workflow Services Components

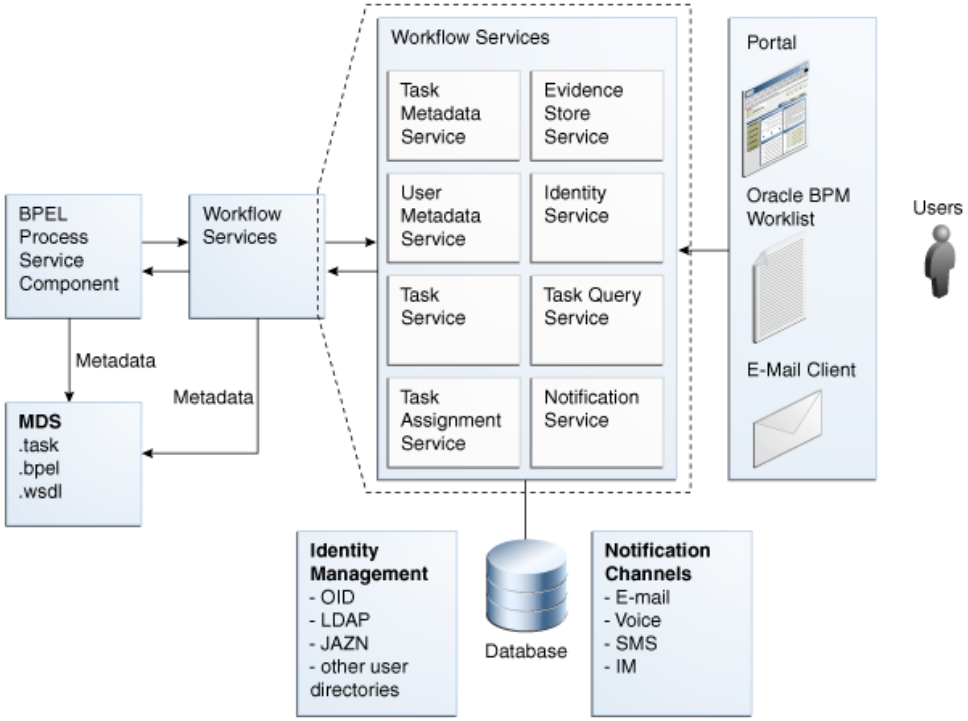
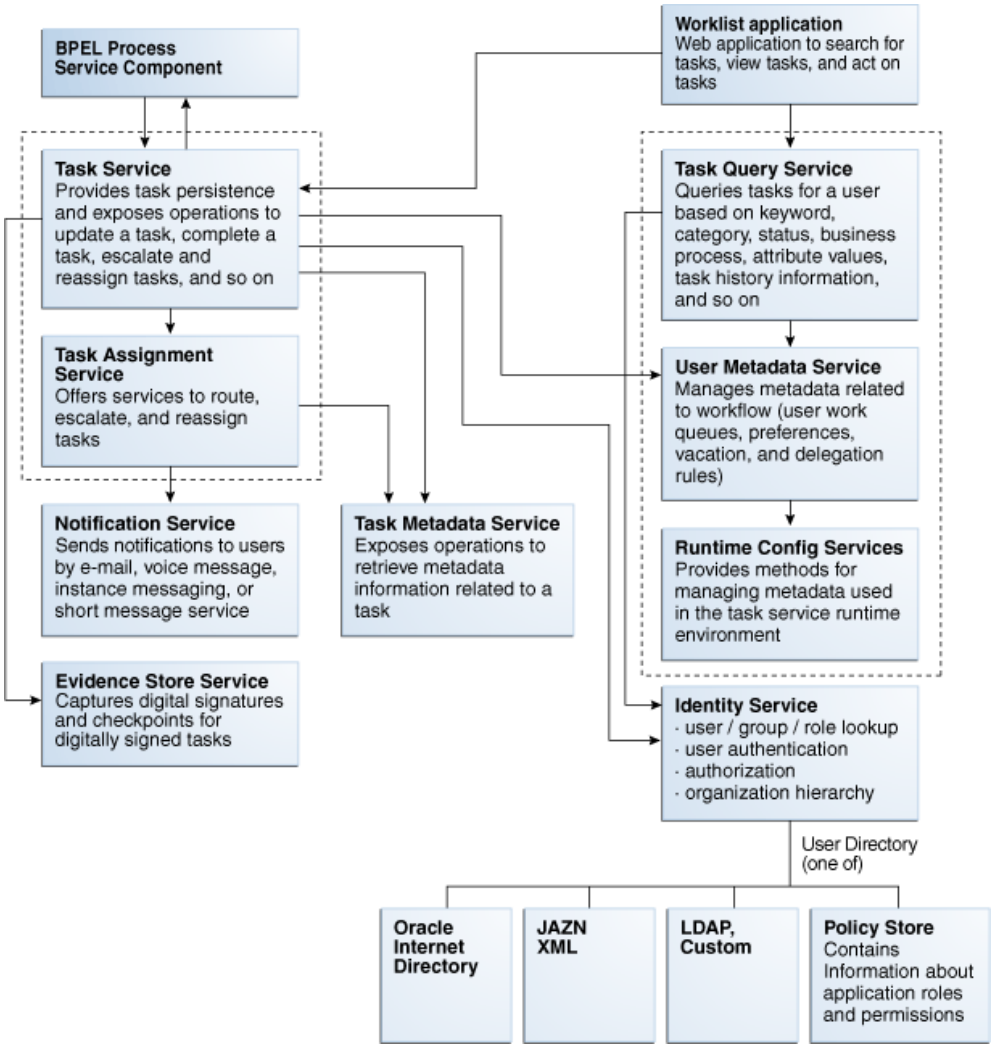


Figure 27-7 shows the interactions between the services and the business process.

Figure 27-7 Workflow Services and Business Process Interactions



Use of Human Task

You can use a human task in the following ways:

- Human task associated with a BPEL process

  You can associate your human task with a BPEL process. The BPEL process integrates a series of activities (including the human task activity) and services into an end-to-end process flow.

- Human task associated with a BPMN process

  You can associate your human task with a BPMN process. The BPMN process may contain other types of BPMN flow objects as part of the flow of the process. The human task is the implementation of a BPMN user task.

- Standalone human task

  You can also create the human task as a standalone component only in the SOA Composite Editor and not associate it with a BPEL process. Standalone human task service components are useful for environments in which there is no need for any

automated activity in an application. In the standalone case, the client can create the task themselves.

Service Engines

During runtime, the business logic and processing rules of the human task service component are executed by the human workflow service engine. Each service component (BPEL process, human workflow, decision service (business rules), and Oracle Mediator) has its own service engine container for performing these tasks. All human task service components, regardless of the SOA composite application of which they are a part, are executed in this single human task service engine.

For more information about configuring, monitoring, and managing the human workflow service engine, see *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

Human Workflow and Business Rule Differences Between Oracle SOA Suite and Oracle BPM Suite

Oracle SOA Suite and Oracle Business Process Management (BPM) Suite both provide support for business rules and human workflow. However, Oracle BPM Suite provides additional business rules and human workflow features that are not available in Oracle SOA Suite.

Table 27-2 identifies which business rule and human workflow features are supported in each suite.

Table 27-2 Business Rule and Human Workflow Features in Oracle SOA Suite and Oracle BPM Suite

Feature	Supported in Oracle BPM Suite?	Supported in Oracle SOA Suite?
Workspaces, process tracking, standard dashboards, case management, and applications menu	Yes	No
Approval groups (participant list)	Yes	No
Human workflow and business rules (participant list, routing rules)	Yes	Yes
Verbal rules	Yes	No
Rules business phrases	Yes	No
Oracle BPM Composer - design time rules editing	Yes	No
Process asset catalog (PAM) for source management between Oracle BPM Studio and Oracle BPM Composer	Yes	No
Rules testing in both Oracle JDeveloper and SOA Composer with usability enhancements	Yes	Yes
Microsoft Excel import/export for rules decision tables	Yes	Yes

For more information about Oracle BPM Suite, see *Developing Business Processes with Oracle Business Process Management Studio*.

28

Designing Human Tasks in Oracle BPM

This chapter describes how to design human tasks using the different editors available in Oracle BPM. It also describes how associate human tasks with the user tasks in your BPM project.

This chapter includes the following sections:

- Introduction to Designing Human Tasks in Oracle BPM
- Creating a Human Task from Oracle BPM Studio
- Editing a Human Task from Oracle BPM Studio
- Creating a Human Task from the SOA Composite Editor
- Implementing a User Task with an Existing Human Task
- Editing a Human Task Using the Human Task Editor
- Configuring a Human Task Using the Human Task Editor

For information on how Oracle BPM shows human tasks in the Business Catalog, see Working with Human Tasks.

For more information on how to define human tasks using Oracle SOA Suite, see the following chapters in *Developing SOA Applications with Oracle SOA Suite*:

- *Designing Human Tasks*
- *Designing Task Forms for Human Tasks*

Introduction to Designing Human Tasks in Oracle BPM

Human tasks enable you to model the interaction with the end user in a a BPM process. You must use human tasks to implement the user tasks in your process.

Oracle BPM Suite provides different editors that you can use according to the requirements of the Human Task you are modeling.

Some human tasks features are only available when using them from Oracle BPM Suite. For more information about this, see Configuring a Human Task Using the Human Task Editor.

Typical Design Workflow

There are different approaches to working with Human Tasks in Oracle BPM:

- Creating the human task using the Human Task editor
- Creating the human task using the simplified interface Oracle BPM provides

- Use an existing human task

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.

Creating the Human Task Using the User Task Properties Dialog

- Create a BPMN process
- Add a user task
- From the user task implementation properties dialog box, create a Human Task

  For more information see Creating a Human Task from Oracle BPM Studio.
- Create the corresponding taskflows using SOA Suite

Creating the Human Task Using the Human Task Editor:

- Create a Human Task from the SOA Composite Editor

  For more information see Creating a Human Task from the SOA Composite Editor.
- 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

  For more information see Implementing a User Task with an Existing Human Task .

Using an existing Human Task:

- Create a BPMN process
- Add a user task.
- In the user task implementation properties dialog box, select the existing Human Task.

  For more information, see How to Implement a User Task With an Existing Human Task .

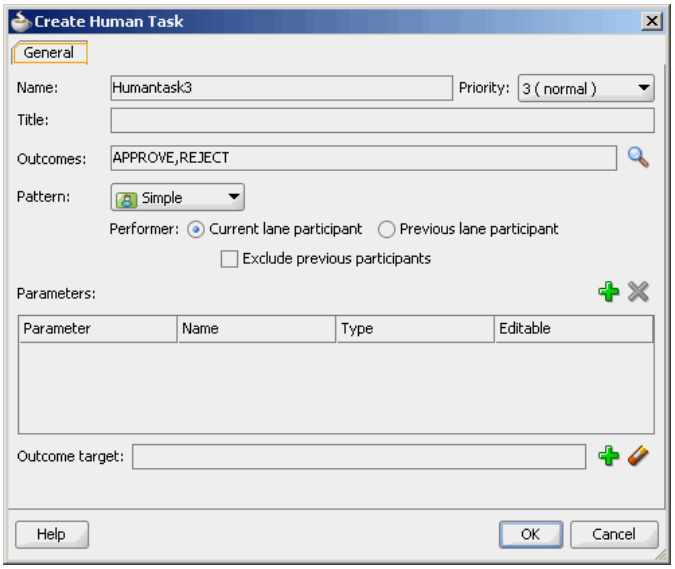
Creating a Human Task from Oracle BPM Studio

You can create a simple Human Task using Oracle BPM Studio. The simplified interface that Oracle BPM Studio provides hides the complexity of the Human Task editor by exposing the most important fields to configure a Human Task used in a business process.

After you create the Human Task using the Create Human Task dialog box, you can edit it using the Human Task editor if needed.

Figure 28-1 shows the Create Human Task dialog box.

Figure 28-1 Create Human Task Dialog



The Create Human Task dialog box enables you to define the following properties:

- **Title**

  Defines the name of the Human Task that is displayed to end-users in the Oracle Process Workspace and Oracle BPM 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.

- **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 value of this outcome is one of the values defined in the Outcomes property.

How to Create a Human Task from Oracle BPM Studio

You can create a Human Task from the User Task Properties dialog box 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 box appears.

   Figure 28-2 shows the Properties - User Task dialog box.
4. Click the **Implementation** tab.
5. Click the **Add** button next to the Human Task field.

   The Create Human Task dialog box appears.

   Figure 28-1 shows the Create Human Task dialog box.
6. In the name field, enter a name to identify the Human Task.
7. From they Priority List, select a priority.
8. Select a Human Task pattern appropriate for your implementation.

   For more information about Human Task patterns, see Using Approval Management.
9. In the Title Field, enter a title for the task to display in the client application (Process Workspace and others).
10. Optionally, you can configure the following:

    - The outcome

      See How to Configure the Outcome of a Human Task for information on how to configure the outcome of a Human Task.

    - The parameters

      See How to Add a Parameter to Human Task for information on how to configure the outcome of a Human Task.

    - The outcome target

      See 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 box closes and the Human Task field in the User Task Properties dialog box shows the Human Task you created.
12. Click **OK**.

    The User Task Properties closes and saves the implementation you configured for the user task.

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 BPM Worklist.

To configure the outcome of a Human Task:

1. In the Create Human Task dialog box, click the **Browse** button next to the Outcomes field.

   The Outcomes dialog box 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 box closes and the selected outcomes appear in the Create Human Task dialog box, in the Outcomes field.

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 box, click the **Add** button in the Parameters table.

   The Data Objects dialog box appears.

2. Select a data object from the Data Objects dialog box and drop it on the Parameters table.

   The selected data object appears in the Parameters table.

3. Close the **Data Objects** dialog box.

4. Optionally you can mark the parameter as editable by selecting the Editable column in the Parameters table.

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.You can base the flow of your process on the value of the outcome target using an exclusive gateway.

To configure the outcome target of a Human Task

1. In the Create Human Task dialog box, click the **Add** button next to the Outcome Target field.

   The Data Objects dialog box appears.

2. Select a String data object from the Data Objects dialog box and drop it on the Outcome Target field.

   To add a new data object, right-click the **Data Objects** node and select **Add**.

The selected data object appears in the Outcome Target field.

3. Close the **Data Objects** dialog box.

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.

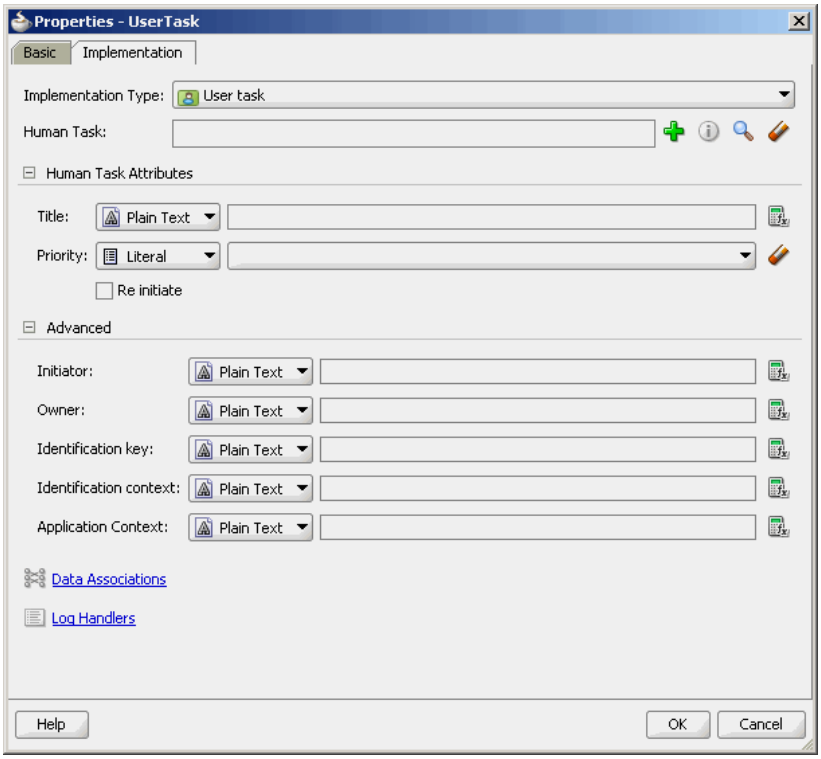
You can edit the created Human Task using the Human Task editor to configure implementation details.

Editing a Human Task from Oracle BPM Studio

You can edit a Human Task using the User Task Properties dialog box or the Human Task editor. Generally you use the Human Task editor for complex human tasks.

Figure 28-2 shows the User Task Properties dialog box.

Figure 28-2 User Task Properties dialog box



The User Task Properties dialog box enables you to define properties using plain text, simple expressions and XPATH expressions:

- **Title**

  Defines the name of the Human Task that is displayed to end-users in the Oracle Process Workspace and Oracle BPM 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.

- **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.

How to Edit a Human Task Using the User Task Properties Dialog

To edit a Human Task using the User Task Properties dialog box:

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.

Creating a Human Task from the SOA Composite Editor

You can add a Human Task to your BPM project from the SOA Composite editor.

Typically you do this when you design human tasks before modeling the user tasks in a BPMN process.

How to Create a Human Task from the SOA Composite Editor

You can add a user task to a BPM project using the SOA Composite editor.

To create a Human Task from the SOA Composite Editor:

1. Select the **Application** view.
2. Expand the BPM project where you want to add the Human Task.
3. Expand the **SOA Content** node.
4. Double-click the composite.xml node to open the SOA Composite editor.
5. From the **Component Palette** grab a **Human Task**.
6. Drop the Human Task in the **Components area** of the SOA Composite.

   The Create Human Task dialog box appears.
7. Enter a name to identify the Human Task.
8. Optionally, modify the URL for the Human Task namespace.
9. Optionally, check the **Create Composite Service with SOAP Bindings** option.
10. Click **OK**.

    The Human Task component appears in the Component area of the SOA Composite.

What Happens When You Create a Human Task from the SOA Composite Editor

The Human Task you created is available to implement the user tasks in your BPM project. For more information on how to do this, see Implementing a User Task with an Existing Human Task .

Implementing a User Task with an Existing Human Task

You can create a Human Task using the 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 Handling Information in Your Process Design.

How to Implement a User Task With an Existing Human Task

You can implement a user task using an existing Human Task that you created for another user task or using the Human Task editor.

To implement a user task with an existing Human Task:

1. Open the BPMN process.
2. Right-click the user task.
3. Select **Properties**.

   The Properties - User Task dialog box appears.
4. Click the **Implementation** tab.

5. Click the **Browse** button next to the Human Task field.

   The Browse Human Tasks dialog box appears.

6. Select a Human Task from the list.

7. Click **OK**.

   The Browse Human Tasks dialog box closes and the selected Human Task appears in the Human Task field.

8. Click **OK**.

What Happens When You Implement a User Task With an Existing Human 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.

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 box 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.

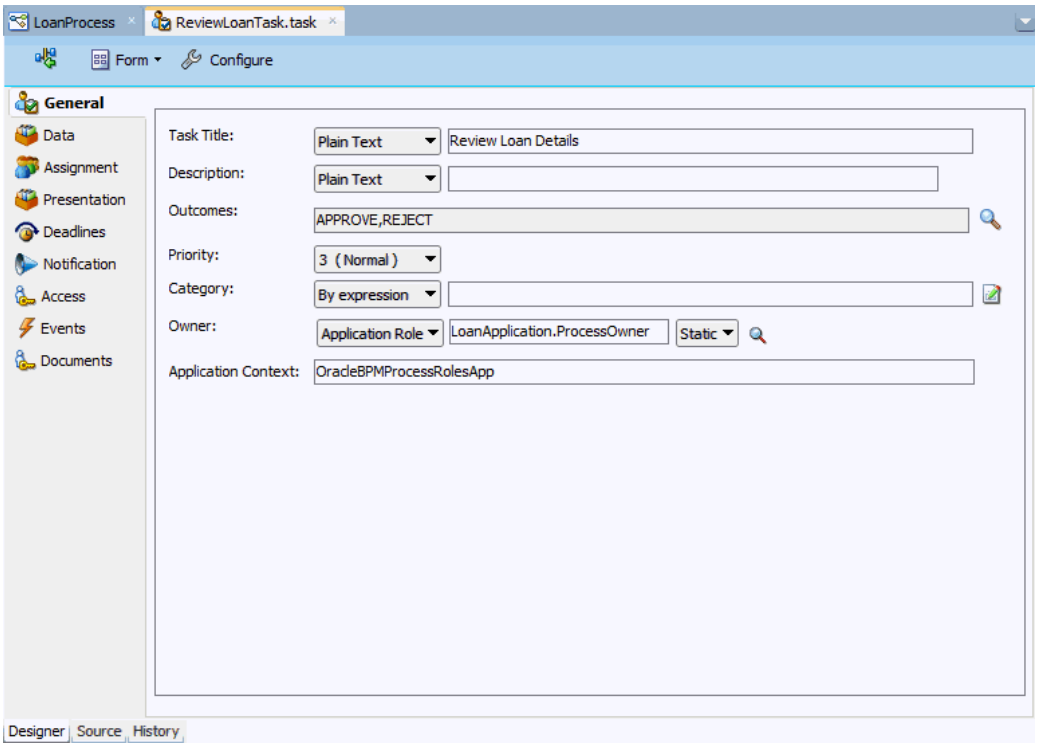
7. Select **Add other payload** from the list in the Add button.
8. Select the payload element in the Type Chooser dialog box.
9. In the Process editor, right-click the user task and select **Properties**.
10. Define the data associations between the process data object and the task payload.

Editing a Human Task Using the Human Task Editor

Configuring complex human tasks usually requires you to use the Human Task Editor to edit them. This allows you to edit properties that are not displayed when you use the simplified interface that Oracle BPM provides.

Figure 28-3 shows the Human Task editor.

Figure 28-3 Human Task Editor



How to Edit a Human Task Using the Human Task Editor

You can edit a Human Task used in your BPM project using the Human Task editor. Generally you use the Human Task editor to edit complex human tasks.

To edit a Human Task using the 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 Human Task editor appears.

4. Make changes to the Human Task.

Configuring a Human Task Using the Human Task Editor

This section covers how to configure those properties that are only available when using human tasks from Oracle BPM Suite.

The rest of the properties are shared with Oracle SOA Suite. For more information on how to configure these properties, see *Creating Human Tasks* in *Developing SOA Applications with Oracle SOA Suite*.

The Human Task editor enables you to configure the following sets of properties:

- **General**

  Enables you to define basic information such as the title, description, priority and owner.

  Note that you can localize the title of a Human Task by selecting the Translation option from the list next to the title field and then clicking Build and Internationalized title. For more information on how to define the resource bundle, see Specifying Multilingual Settings and Style Sheets.

  For more information on how to configure these properties, see How to Specify the Title_ Description_ Outcome_ Priority_ Category_ Owner_ and Application Context.

- **Data**

  Enables you to define the message elements that compose the structure of the task payload.

  For more information on how to configure these properties, see How to Specify the Task Payload Data Structure.

- **Assignment**

  Enables you to assign a participant to the task and to configure routing policies to drive the task through the defined workflow.

  For more information on how to configure these properties, see Assigning Task Participants.

- **Presentation**

  Enables you to configure the presentation used to display the Human Task, using stylesheets and multilingual settings.

  For more information on how to configure these properties, see Specifying Multilingual Settings and Style Sheets.

- **Deadlines**

  Enables you to specify the duration and expiration of a task.

  For more information on how to configure these properties, see Escalating_ Renewing_ or Ending the Task.

- **Notification**

  Enables you to configure how to notify the user when the status of the task changes.

For more information on how to configure these properties, see Specifying Participant Notification Preferences.

For information on how to specify an email address for the recipient of the notification, see How to Specify an E-mail Address for the Recipient of a Notification.

- **Access**

  Enables you to configure access policies and restrictions for the content of the Human Task.

  For more information on how to configure these properties, see Specifying Access Policies and Task Actions on Task Content.

- **Events**

  Enables you to specify how to handle BPEL callbacks.d

  For more information on how to configure these properties, see Specifying Java or Business Event Callbacks.

- Documents

  Enables you to configure the Human Task to store task attachments in Oracle UCM Repository. For more information, see How to Configure Oracle UCM Repository to Store Task Attachments.

Some human tasks features are only available when using humans tasks from an Oracle BPM Suite Installation.

When you use human tasks from an Oracle BPM Suite installation you must take into account the following considerations:

- When you create a Human Task using Oracle BPM Suite, the *enableAutoClaim* property is set to true by default.
- The owner property in the process context is set using the participant assigned to the user task that contains the Human Task.
- If your process contains an initiator task the creator attribute in the process context is automatically set using the participant assigned to the initiator task.
- When you specify the completion criteria for a parallel participant you can set the default outcome as one of the previous outcomes.

How to Specify an E-mail Address for the Recipient of a Notification

When using the Human Task editor in a BPM Suite installation, you can specify an email address for the recipient of a Notification.

To specify an email 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 email address.

   Optionally you can use the buttons next to the Recipient text field to look up the email address in an application server or to specify the email address using XPath.

   Note:

   When sending a notification to a recipient specified using an email address, the notification service uses the user context of an assignee to obtain the task information to include in the notification.

How to Configure Oracle UCM Repository to Store Task Attachments

You can configure Human Tasks to store attachments in the UCM repository. These attachments may contain one or more metadata properties. You can assign values to these properties or configure them to allow the user to provide the value.

Note: When a file is attached from the Process Tracking page, it goes to the BPM repository and not to the UCM repository. Hence, even if the UCM repository is unavailable, the Process Tracking page allows you to upload files. However, when the UCM repository is configured and Human Tasks are designed to upload attachment to the UCM repository, the Tasks page uploads files to the UCM repository. However, the Process Tracking page always uploads the files to BPM repository.

To configure Oracle UCM Repository for task attachments:

1. In the **Project Navigator** tree, expand the Business Catalog node.
2. Expand the **Human Tasks** node.
3. Double-click the Human Task you want to configure.

   The Human Task Editor appears.
4. Click the **Documents** tab.
5. Select **Use Document Package**.

   A section to configure metadata properties appears. The table already contains the mandatory standard metadata: Security Group and Document Type.
6. Optionally add new standard or custom metadata properties:
   a. Click the **Add** button.
   b. Click the **Name** column to select a standard property from the list or to enter a custom name.
   c. Click the **Value** column to assign a value to the property. Select **By Name** to provide the text to assign the value to the property, or **By Expression** to provide an expression.
   d. Click the Display column to select a display mode:

      Editable: the user can provide a value in the task form when uploading the attachment.

Hidden: the value does not appear in the task form

Read-Only: the value appears in the task form but the user cannot modify it

---

Note:

Custom metadata does not appear in the task form, so you must map the value to a task payload or provide a static value.

Working with Screenflows

Screenflows enable users to work on all of the tasks in a process in sequence.

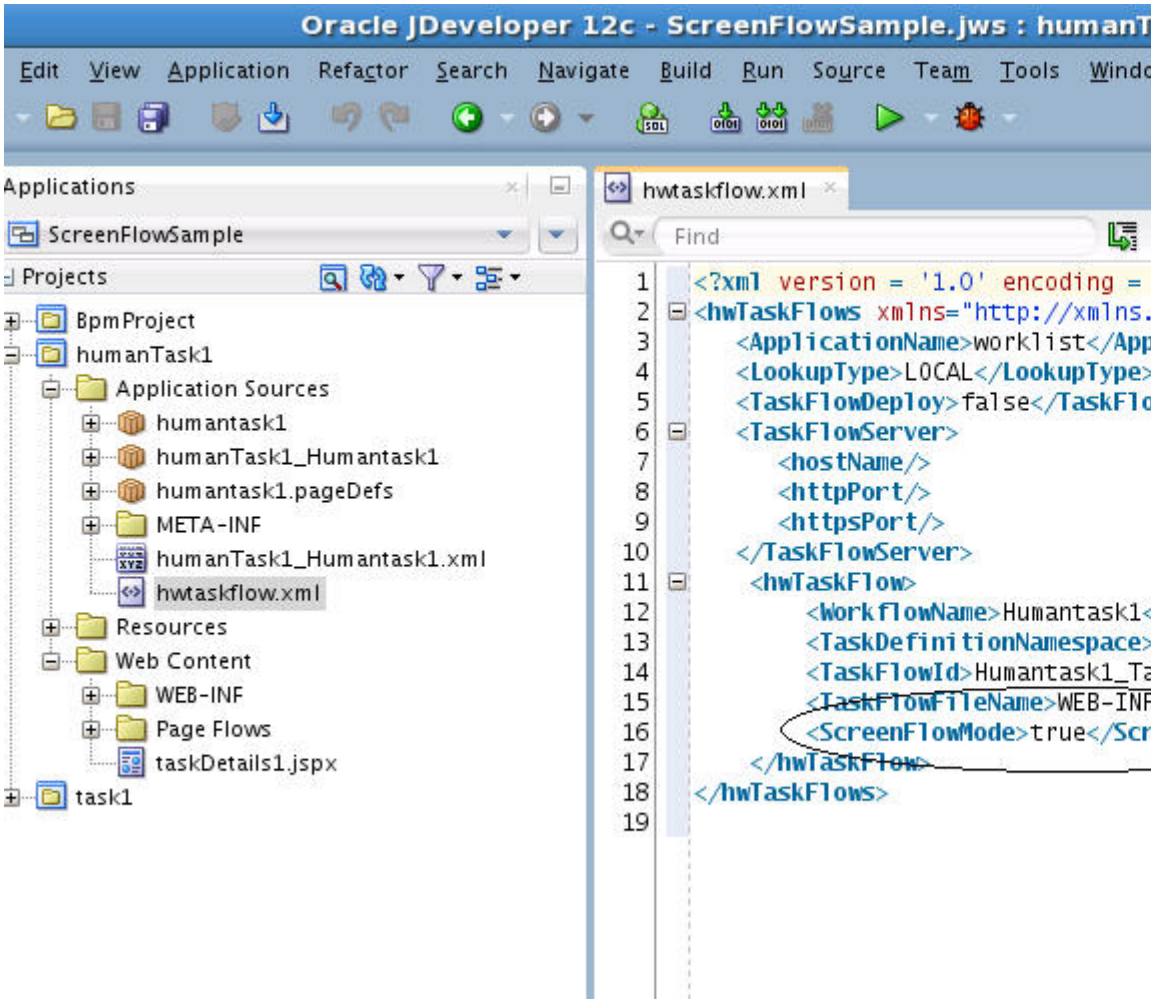
For example, when a user completes a task in a process, and if the next task is assigned to the same user, then the next task is automatically shown in the same flow, without the user having to return to the task list.

Creating a Screenflow

Screenflows are created in HumanTask projects to create a task that automatically flows from screen to screen.

To create a screenflow, add a **ScreenFlowMode** entry in the HumanTask project before deploying it. Add this entry to all of the Human Tasks that are to be part of the screenflow, except for the last task of the screenflow. After creating a screenflow in BPM Studio, enable it in Enterprise Manager, and then test it in Workspace.

1. In BPM Studio, add an entry for ScreenFlowMode in the hwtaskflow.xml file.



2. In Enterprise Manager, enable the Process Broker service:

   a. Log in to Enterprise Manager as an administrator, and navigate to soa-infra > Administration > System MBean Browser.

   b. Navigate to Application Defined MBeans > oracle.as.soainfra.config > BPMNConfig > bpmn or search for the **disableProcessBroker** parameter.

   c. Set the **disableProcessBroker** parameter to false and click **Apply**.

   d. Restart the SOA/BPM WebLogic server.

29

Configuring Human Tasks

This chapter describes how to configure the different properties of a human task. It covers basic properties, task payload data structure, participant assignment, routing policies, localization, escalation, notification preferences, access policies and task actions, restrictions and Java and business event callbacks.

This chapter includes the following sections:

- Accessing the Sections of the Human Task Editor
- Specifying the Title_ Description_ Outcome_ Priority_ Category_ Owner_ and Application Context
- Specifying the Task Payload Data Structure
- Assigning Task Participants
- Selecting a Routing Policy
- Specifying Multilingual Settings and Style Sheets
- Specify What to Show in Task Details in the Worklist
- Escalating_ Renewing_ or Ending the Task
- Specifying Participant Notification Preferences
- Specifying Access Policies and Task Actions on Task Content
- Creating and Implementing Digital Certificates
- Specifying Restrictions on Task Assignments
- Specifying Java or Business Event Callbacks
- Storing Documents in Oracle Enterprise Content Management

For information about troubleshooting human workflow issues, see section “Human Workflow Troubleshooting" of *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

Accessing the Sections of the Human Task Editor

This section describes how to access the sections of the Human Task Editor.

Brief descriptions are provided of each section and references are provided to more specific information.

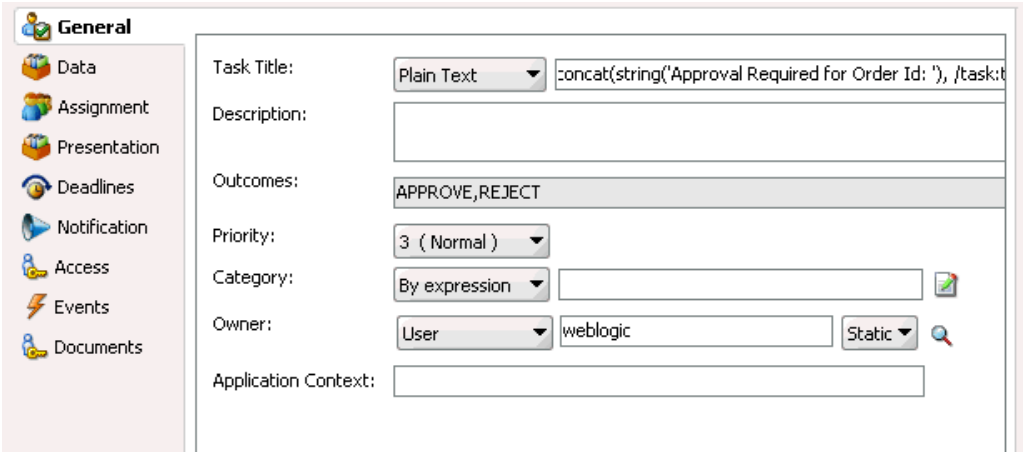
How to Access the Sections of the Human Task Editor

To access the sections of the Human Task Editor:

1. Double-click the **Human Task** icon in the SOA Composite Editor or double-click the Human Task icon in Oracle BPEL Designer.

   The Human Task Editor consists of the main sections shown on the left side in Figure 29-1. These sections enable you to design the metadata of a human task.

Figure 29-1 Human Task Editor



Instructions for using these main sections of the Human Task Editor to create a workflow task are listed in Table 29-1.

Table 29-1 Human Task Editor

Section	Description	See...
General (title, description, outcomes, category, priority, owner, and application context)	Enables you to define task details such as title, task outcomes, owner, and other attributes.	Specifying the Title_ Description_ Outcome_ Priority_ Category_ Owner_ and Application Context
Data	Enables you to define the structure (message elements) of the task payload (the data in the task).	Specifying the Task Payload Data Structure
Assignment	Enables you to assign participants to the task and create a policy for routing the task through the workflow.	Assigning Task Participants Selecting a Routing Policy
Presentation	Enables you to specify the following settings: • Multilingual settings • WordML and custom style sheets for attachments	Specifying Multilingual Settings and Style Sheets

Table 29-1 (Cont.) Human Task Editor

Section	Description	See...
Deadlines	Enables you to specify the expiration duration of a task, custom escalation Java classes, and due dates.	Escalating_ Renewing_ or Ending the Task
Notification	Enables you to create and send notifications when a user is assigned a task or informed that the status of the task has changed.	Specifying Participant Notification Preferences
Access	Enables you to specify access rules for task content and task actions, workflow signature policies, and assignment restrictions.	Specifying Access Policies and Task Actions on Task Content How to Specify a Workflow Digital Signature Policy Specifying Restrictions on Task Assignments
Events	Enables you to specify callback classes and task and routing assignments in BPEL callbacks.	Specifying Java or Business Event Callbacks
Documents	-	Storing Documents in Oracle Enterprise Content Management

Specifying the Title, Description, Outcome, Priority, Category, Owner, and Application Context

This section enables you to specify details such as the task title, description, task outcomes, task category, task priority, and task owner.

This section contains these topics:

- How to Specify the Title_ Description_ Outcome_ Priority_ Category_ Owner_ and Application Context
- How to Specify a Task Title
- How to Specify a Task Description
- How to Specify a Task Outcome
- How to Specify a Task Priority
- How to Specify a Task Category
- How to Specify a Task Owner
- How To Specify an Application Context

How to Specify the Title, Description, Outcome, Priority, Category, Owner, and Application Context

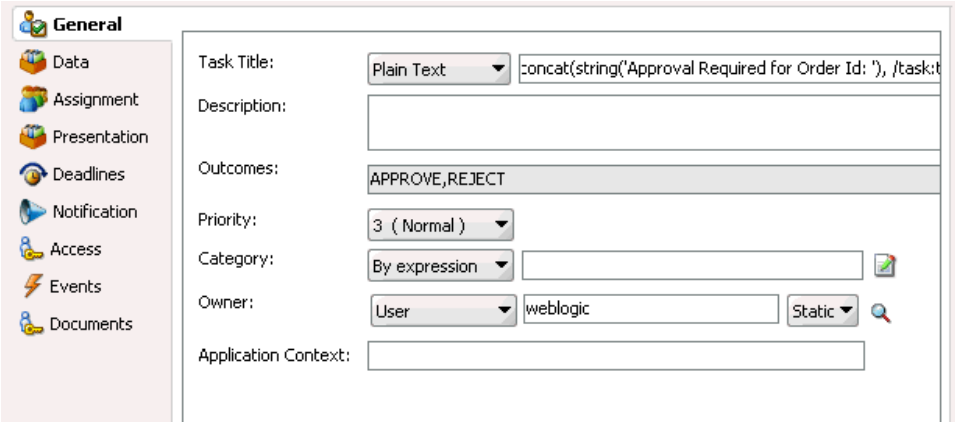
To specify the title, description, outcome, priority, category, owner, and application context:

1. Click the **General** tab.

   Figure 29-2 shows the **General** section of the Human Task Editor.

   This section enables you to specify details such as the task title, description, task outcomes, task category, task priority, and task owner.

Figure 29-2 Human Task Editor — General Section



Instructions for configuring the following subsections of the **General** section are listed in Table 29-2:

Table 29-2 Human Task Editor — General Section

For This Subsection...	See...
Title	How to Specify a Task Title
Description	How to Specify a Task Description
Outcomes	How to Specify a Task Outcome
Priority	How to Specify a Task Priority
Category	How to Specify a Task Category
Owner	How to Specify a Task Owner
Application Context	How To Specify an Application Context

How to Specify a Task Title

To specify a task title:

Enter an optional task title. The title defaults to this value only if the initiated task does not have a title set in it. The title provides a visual identifier for the task. The task title

displays in Oracle BPM Worklist. You can also search on titles in Oracle BPM Worklist.

1. In the **Task Title** field of the **General** section, select a method for specifying a task title:

   - **Plain Text**: Manually enter a name (for example, `Vacation Request Approved`).
   - **Text and XPath**: Enter a combination of manual text and a dynamic expression. After manually entering a portion of the title (for example, `Approval Required for Order Id:`), place the cursor one blank space to the right of the text and click the icon to the right of this field. This displays the Expression Builder for dynamically creating the remaining portion of the title. After completing the dynamic portion of the name, click **OK** to return to this field. The complete name is displayed. For example:

     ```
     Approval Required for Order Id: <%/task:task/task:payload/task:orderId%>
     ```

     The expression is resolved during runtime with the exact order ID value from the task payload.
   - **Translation**: Click the Lookup button and locate a translation bundle to use to specify the title.
   - **Resource Xpath**: Click the Lookup button and locate a resource bundle to use to specify the title.

   If you entered a title in the **Task Title** field of the **General** tab of the Create Human Task dialog box, the title you enter here is overridden. See the section on specifying task titles in *Developing SOA Applications with Oracle SOA Suite*.

How to Specify a Task Description

You can optionally specify a description of the task in the **Description** field of the **General** section. The description enables you to provide additional details about a task. For example, if the task title is `Computer Upgrade Request`, you can provide additional details in this field, such as the model of the computer, amount of CPU, amount of RAM, and so on. The description does not display in Oracle BPM Worklist.

To add a task description:

1. Select the drop-down menu and choose either **Plain Text** or **Translation**.
2. Provide the description:

   Plain text:

   a. Type a description into the dialog box.

   b. Click **Ok**.

   Translation:

   a. Click the **Lookup** button.

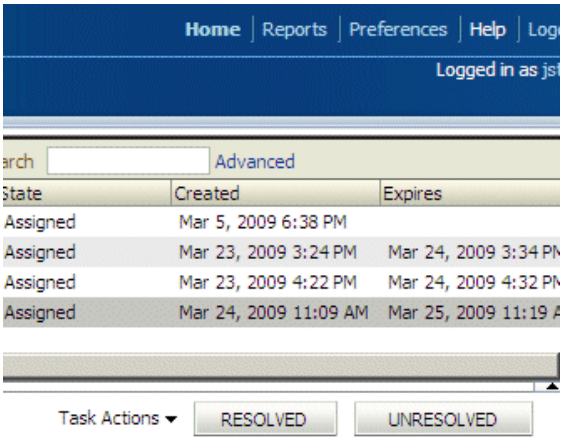
   b. Locate a resource bundle and provide a description.

   c. Click **Ok**.

How to Specify a Task Outcome

Task outcomes capture the possible outcomes of a task. Oracle BPM Worklist displays the outcomes you specify here as the possible task actions to perform during runtime. Figure 29-3 provides details.

Figure 29-3 Outcomes in Oracle BPM Worklist



You can specify the following types of task outcomes:

- Select a seeded outcome
- Enter a custom outcome

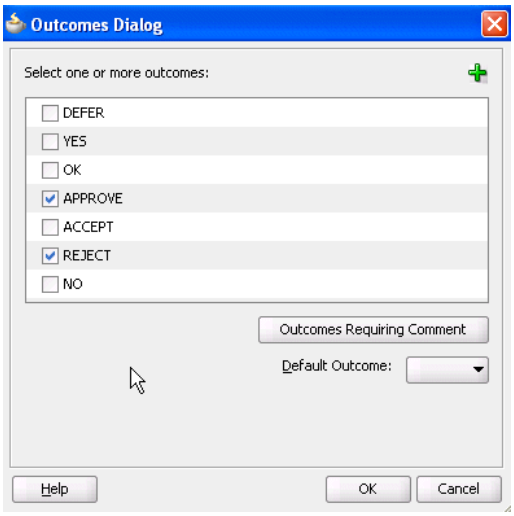
The task outcomes can also have runtime display values that are different from the actual outcome value specified here. This permits outcomes to be displayed in a different language in Oracle BPM Worklist. For more information about internationalization, see How to Specify Multilingual Settings.

To specify a task outcome:

1. To the right of the **Outcomes** field in the **General** section, click the **Search** icon.

   The Outcomes dialog box shown in Figure 29-4 displays the possible outcomes for tasks. **APPROVE** and **REJECT** are selected by default.

Figure 29-4 Outcomes Dialog



2. Enter the information shown in Table 29-3.

Table 29-3 Outcomes Dialog

Field	Description
Select one or more outcomes	Select additional task outcomes or deselect the default outcomes.
Add icon	Click to invoke a dialog box for adding custom outcomes. In the **Name** field of the dialog box, enter a custom name, and click **OK**. Your outcome displays in the Outcomes field. **Notes**: Be aware of the following naming restrictions: • Do not specify a custom name that matches a name listed in the **Actions** tab of the **Access** section of the Human Task Editor (for example, do not specify `Delete`). Specifying the same name can cause problems at runtime. • Do not specify a custom name with blank spaces (for example, `On Hold`). This causes an error when the custom outcome is accessed in Oracle BPM Worklist. If you must specify an outcome with spaces, use a resource bundle. • A custom task outcome must begin with a letter of the alphabet, either upper or lower case. It should contain only letters of the alphabet and the numbers zero (0) through nine (9).
Outcomes Requiring Comment	Click to select an outcome to which an assignee adds comments in Oracle BPM Worklist at runtime. The assignee must add the comments and perform the action without saving the task at runtime.
Default Outcome	Select the default outcome for this outcome.

The seeded and custom outcomes selected here display for selection in the **Majority Voted Outcome** section of the parallel participant type.

3. For more information, see Specifying the Voting Outcome.

How to Specify a Task Priority

Specify the priority of the tasks. Priority can be **1** through **5**, with **1** being the highest. By default, the priority of a task is **3**. This priority value is overridden by any priority value you select in the **General** tab of the Create Human Task dialog box. You can filter tasks based on priority and create views on priorities in Oracle BPM Worklist.

To specify a task priority:

1. From the **Priority** list in the **General** section, select a priority for the task.

For more information about specifying a priority value in the Create Human Task dialog box, see *Developing SOA Applications with Oracle SOA Suite*.

How to Specify a Task Category

You can optionally specify a task category in the **Category** field of the **General** section. This categorizes tasks created in a system. For example, in a help desk environment, you may categorize customer requests as either software-related or hardware-related. The category displays in Oracle BPM Worklist. You can filter tasks based on category and create views on categories in Oracle BPM Worklist.

To specify a task category:

1. Select a method for specifying a task category in the **Category** field of the **General** section:
   - **By Name**: Manually enter a name.
   - **By Expression**: Click the icon to the right of this field to display the Expression Builder for dynamically creating a category.
   - **Translation**: If the composite contains a resource bundle file, then use the Lookup button to locate the resource bundle file and to specify a category.

How to Specify a Task Owner

The task owner can view the tasks belonging to business processes they own and perform operations on behalf of any of the assigned task participant types. Additionally, the owner can also reassign, withdraw, or escalate tasks. The task owner can be considered the business administrator for a task. The task owner can also be specified in the **Advanced** tab of the Create Human Task dialog box described in the section on specifying a task owner in *Developing SOA Applications with Oracle SOA Suite*. The task owner specified in the **Advanced** tab overrides any task owner you enter here.

For more information about the task owner, see Task Stakeholders.

To specify a task owner:

1. Select a method for specifying the task owner:
   - Statically through the identity service user directory or the list of application roles

- Dynamically through an XPath expression

  For example:

  – If the task has a payload message attribute named `po` within which the `owner` is stored, you can specify an XPath expression such as: `/task:task/task:payload/po:purchaseOrder/po:owner`

  – `ids:getManager('jstein', 'jazn.com')`

    The manager of `jstein` is the task owner.

For more information about users, groups, and application roles, see Participant Assignment.

Specifying a Task Owner Statically Through the User Directory or a List of Application Roles

Task owners can be selected by browsing the user directory (Oracle Internet Directory, Java AuthoriZatioN (JAZN)/XML, LDAP, and so on) or a list of application roles configured for use with Oracle SOA Suite.

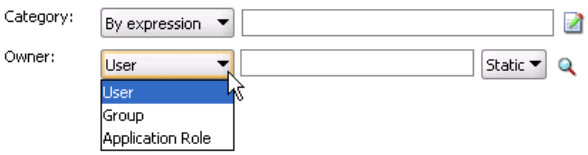
To specify a task owner statically through the user directory or a list of application roles:

1. In the first list to the right of the **Owner** field in the **General** section, select **User**, **Group**, or **Application Role** as the type of task owner. Figure 29-5 provides details.

   Note:

   By default, group names in human tasks are case sensitive. Therefore, if you select **Group** and enter a name in upper case text (for example, `LOANAGENTGROUP`), no task is displayed under the **Administrative Tasks** tab in Oracle BPM Worklist. To enable group names to be case agnostic (case insensitive), you must set the **caseSensitiveGroups** property to `false` in the System MBeans Browser. For information, see *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

   Figure 29-5 Specify a Task Owner By Browsing the User Directory or Application Roles

2. In the second list to the right of the **Owner** field in the **General** section, select **Static**.

3. See the step in Table 29-4 based on the type of owner you selected.

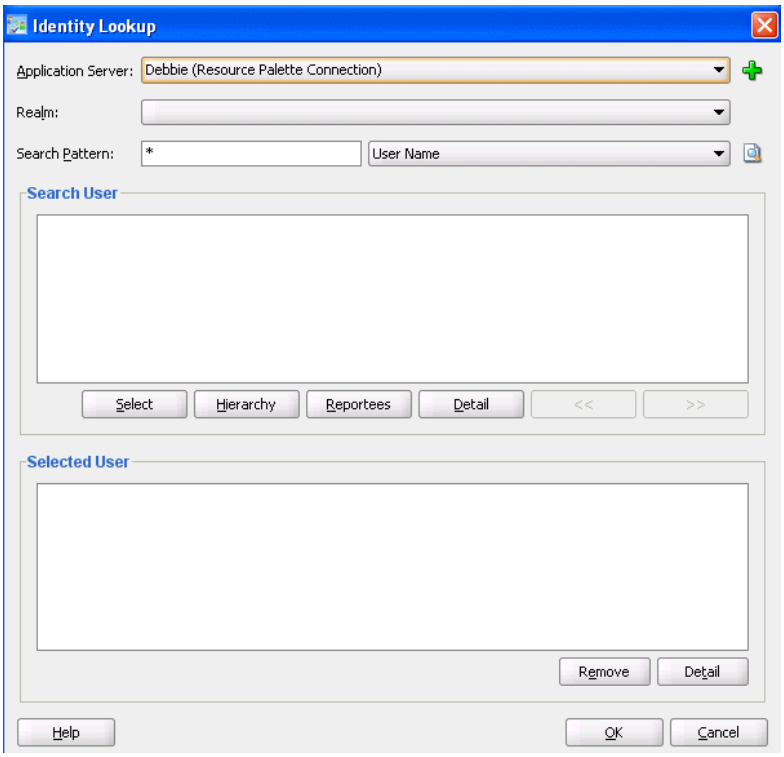
   Table 29-4 Type of Owner

Table 29-4 (Cont.) Type of Owner

If You Selected...	See Step...
User or **Group**	4
Application Role	5

4. If you selected **User** or **Group**, the Identity Lookup dialog box shown in Figure 29-6 appears.

Figure 29-6 Identity Lookup Dialog



To select a user or group, you must first create an application server connection by clicking the **Add** icon. Note the following restrictions:

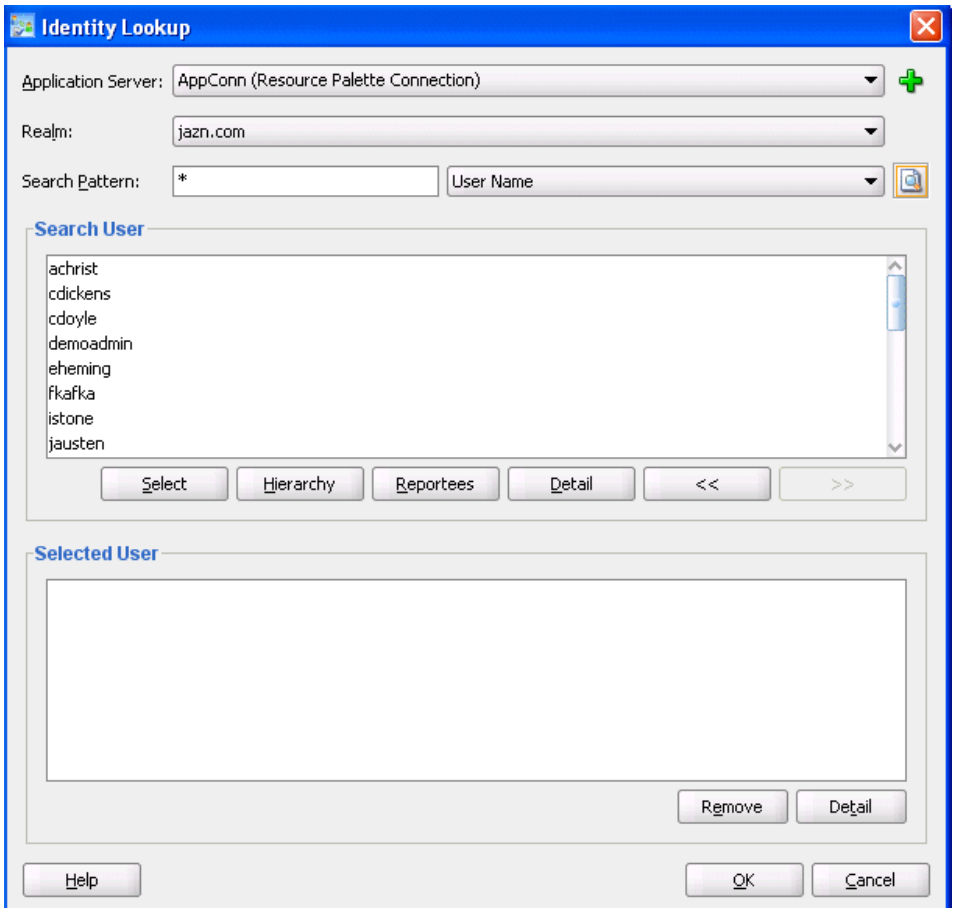
- Do not create an application server connection to an Oracle WebLogic Administration Server from which to retrieve the list of identity service realms. This is because there is no identity service running on the Administration Server. Therefore, no realm information displays and no users display when performing a search with a search pattern in the Identity Lookup dialog box. Instead, create an application server connection to a managed Oracle WebLogic Server.
- You must select an application server connection configured with the complete domain name (for example, `myhost.us.example.com`). If you select a connection configured only with the hostname (for example, `myhost`), the **Realm** list may not display the available realms. If the existing connection does not include the domain name, perform the following steps:
  - In the **Resource Palette**, right-click the application server connection.
  - Select **Properties**.

- In the **Configuration** tab, add the appropriate domain to the hostname.
- Return to the Identity Lookup dialog box and reselect the connection.

a. Select or create an application server connection to display the realms for selection. A realm provides access to a policy store of users and roles (groups).

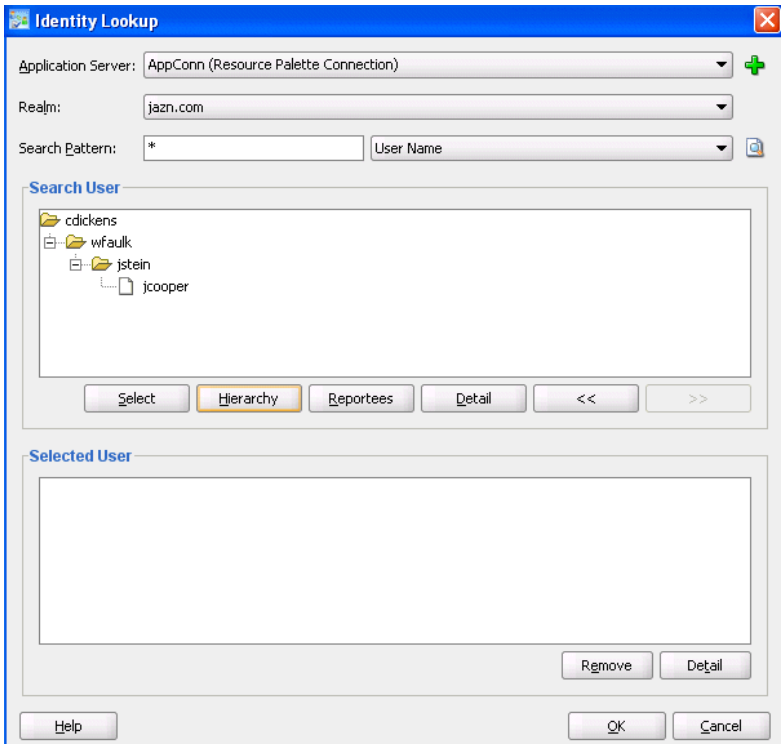
b. Search for the owner by entering a search string such as `jcooper`, `j*`, `*`, and so on. Clicking the **Lookup** icon to the right of the **User Name** field fetches all the users that match the search criteria. Figure 29-7 provides details. One or more users or groups can be highlighted and selected by clicking **Select**.

Figure 29-7 Identity Lookup with Realm Selected



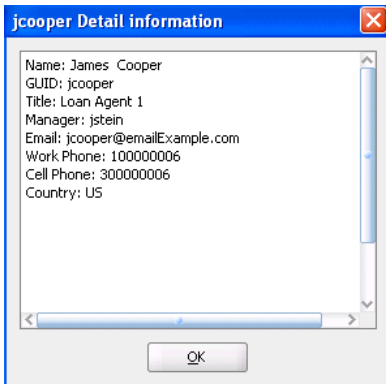
c. View the hierarchy of a user by highlighting the user and clicking **Hierarchy**. Similarly, clicking **Reportees** displays the reportees of a selected user or group. Figure 29-8 provides details.

Figure 29-8 User Hierarchy in Identity Lookup Dialog



d. View the details of a user or group by highlighting the user or group and clicking **Detail**. Figure 29-9 provides details.

Figure 29-9 User or Group Details



e. Click **OK** to return to the Identity Lookup dialog box.

f. Click **Select** to add the user to the **Selected User** section.

g. Click **OK** to return to the Human Task Editor.

Your selection displays in the **Owner** field.

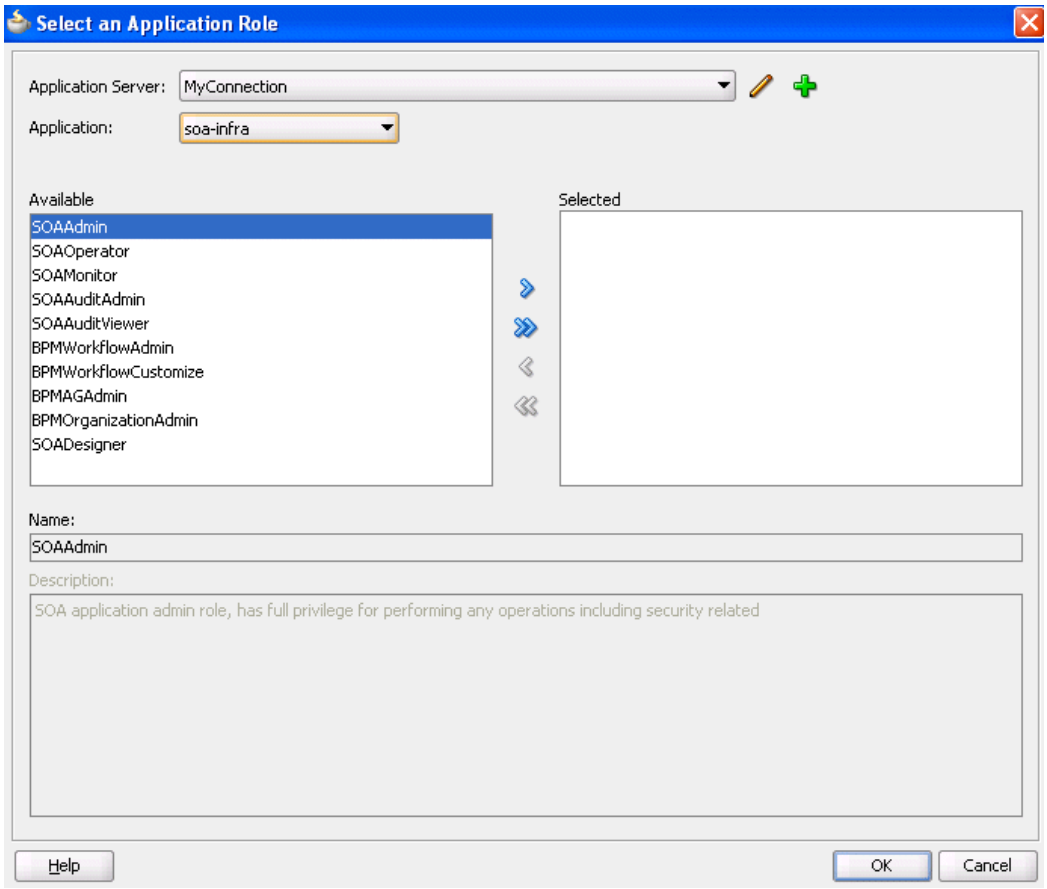
5. If you selected **Application Role**, the Select an Application Role dialog box appears.

a. In the **Application Server** list, select the type of application server that contains the application role or click the **Add** icon to launch the Create Application Server Connection wizard to create a connection.

b. In the **Application** list, select the application that contains the application roles (for example, a custom application or **soa-infra** for the SOA Infrastructure application).

c. In the **Available** section, select appropriate application roles and click the > button. To select all, click the >> button. Figure 29-10 provides details.

Figure 29-10 Application Role



d. Click **OK**.

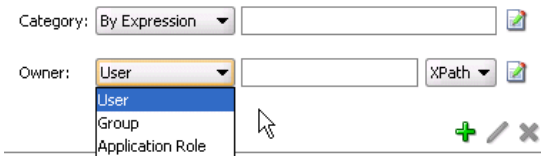
Specifying a Task Owner Dynamically Through an XPath Expression

Task owners can be selected dynamically in the Expression Builder dialog box.

To specify a task owner dynamically:

1. In the first list to the right of the **Owner** field in the **General** section, select **User**, **Group**, or **Application Role** as the type of task owner. Figure 29-11 provides details.

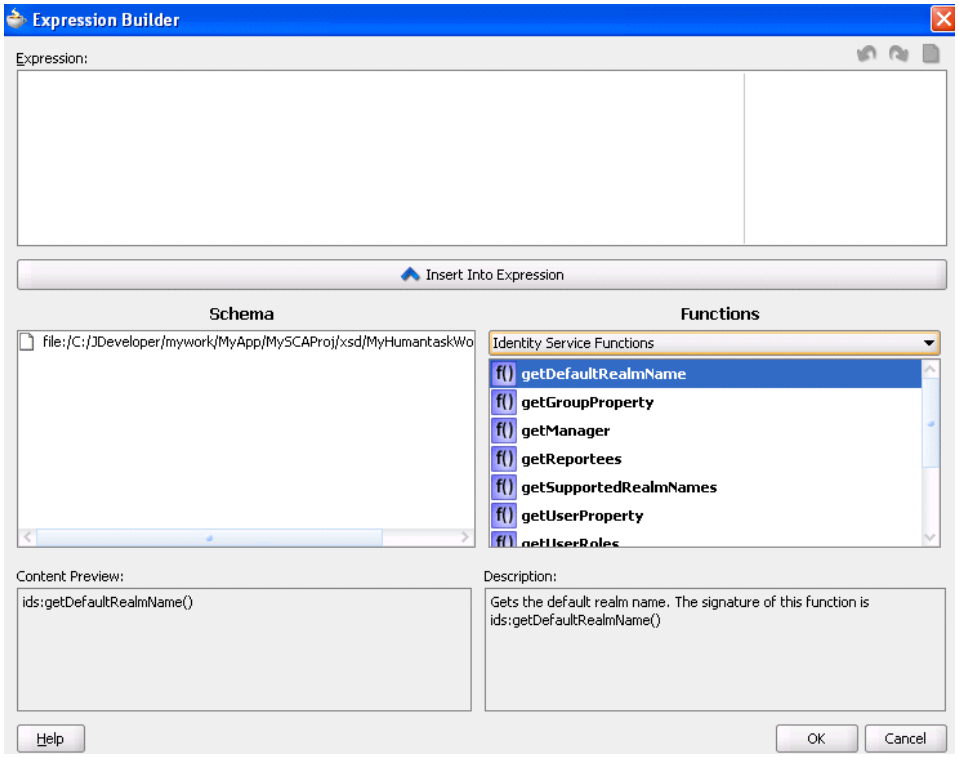
Figure 29-11 Specify a Task Owner Dynamically



2. In the second list to the right of the **Owner** field in the **General** section, select **XPath**.
3. Click the icon to launch the Expression Builder.

   This displays the Expression Builder dialog box shown in Figure 29-12:

Figure 29-12 Expression Builder



4. Browse the available variable schemas and functions to create a task owner.
5. Click **OK** to return to the Human Task Editor.

   Your selection displays in the **Owner** field.

   For more information, see the following:

   - Click **Help** for instructions on using the Expression Builder dialog box and XPath Building Assistant
   - See the Appendix on XPath Extension Functions in *Developing SOA Applications with Oracle SOA Suite* for information about workflow service dynamic assignment functions, identity service functions, and instructions on using the XPath Building Assistant

How To Specify an Application Context

You can specify the name of the application that contains the application roles used in the task. This indicates the context in which the application role operates. If you do not explicitly create a task, but end up having one, you can set up the context.

Note:

An application context is required to be set in the task definition in order to be able to reassign the task to an application role in the Oracle Process Workspace and Oracle BPM Worklist applications.

1. In the **Application Context** field of the **General** section, enter the name.

Specifying the Task Payload Data Structure

This section enables you to specify the structure (message elements) of the task payload (the data in the task) defined in the XSD file.

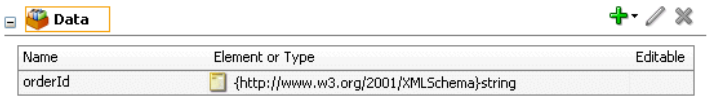
Figure 29-13 shows the **Data** section of the Human Task Editor.

You create parameters to represent the elements in the XSD file. This makes the payload data available to the workflow task. For example:

- You create a parameter for an order ID element for placing an order from a store front application.
- You create parameters for the location, type, problem description, severity, status, and resolution elements for creating a help desk request.

Task payload data consists of one or more elements or types. Based on your selections, an XML schema definition is created for the task payload.

Figure 29-13 Human Task Editor — Parameters Section



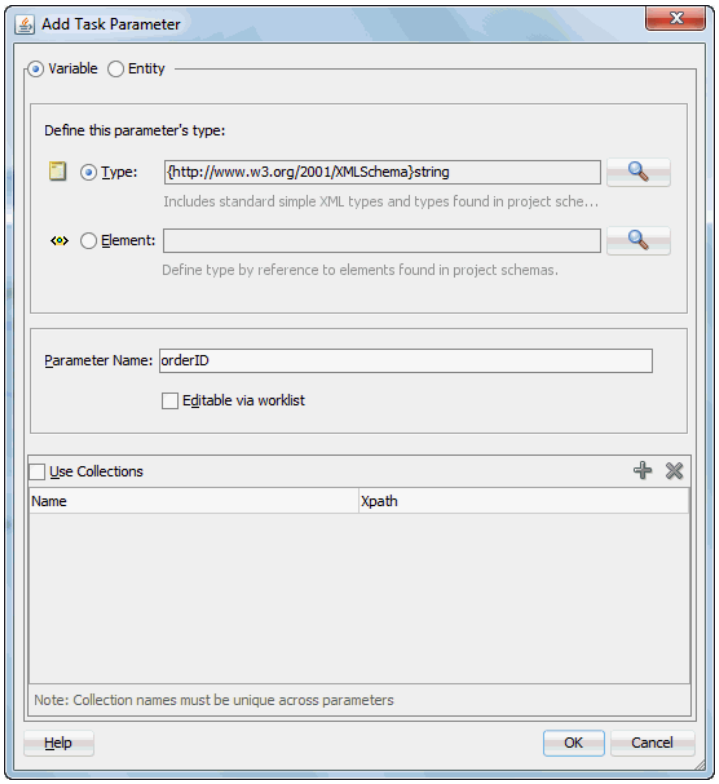
How to Specify the Task Payload Data Structure

To specify the task payload data structure:

1. Click the **Data** tab.
2. Click the **Add** icon and select a payload type:
   - String
   - Integer
   - Boolean
   - Other

   The Add Task Parameter dialog box is displayed, as shown in Figure 29-14.

Figure 29-14 Add Task Parameter Dialog



3. Enter the details described in Table 29-5:

Table 29-5 Add Task Parameter Dialog Fields and Values

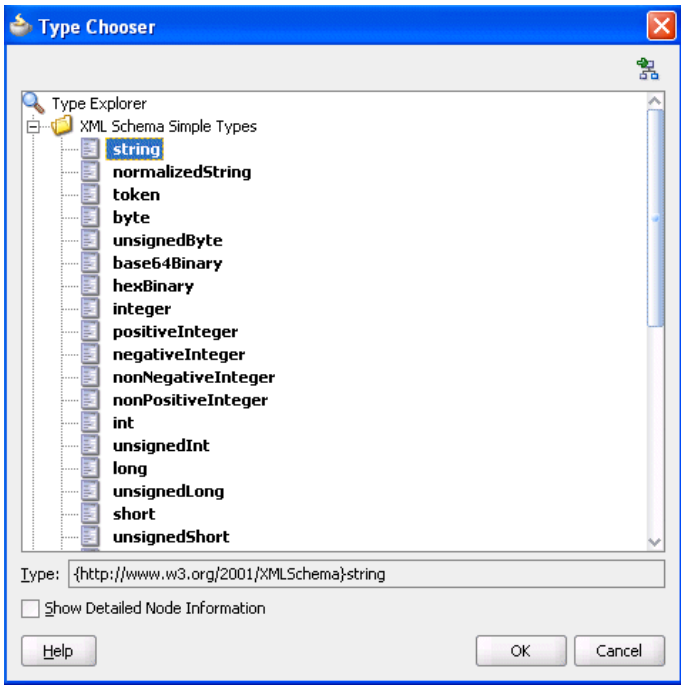
Field	Description
Parameter Type	Select **Type** or **Element** and click the **Search** icon to display the Type Chooser dialog box for selecting the task parameter.
Parameter Name	Accept the default name or enter a custom name. This field only displays if **Type** is the selected parameter type.
Editable via worklist	Select this check box to enable users to edit this part of the task payload in Oracle BPM Worklist. For example, for a loan approval task, the APR attribute may need to be updated by the user reviewing the task, but the SSN field may not be editable. You can also specify access rules that determine the parts of a task that participants can view and update. For more information, see How to Specify Access Policies on Task Content.
Use Collections	If a task uses collections, then define this parameter to use collections. Click the **Add** button to provide the collection name and the Xpath expression for the collection type. Use Expression Builder to look up the collection type from the schema.

Note:

You can only define payload mapped attributes (previously known as flex field mappings) in Oracle BPM Worklist for payload parameters that are simple XML types (string, integer, and so on) or complex types (for example, a purchase order, and so on). If you must search tasks using keywords or define views or delegation rules based on task content, then you must use payload parameters based on simple XML types. These simple types can be mapped to flex columns in Oracle BPM Worklist.

4. Select the type, as shown in Figure 29-15.

Figure 29-15 Parameter Type



5. Click **OK** to return to the Human Task Editor.

   Your selection displays in the **Data** section.

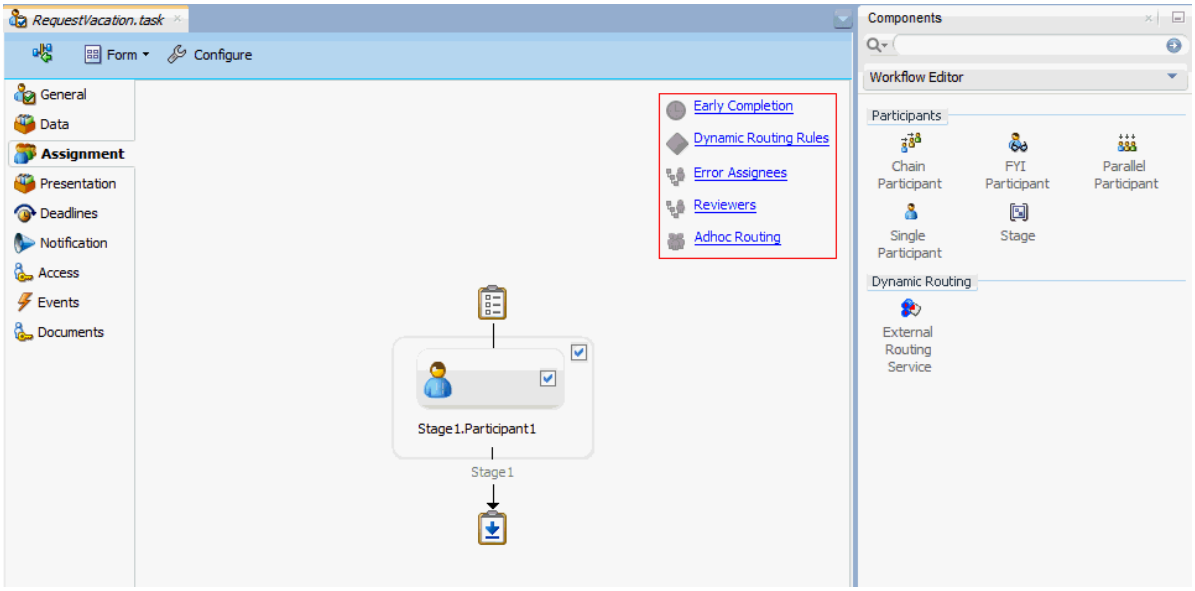
6. To edit your selection, select it and click the **Edit** icon in the upper right part of the **Data** section.

Assigning Task Participants

This section enables you to select a participant type that meets your business requirements. While configuring the participant type, you build lists of users, groups, and application roles to act upon tasks.

Figure 29-16 shows the **Assignment** section of the Human Task Editor.

Figure 29-16 Human Task Editor — Assignment Section



You can easily mix and match participant types to create simple or complex workflow routing policies. You can also extend the functionality of a previously configured human task to model more complex workflows.

A participant type is grouped in a block under a stage (for example, named **Stage1** in Figure 29-16). A stage is a way of organizing the approval process for blocks of participant types. You can have one or more stages in sequence or in parallel. Within each stage, you can have one or more participant type blocks in sequence or in parallel. The up and down keys enable you to rearrange the order of your participant type blocks.

For example:

- You can create all participant type blocks in a single stage (for example, a purchase order request in which the entire contents of the order are approved or rejected as a whole).
- You can create more complex approval tasks that may include one or more stages. For example, you can place one group of participant type blocks in one stage and another block in a second stage. The list of approvers in the first stage handles line entry approvals and the list of approvers in the second stage handles header entry approvals.

Each of the participant types has an associated editor that you use for configuration tasks. The sequence in which the assignees are added indicates the execution sequence.

To specify a different stage name or have a business requirement that requires you to create additional stages, perform the following steps. Creating additional stages is an advanced requirement that may not be necessary for your environment.

This section contains these topics:

- How to Specify a Stage Name and Add Parallel and Sequential Blocks
- How to Assign Task Participants
- How to Configure the Single Participant Type

- How to Configure the Parallel Participant Type
- How to Configure the Serial Participant Type
- How to Configure the FYI Participant Type

For more information about participant types, see Task Assignment and Routing.

How to Specify a Stage Name and Add Parallel and Sequential Blocks

To specify a stage name and add parallel and sequential blocks:

The stage is named **Stage1** by default, however you can change the name.

1. Double-click the name.

   The Edit dialog box displays.

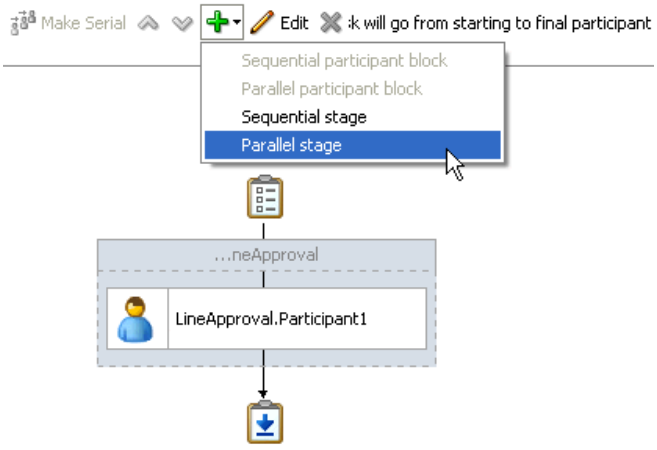
   Stage: The name of the stage. Non-Repeating: Do not stage in parallel for each item in the collection. Repeated: Stage in parallel for each item in a collection. Choose one collection from the drop-down list to specify which collection type to use for the repeated stages.

2. In the Edit dialog box, Enter a stage name, selected Non-Repeating or Repeated as applicable, and click **OK**.
3. Select the stage and its participant type block, as shown in Figure 29-17.
4. Drag and drop the Participant of type Stage to the right of the current stage.

   You can select the participant of type Stage from the workflow editor on the component pallet.

5. Click the **Add** icon.

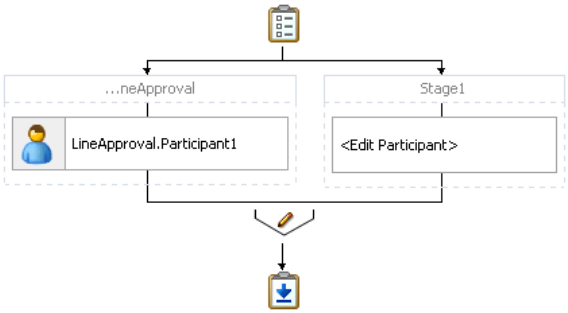
Figure 29-17 Add a Second Stage



6. Select an option from the list (for example, **Parallel stage**).

   A second stage is added in parallel to the first stage, as shown in Figure 29-18.

Figure 29-18 Parallel Stage



7. Drag and drop the Stage participant from the Components window below Stage 1.

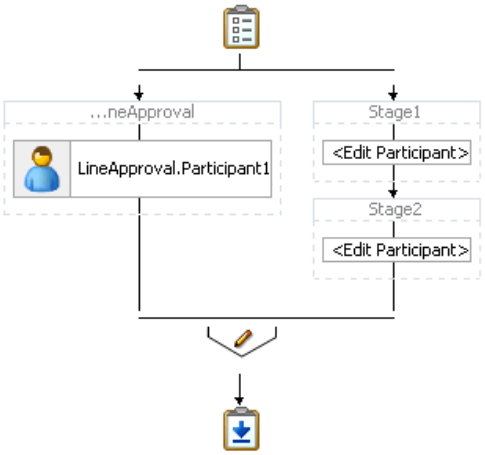
   When you bring the new stage below the current stage, four green dots display around the current stage. Choose the green dot that is below the current stage.

   If you do not select the second stage (for this example, named **Stage1** in Figure 29-19) and instead select only the participant type block (for example, named **Edit Participant** in Figure 29-19), all options under the **Add** icon are disabled.

8. Select **Sequential stage**.

   A sequential stage is added below the selected block.

Figure 29-19 Sequential Stage



You create participant types within these blocks.

How to Assign Task Participants

To assign task participants:

1. In the **Assignment** section, perform one of the following tasks:

   a. Drag and drop Participants from the Components window onto Stage. The first time you create a task participant, the box is labeled **<Edit Participant>**.

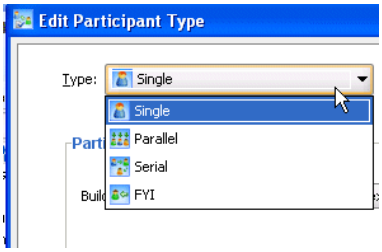
   or

   b. Double-click the participant box.

The Edit Participant Type dialog box appears. This dialog box enables you to select a specific participant type.

2. From the **Type** list, select a participant type shown in Figure 29-20.

Figure 29-20 Type List



3. See the section shown in Table 29-6 based on your selection.

Table 29-6 Participant Types

Participant Type	For a Description of this Participant Type, See...	For Instructions on Configuring this Participant Type, See...
• Single • Parallel • Serial • FYI	Participant Type	How to Configure the Single Participant Type How to Configure the Parallel Participant Type How to Configure the Serial Participant Type How to Configure the FYI Participant Type

Disable Task Participants and Task Stages

You can disable the participants or the stages that you added to the task. When you need to make temporary changes to the tasks, use this disable feature instead of deleting and adding the entire stage or participant again. To disable the task participants or the task stages, deselect the check boxes next to participants or the stage.

How to Configure the Single Participant Type

Figure 29-21 shows the Edit Participant Type dialog box for the single participant type. Figure 29-22 shows the expanded **Advanced** section.

Figure 29-21 Edit Participant Type — Single Type

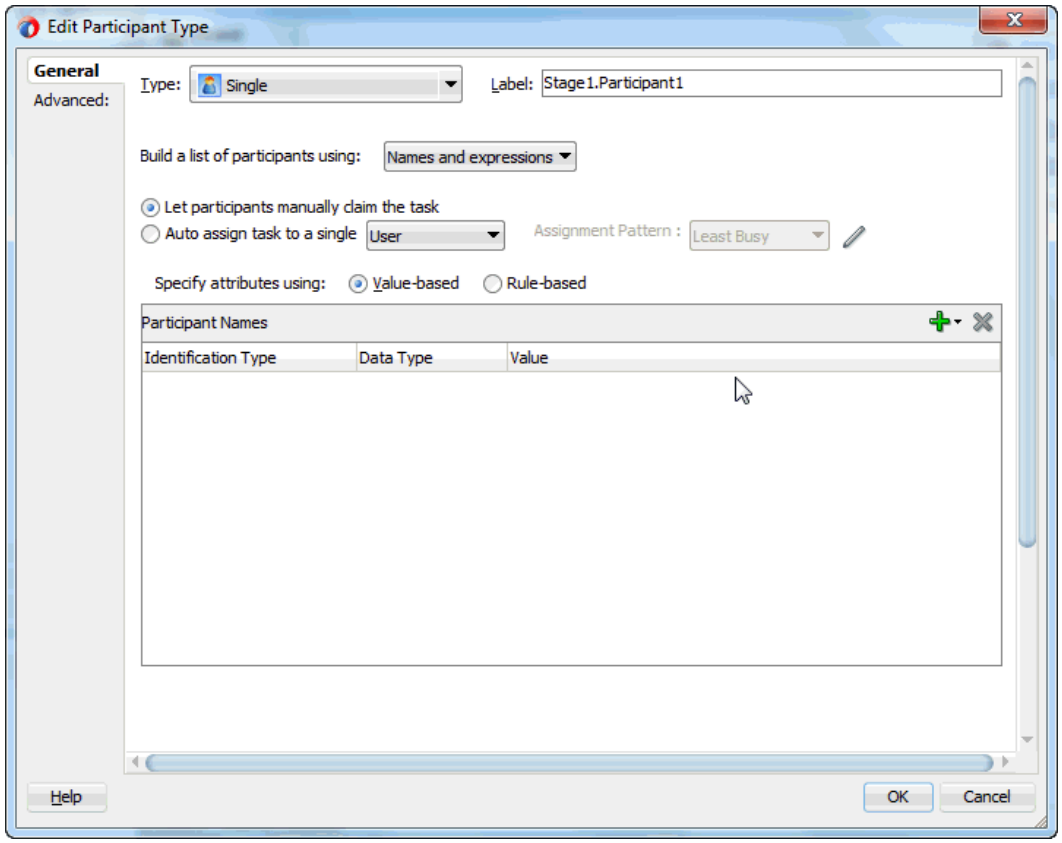
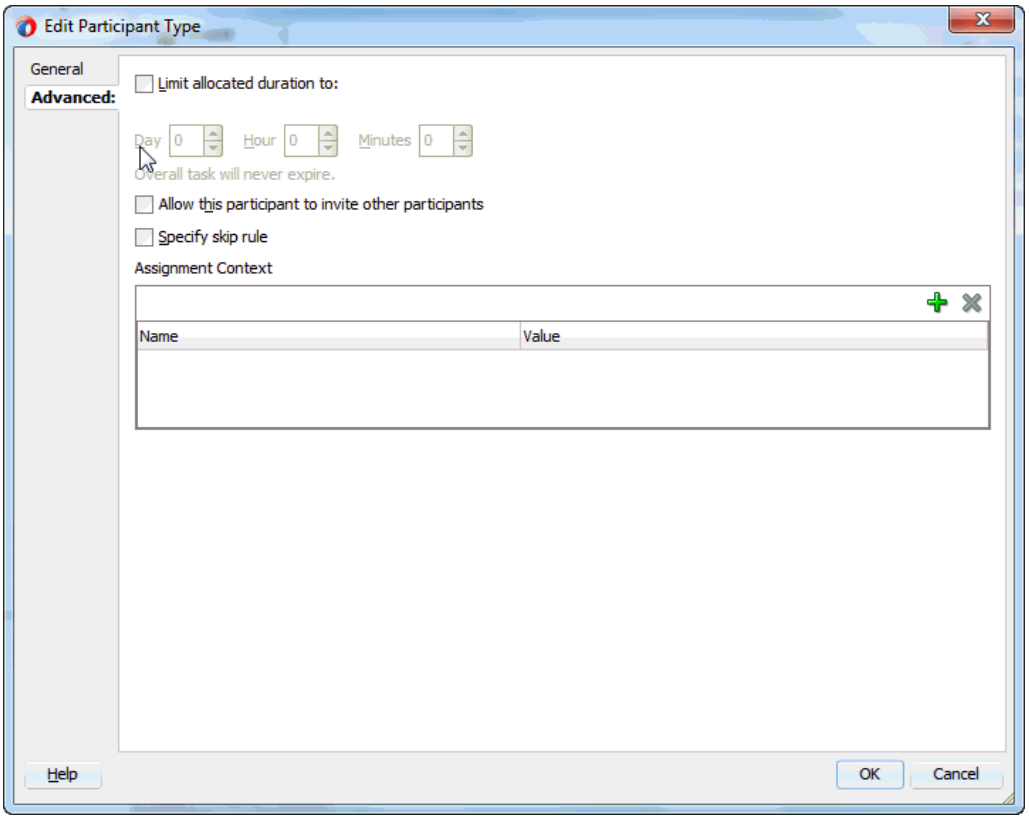


Figure 29-22 Edit Participant Type — Advanced Tab



To be dynamically assigned to a task, a single participant can be selected from a group, an application role, or a participant list.

To configure the single participant type:

1. In the **Label** field, enter a recognizable label for this participant. This label must be unique among all the participants in the task definition (for example, `Approval Manager`, `Primary Reviewers`, and so on).

   Instructions for configuring the following subsections of the Edit Participant Type dialog box for the single participant type are listed in Table 29-7:

Table 29-7 Edit Participant Type — Single Type

For This Subsection...	See...
Participant List	Creating a Single Task Participant List
Limit allocated duration to (under the **Advanced** section)	Specifying a Time Limit for Acting on a Task
Allow this participant to invite other participants (under the **Advanced** section)	Inviting Additional Participants to a Task
Specify skip rule (under the **Advanced** section)	Bypassing a Task Participant

Table 29-7 (Cont.) Edit Participant Type — Single Type

For This Subsection...	See...
Assignment Control (under the **Advanced** section)	If this participant is associated with a particular assignment context, then add that name here. Use the Add button to add new entry. Use the drop-down list to select the assignment context Name and provide a value for this assignment context.
Let participants manually claim task (under the **General** section)	Creating Participant Lists Consisting of Value-Based Names and Expressions
Auto assign task to a single user/group/ application role (under the **General** section)	Creating Participant Lists Consisting of Value-Based Names and Expressions

Creating a Single Task Participant List

Users assigned to a participant list can act upon tasks. In a single-task participant list, only one user is required to act on the task. You can specify either a single user or a list of users, groups, or application roles for this pattern. If a list is specified, then all users on the list are assigned the task. You can specify either that one of them must manually claim and act upon the task, or that one user from the list is automatically selected by an assignment pattern. When one user acts on the task, the task is withdrawn from the task list of other assignees.

You can create several types of lists for the single user participant, and for the parallel, serial, and FYI user participants, for example:

- Value-based name and expression lists

  These lists enable you to statically or dynamically select users, groups, or application roles as task assignees.

- Value-based management chain lists

  Management chains are typically used for serial approvals through multiple users in a management chain hierarchy. Therefore, this list is most likely useful with the serial participant type. This is typically the case if you want all users in the hierarchy to act upon the task. Management chains can also be used with the single participant type. In this case, however, all users in the hierarchy get the task assigned at the same time. As soon as one user acts on the task, it is withdrawn from the other users.

  For example, a purchase order is assigned to a manager. If the manager approves the order, it is assigned to their manager. If that manager approves it, it is assigned to their manager, and so on until three managers approve the order. If any managers reject the request or the request expires, the order is rejected if you specify an abrupt termination condition. Otherwise, the task flow continues to be routed.

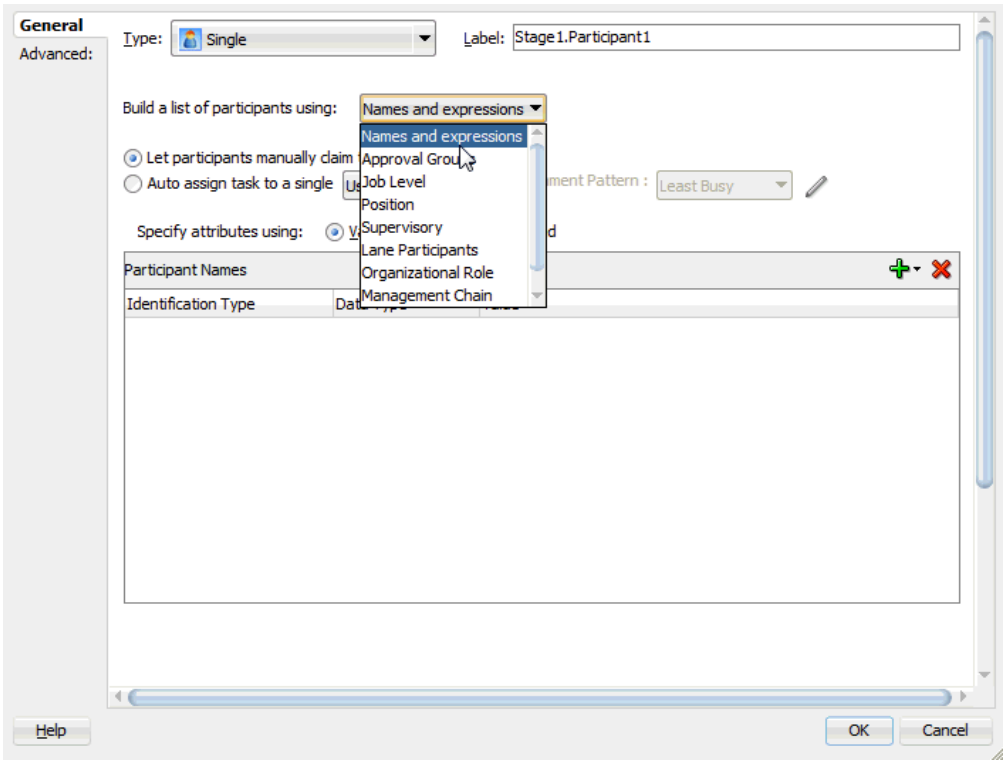
- Rule-based names and expression lists and management chain lists

  Business rules enable you to create the list of task participants with complex expressions. For example, you create a business rule in which a purchase order request below $5000 is sent to a manager for approval. However, if the purchase order request exceeds $5000, the request is sent to the manager of the manager for

approval. Two key features of business rules are facts and action types, which are described in How to Specify Advanced Task Routing Using Business Rules.

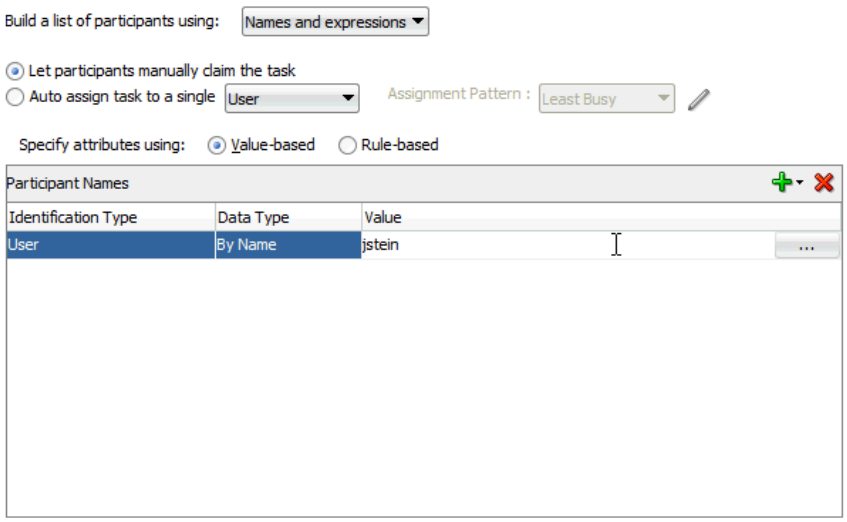
When you select a participant type, a dialog box enables you to choose an option for building your list of task participant assignees (users, groups, or application roles), as shown in Figure 29-23. The three selections described above are available: **Names and expressions**, **Management Chain**, and **Rule-based**.

Figure 29-23 Build a List of Participants



After selecting an option, you dynamically assign task participant assignees (users, groups, or application roles) and a data type, as shown in Figure 29-24.

Figure 29-24 Assignment of Task Assignees



This section describes how to create these lists of participants.

Creating Participant Lists Consisting of Value-Based Names and Expressions

Select a method for statically or dynamically assigning a user, group, or application role as a task participant. If the participant list contains a user, the selecting a group or an application role causes the dynamic assignment to fail.

For conceptual information, see the following:

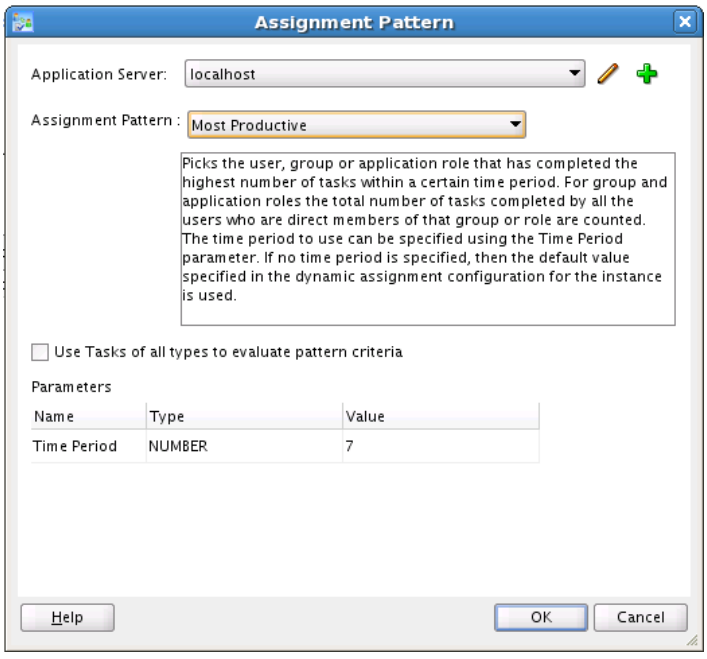
- Users, groups, or application roles, see Participant Assignment.
- Statically and dynamically assigning task participants, see Static_ Dynamic_ and Rule-Based Task Assignment.

To create participant lists consisting of value-based names and expressions:

1. From the **Build a list of participants using** list, select **Names and expressions**.
2. Do either of the following:
   - Select **Let participants manually claim the task**. If you select this option, then the task is assigned to all participants in the list. An individual user from the task assignees can then manually claim the task to work on it.
   - Select **Auto-assign to a single** list, select **User**, **Group**, or **Application Role**, then select an assignment pattern.

     To find out more about each assignment pattern, and to select and configure it, click **Assignment Pattern**. The Assignment Pattern dialog box appears. Figure 29-25 shows an example of an Assignment Pattern dialog box.

Figure 29-25 Selecting and Configuring an Assignment Pattern



When you specify an application server connection in the Application Server field, the assignment patterns are loaded into the Assignment Pattern list. When

you select one of the patterns from the Assignment Pattern list, a description of your selection appears in the text box.

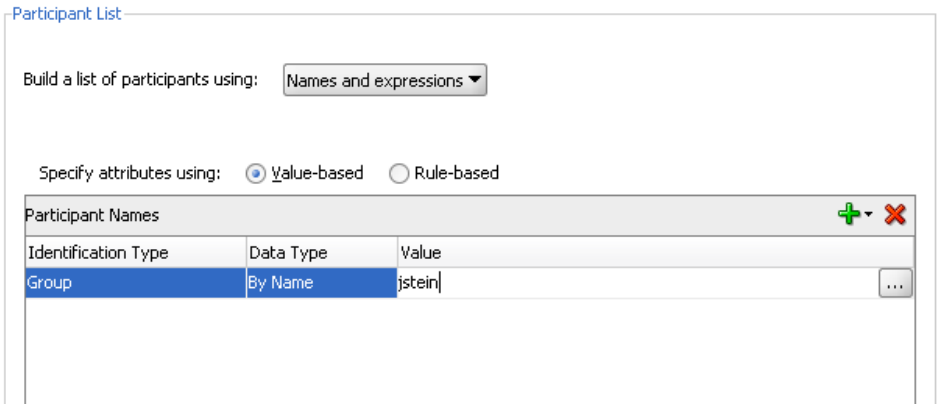
If you want the assignment pattern to consider all types of tasks, then select **Use tasks of all types to evaluate pattern criteria**. Otherwise, the pattern considers only this task type when determining the selected user. For example, to assign a vacation request task to the least busy user, and you select **Use tasks of all types to evaluate pattern criteria**, then all assigned tasks are taken into consideration when determining the least busy user. If you do *not* select **Use tasks of all types to evaluate pattern criteria**, then only assigned vacation request tasks are considered when determining the least busy user.

A particular pattern may enable you to specify input parameters that control how the pattern is evaluated. For example, as shown in Figure 29-25, the Most Productive pattern enables you to specify the Time Period (in days) over which the productivity is calculated. Input values can be static, or can be dynamically set by using an XPath expression. Not all patterns accept parameters.

3. From the **Specify attributes using** list, select **Value-based**.

   The dialog box refreshes to display the fields shown in Figure 29-26.

Figure 29-26 Value-Based Names and Expressions



4. Click the **Add** icon and select a user, group, or application role as a task participant.

   The **Identification Type** column of the **Participant Names** table displays your selection of user, group, or application role.

5. To change your selection in the **Identification Type** column, click it to invoke a drop-down list.

6. In the **Data Type** column, click your selection to invoke a drop-down list to assign a value:

   - **By Name:** If your identification type is a user or group, click the **Browse** icon (the dots) on the right to display a dialog box for selecting a user or group configured through the identity service. The identity service enables the lookup of user properties, roles, and group memberships. User information is obtained from an LDAP server such as Oracle Internet Directory. You can use wild cards (*) to search for IDs.

     If your selection is an application role, click the **Browse** icon to display the Select an Application Role dialog box for selecting an application role. To search

for application roles, you must first create a connection to the application server. When searching, you must specify the application name to find the name of the role. The task definition can refer to only one application name. You cannot use application roles from different applications as assignees or task owners.

- **By Expression**: For a user, group, or application role, click the **Browse** icon to dynamically select a task assignee in the Expression Builder dialog box. Use the `bpws:getVariableData(...)` expression or the `ids:getManager()` XPath function.

The **Value** column displays the value you specified.

7. To manually enter a value, click the field in the **Value** column and specify a value.

Creating Participant Lists Consisting of Value-Based Management Chains

Select a method for statically or dynamically assigning management chain parameters as task participants.

For conceptual information about the following:

- Users, groups, or application roles, see Participant Assignment.
- Statically and dynamically assigning task participants, see Static_ Dynamic_ and Rule-Based Task Assignment.
- Management chains, see Creating a Single Task Participant List.

To create participant lists based on value-based management chains:

1. From the **Build a list of participants using** list, select **Management Chain**.
2. Do either of the following:
   - Select **Let participants manually claim the task**. If you select this option, then the task is assigned to all participants in the list. An individual user from the task assignees can then manually claim the task to work on it.
   - Select **Auto-assign to a single** list, select **User**, then select an assignment pattern.

     To find out more about each assignment pattern, and to select and configure it, click **Assignment Pattern**. The Assignment Pattern dialog box appears. Figure 29-25 shows an example of an Assignment Pattern dialog box.

     When you specify an application server connection in the Application Server field, the assignment patterns are loaded into the Assignment Pattern list. When you select one of the patterns from the Assignment Pattern list, a description of your selection appears in the text box.

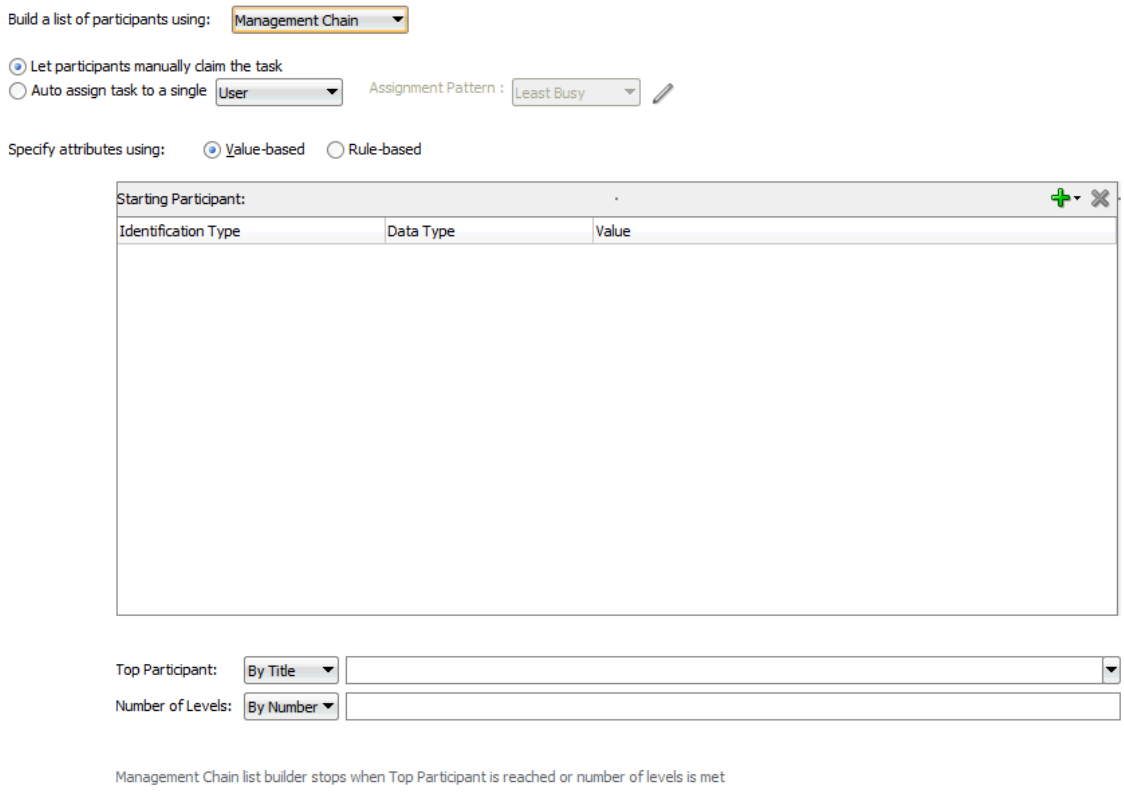
     If you want the assignment pattern to consider all types of tasks, then select **Use tasks of all types to evaluate pattern criteria**. Otherwise, the pattern considers only this task type when determining the selected user. For example, to assign a vacation request task to the least busy user, and you select **Use tasks of all types to evaluate pattern criteria**, then all assigned tasks are taken into consideration when determining the least busy user. If you do *not* select **Use tasks of all types to evaluate pattern criteria**, then only assigned vacation request tasks are considered when determining the least busy user.

A particular pattern may enable you to specify input parameters that control how the pattern is evaluated. For example, as shown in Figure 29-25, the Most Productive pattern enables you to specify the Time Period (in days) over which the productivity is calculated. Input values can be static, or can be dynamically set by using an XPath expression. Not all patterns accept parameters.

3. From the **Specify attributes using** list, select **Value-based**.

   The dialog box refreshes to display the fields shown in Figure 29-27.

Figure 29-27 Value-Based Management Chains



4. See Step 4 through Step 7 of Creating a Single Task Participant List for instructions on assigning a user, group, or application role to a list in the **Starting Participant** table.

5. In the **Top Participant** list, select a method for assigning the number of task participant levels:

   - **By Title**: Select the title of the last (highest) approver in the management chain.
   - **XPath**: Select to dynamically enter a top participant through the Expression Builder dialog box.

6. In the **Number of Levels** list, select a method for assigning a top participant:

   - **By Number**: Enter a value for the number of levels in the management chain to include in this task. For example, if you enter `2` and the task is initially assigned to user `jcooper`, both the user `jstein` (manager of `jcooper`) and the user `wfaulk` (manager of `jstein`) are included in the list (apart from `jcooper`, the initial assignee).

- **XPath**: Select to dynamically enter a value through the Expression Builder dialog box.

Creating Participant Lists Consisting of Rulesets

A ruleset provides a unit of execution for rules and for decision tables. In addition, rulesets provide a unit of sharing for rules; rules belong to a ruleset. Multiple rulesets can be executed in order. This is called rule flow. The ruleset stack determines the order. The order can be manipulated by rule actions that push and pop rulesets on the stack. In rulesets, the priority of rules applies to specify the order of firing of rules in the ruleset. Rulesets also provide an effective date specification that identifies that the ruleset is always active, or that the ruleset is restricted based on a time and date range, or a starting or ending time and date.

The method by which you create a ruleset is based on how you access it. This is described in the following section.

Note:

You cannot update facts after the rule dictionary is created.

To specify participant lists based on rulesets:

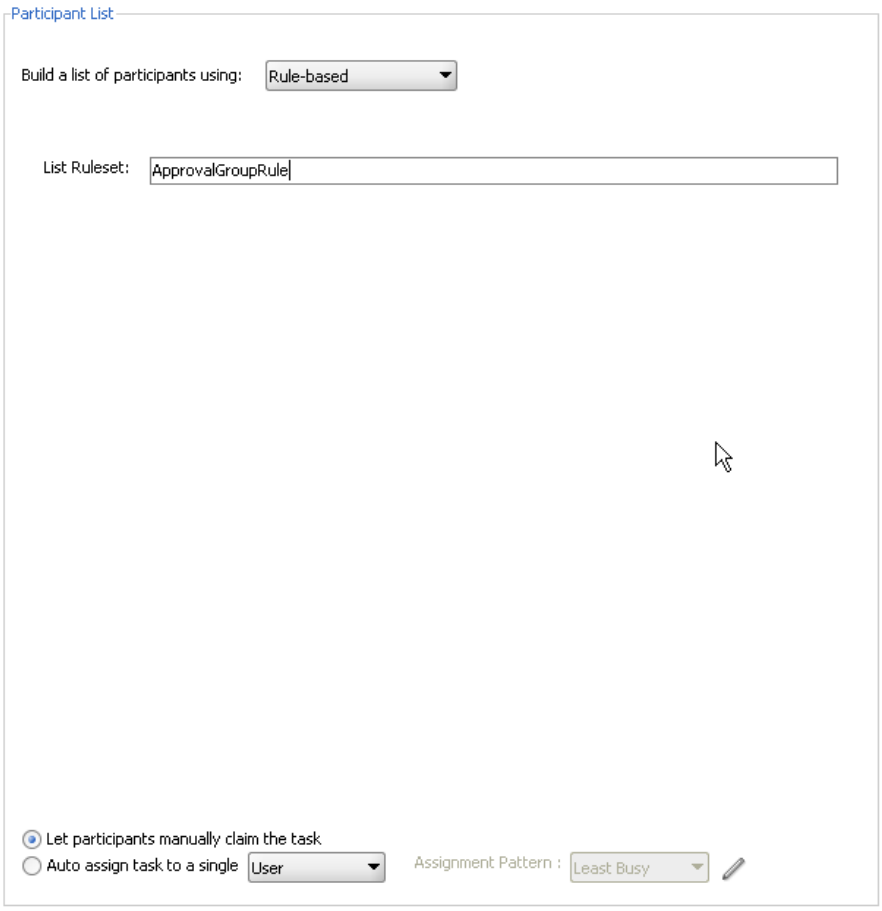
Business rules can define the participant list. There are two options for using business rules:

- Rules define parameters of a specific list builder (such as **Names and Expressions** or **Management Chain**). In this case, the task routing pattern is modeled to use a specific list builder. In the list builder, the parameters are listed as coming from rules. Rules return the list builder of the same type as the one modeled in Oracle JDeveloper.

  1. From the **Build a list of participants using** list, select **Names and expressions** or **Management Chain**.
  2. From the **Specify attributes using** list, select **Rule-based**.
  3. In the **List Ruleset** field, enter a ruleset name.

     Figure 29-28 provides details.

Figure 29-28 Rulesets



4. Do either of the following:

   – Select **Let participants manually claim the task**. If you select this option, then the task is assigned to all participants in the list. An individual user from the task assignees can then manually claim the task to work on it.

   – Select **Auto-assign to a single** list, select **User**, **Group**, or **Application Role**, then select an assignment pattern.

     To find out more about each assignment pattern, and to select and configure it, click **Assignment Pattern**. The Assignment Pattern dialog box appears. Figure 29-25 shows an example of an Assignment Pattern dialog box.

     When you specify an application server connection in the Application Server field, the assignment patterns are loaded into the Assignment Pattern list. When you select one of the patterns from the Assignment Pattern list, a description of your selection appears in the text box.

     If you want the assignment pattern to consider all types of tasks, then select **Use tasks of all types to evaluate pattern criteria**. Otherwise, the pattern considers only this task type when determining the selected user. For example, to assign a vacation request task to the least busy user, and you select **Use tasks of all types to evaluate pattern criteria**, then all assigned tasks are taken into consideration when determining the least busy user. If you do *not* select **Use tasks of all types to evaluate pattern criteria**, then

only assigned vacation request tasks are considered when determining the least busy user.

A particular pattern may enable you to specify input parameters that control how the pattern is evaluated. For example, as shown in Figure 29-25, the Most Productive pattern enables you to specify the Time Period (in days) over which the productivity is calculated. Input values can be static, or can be dynamically set by using an XPath expression. Not all patterns accept parameters.

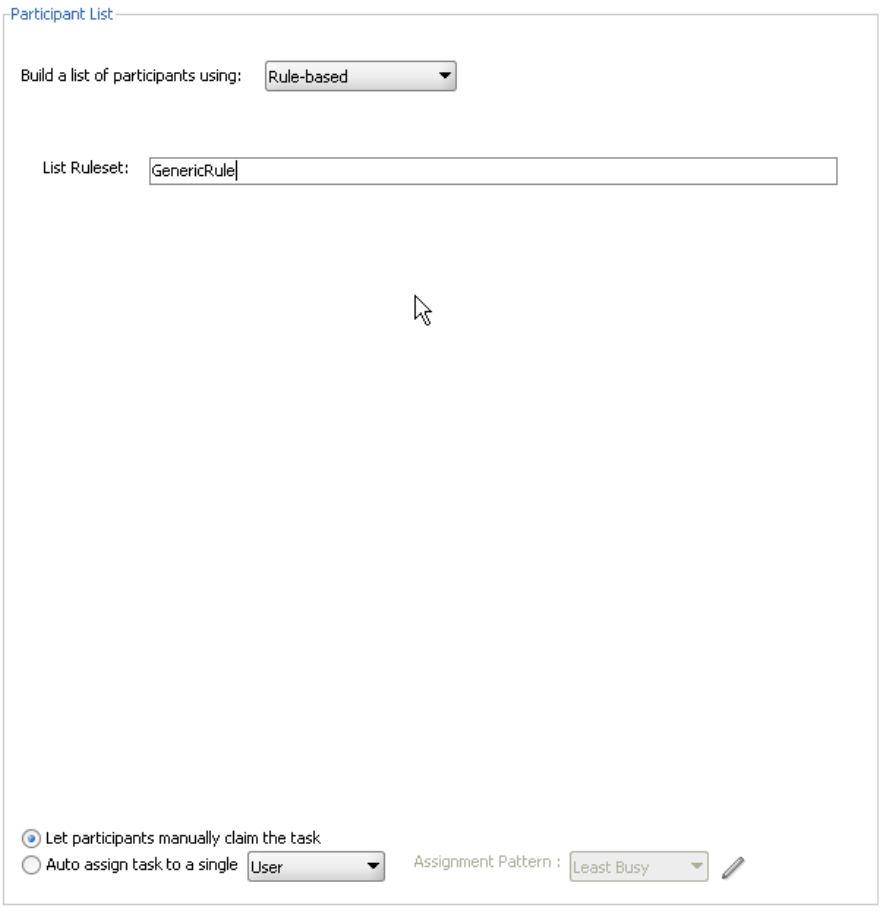
5. Click **OK**.

- Rules define the list builder and the list builder parameters. In this case, the list itself is built using rules. The rules define the list builder and the parameters.

1. From the **Build a list of participants using** list, select **Rule-based**.
2. In the **List Ruleset** field, enter a ruleset name.

   Figure 29-29 provides details.

Figure 29-29 Rulesets



3. Do either of the following:

   – Select **Let participants manually claim the task**. If you select this option, then the task is assigned to all participants in the list. An individual user from the task assignees can then manually claim the task to work on it.

– Select **Auto-assign to a single** list, select **User**, **Group**, or **Application Role**, then select an assignment pattern.

To find out more about each assignment pattern, and to select and configure it, click **Assignment Pattern**. The Assignment Pattern dialog box appears. Figure 29-25 shows an example of an Assignment Pattern dialog box.

When you specify an application server connection in the Application Server field, the assignment patterns are loaded into the Assignment Pattern list. When you select one of the patterns from the Assignment Pattern list, a description of your selection appears in the text box.

If you want the assignment pattern to consider all types of tasks, then select **Use tasks of all types to evaluate pattern criteria**. Otherwise, the pattern considers only this task type when determining the selected user. For example, to assign a vacation request task to the least busy user, and you select **Use tasks of all types to evaluate pattern criteria**, then all assigned tasks are taken into consideration when determining the least busy user. If you do *not* select **Use tasks of all types to evaluate pattern criteria**, then only assigned vacation request tasks are considered when determining the least busy user.

4. Click **OK**.

Both options create a rule dictionary, if one is not already created, and pre-seed several rule functions and facts for easy specifications of the participant list. In the rule dictionary, the following rule functions are seeded to create participant lists:

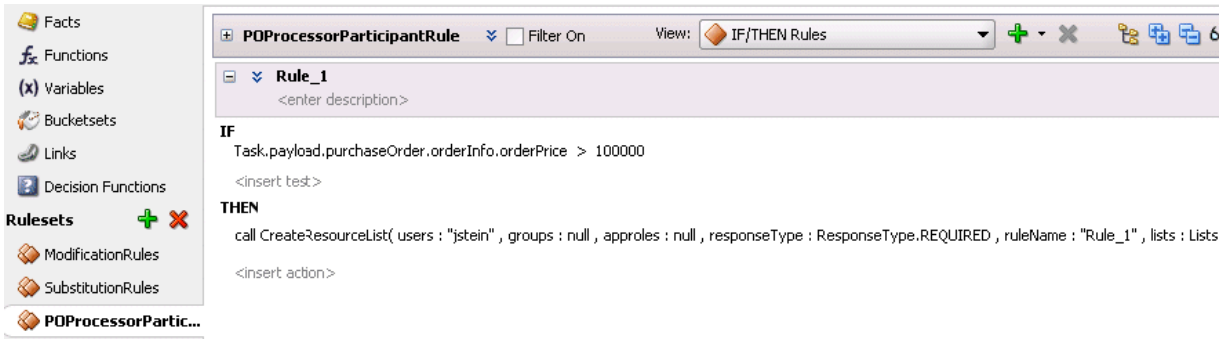
- `CreateResourceList`
- `CreateManagementChainList`

The `Task` fact is asserted by the task service for basing rule conditions.

After the rule dictionary is created, the Oracle Business Rules Designer is displayed.

1. Model your rule conditions. In the action part, call one of the above functions to complete building your lists. Figure 29-30 provides details.

Figure 29-30 Business Rules



The parameters for the rule functions are similar to the ones in Oracle JDeveloper modeling. In addition to the configurations in Oracle JDeveloper, some additional options are available in the Oracle Business Rules Designer for the following attributes:

- **responseType**: If the response type is **REQUIRED**, the assignee must act on the task. Otherwise, the assignment is converted to an FYI assignment.

- **ruleName**: The rule name can create reasons for assignments.
- **lists**: This object is a holder for the lists that are built. Clicking this option shows a pre-asserted fact **Lists** object to use as the parameter.

An example of rules specifying management chain-based participants is shown in Figure 29-31.

Figure 29-31 Business Rules

If multiple rules are fired, the list builder created by the rule with the highest priority is selected.

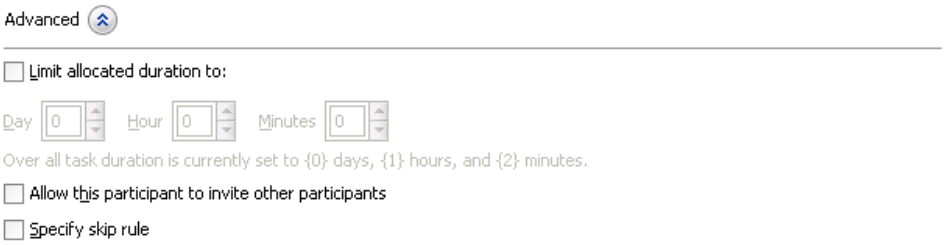
Specifying a Time Limit for Acting on a Task

You can specify the amount of time a user, group, or application role receives to act on a task. If the user, group, or role does not act in the time specified, the global escalation and renewal policies that you set in the **Deadlines** section (known as the routing slip level) of the Human Task Editor are applied. For example, if the global policy is set to escalate the task and this participant does not act in the duration provided, the task is escalated to the manager or another user, as appropriate.

To specify a time limit for acting on a task:

1. Expand the **Advanced** section of the Edit Participant Type dialog box for the single type, as shown in Figure 29-32.

   Figure 29-32 Advanced Section of Edit Participant Type — Single Type

2. Select **Limit allocated duration to**.
3. Specify the amount of time.

   For more information about setting the global escalation and renewal policies in the **Deadlines** section of the Human Task Editor, see Escalating_ Renewing_ or Ending the Task.

Inviting Additional Participants to a Task

You can allow a task assignee to invite other participants into the workflow before routing it to the next assignee in this workflow. For example, assume the approval workflow goes from James Cooper to John Steinbeck. If this option is checked, James Cooper can decide to first route it to Irving Stone before it goes to John Steinbeck.

This is also known as ad hoc routing. If this option is selected, **Adhoc Route** is added to the **Actions** list in Oracle BPM Worklist at runtime.

Note:

Do not add adhoc assignees either above or below an FYI participant.

To invite additional participants to a task:

1. Expand the **Advanced** section of the Edit Participant Type dialog box for the single type, as shown in Figure 29-32.
2. Select **Allow this participant to invite other participants**.

Bypassing a Task Participant

You can bypass a task participant (user, group, or application role) if a specific condition is satisfied. For example, if a user submits a business trip expense report that is under a specific amount, no approval is required by their manager.

To bypass a task:

1. Expand the **Advanced** section of the Edit Participant Type dialog box for the single type, as shown in Figure 29-32.
2. Select **Specify skip rule**.

   This action displays an icon for accessing the Expression Builder dialog box for building a condition.

   The expression to bypass a task participant must evaluate to a boolean value. For example, **/task:task/task:payload/order:orderAmount < 1000** is a valid XPath expression for skipping a participant.

   For more information about creating dynamic rule conditions, see How to Specify Advanced Task Routing Using Business Rules.

How to Configure the Parallel Participant Type

Figure 29-33 and Figure 29-34 display the upper and lower sections of the Parallel dialog box.

This participant type is used when multiple users, working in parallel, must act simultaneously, such as in a hiring situation when multiple users vote to hire or reject an applicant. You specify the voting percentage that is needed for the outcome to take effect, such as a majority vote or a unanimous vote.

For example, a business process collects the feedback from all interviewers in the hiring process, consolidates it, and assigns a hire or reject request to each of the interviewers. At the end, the candidate is hired if the majority of interviewers vote for hiring instead of rejecting.

Figure 29-33 Edit Participant Type — Parallel Type (Upper Section of Dialog)

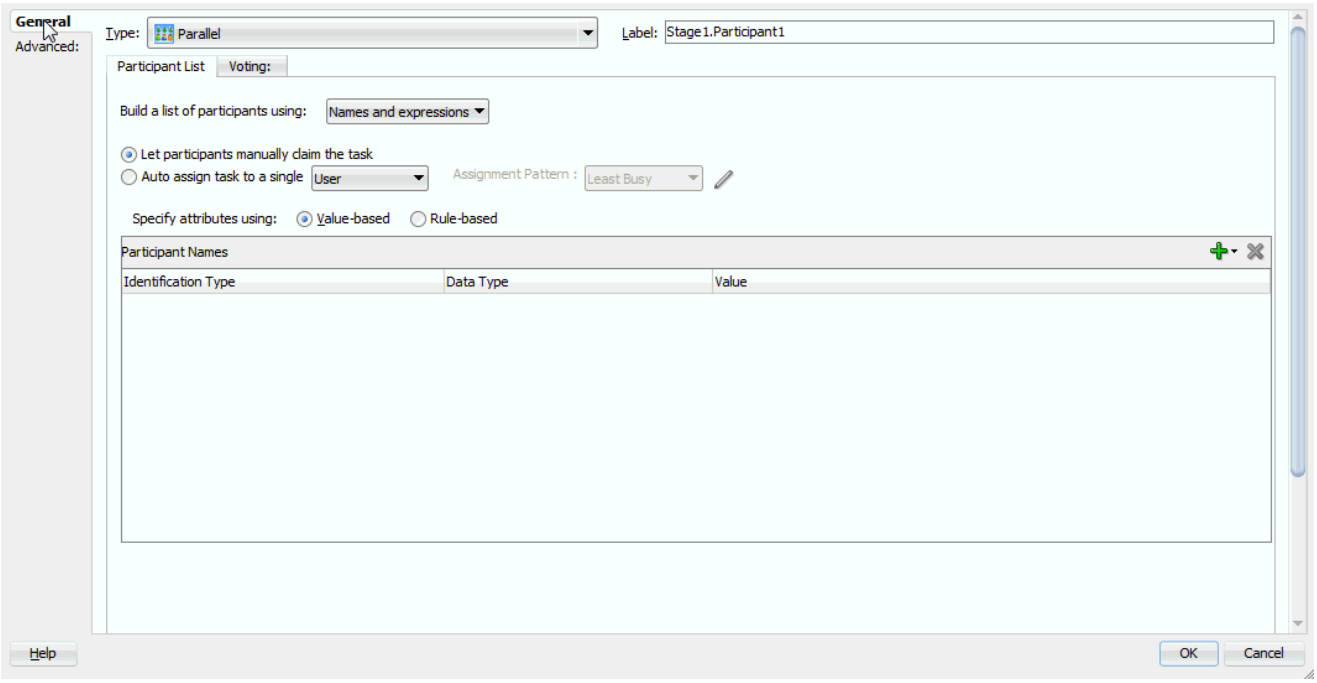


Figure 29-34 Edit Participant Type — Parallel Type (Lower Section of Dialog)

To assign participants to the parallel participant type:

1. In the **Label** field, enter a recognizable label for this participant. This label must be unique among all the participants in the task definition (for example, `Approval Manager`, `Primary Reviewers`, and so on).

   Instructions for configuring the following subsections of the Edit Participant Type dialog box for the parallel participant type are listed in Table 29-8:

Table 29-8 Edit Participant Type — Parallel Type

For This Subsection...	See...
Vote Outcome	Specifying the Voting Outcome
Participant List	Creating a Parallel Task Participant List
Limit allocated duration to (under the **Advanced** section)	Specifying a Time Limit for Acting on a Task

Table 29-8 (Cont.) Edit Participant Type — Parallel Type

For This Subsection...	See...
Allow this participant to invite other participants (under the **Advanced** section)	Inviting Additional Participants to a Task
Specify skip rule (under the **Advanced** section)	Bypassing a Task Participant
Add Assignment Context (under the **Advanced** section)	If this participant is associated with a particular assignment context, then add that name here. Use the Add button to add a new entry. Use the drop-down list to select an assignment context Name and to provide a value for this assignment context.

Specifying the Voting Outcome

You can specify a voted-upon outcome that overrides the default outcome selected in the **Default Outcome** list. This outcome takes effect if the required percentage is reached. Outcomes are evaluated in the order listed in the table.

To specify group voting details:

1. Go to the **Vote Outcome** section of the Edit Participant Type dialog box for the parallel type.
2. From the list in the **Voted Outcomes** column, select an outcome for the task (for example, **Any, ACCEPT**, **REJECT**, or any other outcome specified in How to Specify a Task Outcome).

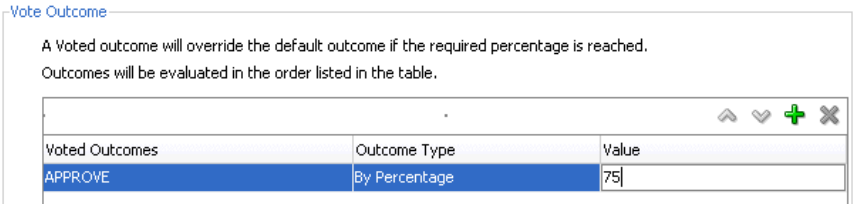
   The **Any** outcome enables you to determine the outcome dynamically at runtime. For example, if you select **Any** and set the outcome percentage to `60`, then at runtime, whichever outcome reaches 60% becomes the final voted outcome. If 60% of assignees vote to reject the outcome, then it is rejected.
3. From the list in the **Outcome Type** column, select a method for determining the outcome of the final task.

   - **By Expression**: Dynamically specify the details with an XPath expression.
   - **By Percentage**: Specify a percentage value that determines when the outcome of this task takes effect.
4. From the list in the **Value** column, specify a value based on your selection in Step 3.

   - If you selected **By Expression**, click the **Browse** icon to the right of the field to display the Expression Builder dialog box for creating an expression.
   - If you selected **By Percentage**, enter a percentage value required for the outcome of this task to take effect (for example, a majority vote (`51`) or a unanimous vote (`100`)). For example, assume there are two possible outcomes (**ACCEPT** and **REJECT**) and five subtasks. If two subtasks are accepted and three are rejected, and the required acceptance percentage is 50%, the outcome of the task is rejected. Figure 29-35 provides details.

     This functionality is nondeterministic. For example, selecting a percentage of **30%** when there are two subtasks does not make sense.

Figure 29-35 Vote Outcomes Section



5. Click the **Add** icon to specify additional outcomes.
6. In the **Default Outcome** list, select the default outcome or enter an XPath expression for this task to take effect if the consensus percentage value is not satisfied. This happens if there is a tie or if all participants do not respond before the task expires. Seeded and custom outcomes that you entered in the Outcomes dialog box in How to Specify a Task Outcome display in this list.
7. Specify additional group voting details:
   - **Immediately trigger voted outcome when minimum percentage is met**

     If selected, the outcome of the task can be computed early with the outcomes of the completed subtasks, enabling the pending subtasks to be withdrawn. For example, assume four users are assigned to act on a task, the default outcome is **APPROVE**, and the consensus percentage is set at **50**. If the first two users approve the task, the third and fourth users do not need to act on the task, since the consensus percentage value has been satisfied.

   - **Wait until all votes are in before triggering outcome**

     If selected, the workflow waits for all responses before an outcome is initiated.
8. To share comments and attachments with all group collaborators or workflow participants for a task, select **Share attachments and comments**. This information typically displays in the footer region of Oracle BPM Worklist.

Creating a Parallel Task Participant List

Users assigned to the list of participants can act upon tasks. You can create several types of lists:

- Value-based name and expression lists
- Value-based management chain lists
- Rule-based names and expression lists
- Rule-based management chain lists
- Rule-based links

For information about creating these lists of participants, see section Creating a Single Task Participant List.

Specifying a Time Limit for Acting on a Task

You can specify the amount of time a user, group, or application role receives to act on a task. If the user, group, or role does not act in the time specified, the global escalation and renewal policies that you set in the **Deadlines** section (known as the routing slip

level) of the Human Task Editor are applied. For example, if the global policy is set to escalate the task and this participant does not act in the duration provided, the task is escalated to the manager or another user, as appropriate.

To specify a time limit for acting on a task:

1. In the **Advanced** section of the Edit Participant Type dialog box for the parallel type, click the **Advanced** tab to expand the section shown in Figure 29-34.
2. Select **Limit allocated duration to**.
3. Specify the amount of time.

For more information about setting the global escalation and renewal policies in the **Deadlines** section of the Human Task Editor, see Escalating_ Renewing_ or Ending the Task.

Inviting Additional Participants to a Task

You can allow a task assignee to invite other participants into the workflow before routing it to the next assignee in this workflow. For example, assume the approval workflow goes from James Cooper to John Steinbeck. If this option is checked, James Cooper can decide to first route it to Irving Stone before it goes to John Steinbeck.

To invite additional participants to a task:

1. In the **Advanced** section of the Edit Participant Type dialog box for the parallel type, click the **Advanced** icon to expand the section (if not expanded).
2. Select **Allow this participant to invite other participants**.

Bypassing a Task Participant

You can bypass a task participant (user, group, or application role) if a specific condition is satisfied. For example, if a user submits a business trip expense report that is under a specific amount, no approval is required by their manager.

To bypass a task participant:

1. In the Edit Participant Type dialog box for the parallel type, select the **Specify skip rule** check box.

   This action displays an icon for accessing the Expression Builder dialog box for building a condition. The expression must evaluate to a boolean value.

   For information about a valid XPath expression for skipping a participant, see Bypassing a Task Participant.

How to Configure the Serial Participant Type

Figure 29-36 displays the Serial dialog box. Figure 29-37 shows the expanded **Advanced** section.

This participant type enables you to create a list of sequential participants for a workflow. For example, if you want a document to be reviewed by John, Mary, and Scott in sequence, use this participant type. For the serial participant type, they can be any list of users or groups.

Figure 29-36 Edit Participant Type — Serial Type

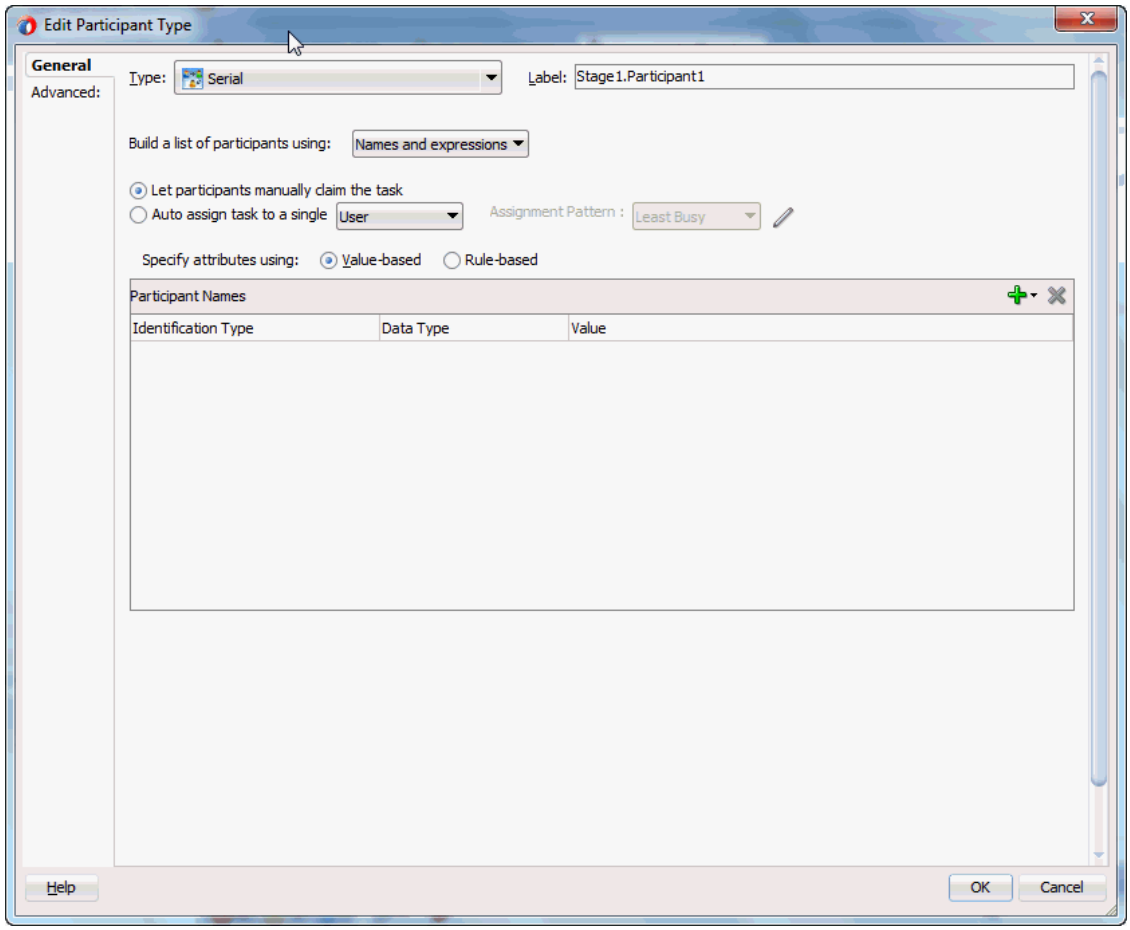
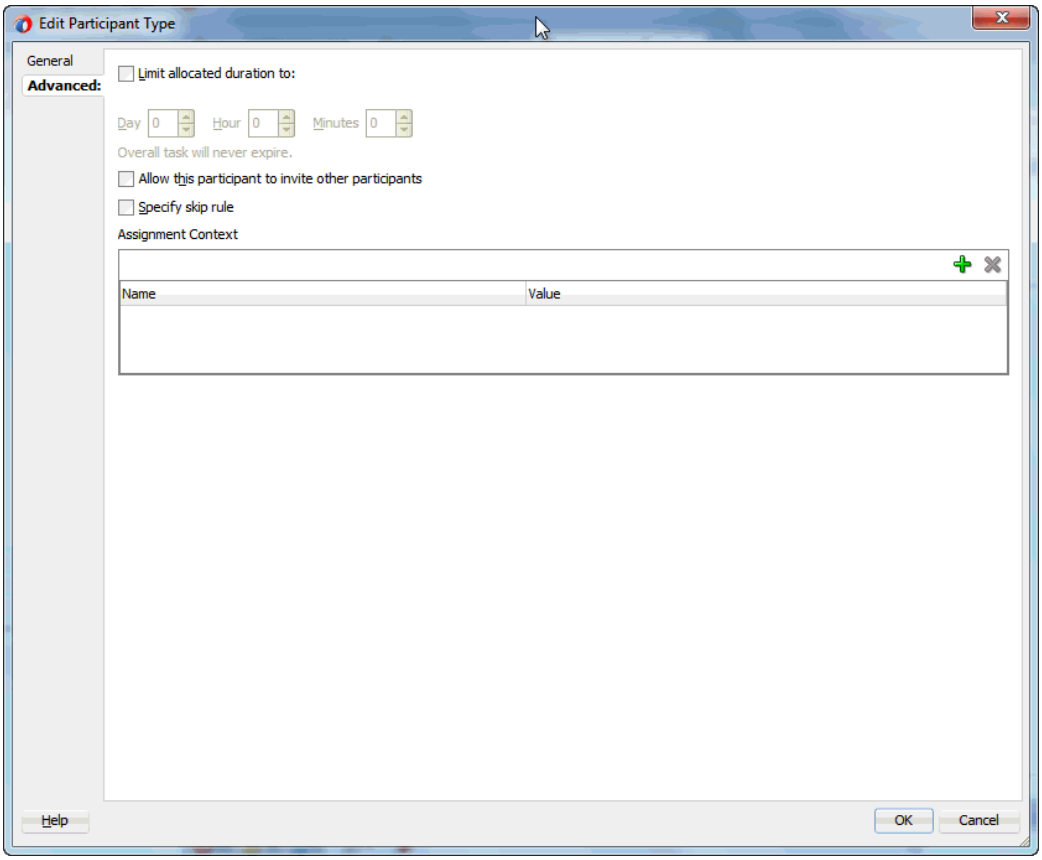


Figure 29-37 Edit Participant Type — Serial Type (Advanced Tab)



To configure the serial participant type:

1. In the **Label** field, enter a recognizable label for this participant. This label must be unique among all the participants in the task definition (for example, `Approval Manager`, `Primary Reviewers`, and so on).

   Instructions for configuring the following subsections of the Edit Participant Type dialog box for the serial participant type are listed in Table 29-9.

Table 29-9 Edit Participant Type — Serial Type

For This Subsection...	See...
Participant List	Creating a Serial Task Participant List
Limit allocated duration to (under the **Advanced** section)	Specifying a Time Limit for Acting on a Task Note that if you specify the task expiry time at the level of a serial participant, then, when that time expires, the task does not move to the next participant in the series. Rather, the entire task expires.
Allow this participant to invite other participants (under the **Advanced** section)	Inviting Additional Participants to a Task

Table 29-9 (Cont.) Edit Participant Type — Serial Type

For This Subsection...	See...
Specify skip rule (under the **Advanced** section)	Bypassing a Task Participant
Assignment Context (under the **Advanced** section)	If this participant is associated with a particular assignment context, then add that name here. Use the Add button to add a new entry. Use the drop-down list to select assignment context Name and to provide a value for this assignment context.

Creating a Serial Task Participant List

Users assigned to the list of participants can act upon tasks. You can create several types of lists:

- Value-based name and expression lists
- Value-based management chain lists
- Rule-based names and expression lists
- Rule-based management chain lists
- Rule-based lists

See section Creating a Single Task Participant List for instructions on creating these lists of participants.

Specifying a Time Limit for Acting on a Task

You can specify the amount of time a user, group, or application role receives to act on a task. If the user, group, or role does not act in the time specified, the global escalation and renewal policies that you set in the **Deadlines** section (known as the routing slip level) of the Human Task Editor are applied. For example, if the global policy is set to escalate the task and this participant does not act in the duration provided, the task is escalated to the manager or another user, as appropriate.

To specify a time limit for acting on a task:

1. In the **Advanced** tab of the Edit Participant Type dialog box for the serial type, click the **Advanced** icon to expand the section shown in Figure 29-36.
2. Click **Limit allocated duration to**.
3. Specify the amount of time.

Note:

If you specify the task expiry time at the level of a serial participant, then, when that specified time limit is reached, the task does not move to the next participant in the series. Rather, the entire task expires.

For more information about setting the global escalation and renewal policies in the **Deadlines** section of the Human Task Editor, see Escalating_ Renewing_ or Ending the Task.

Inviting Additional Participants to a Task

You can allow a task assignee to invite other participants into the workflow before routing it to the next assignee in this workflow. For example, assume the approval workflow goes from James Cooper to John Steinbeck. If this option is checked, James Cooper can decide to first route it to Irving Stone before it goes to John Steinbeck.

To invite additional participants to a task:

1. In the **Advanced** section of the Edit Participant Type dialog box for the serial type, click the **Advanced** icon to expand the section (if not already expanded).
2. Select **Allow this participant to invite other participants**.

Note:

For the serial participant type, additional participants can be invited as follows:

- Globally specifying that the ad hoc participants can be invited at anytime. In this case, even in a sequential workflow, approvers can invite other participants at any level in the sequential workflow.
- Specifying that an ad hoc invitation of other participants can be done only in specific points in the workflow. In this case, other ad hoc participants are invited only when a series is complete.

Bypassing a Task Participant

You can bypass a task participant (user, group, or application role) if a specific condition is satisfied. For example, if a user submits a business trip expense report that is under a specific amount, no approval is required by their manager.

To bypass a task participant:

1. In the **Advanced** section of the Edit Participant Type dialog box for the serial type, select the **Specify skip rule** check box.

   This action displays an icon for accessing the Expression Builder dialog box for building a condition. The expression must evaluate to a boolean value.

   For more information about a valid XPath expression for skipping a participant, see Bypassing a Task Participant.

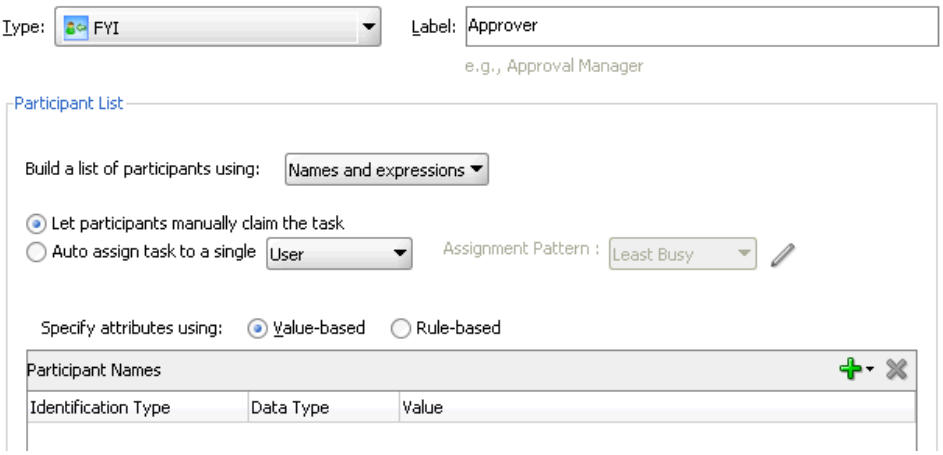
How to Configure the FYI Participant Type

Figure 29-38 displays the Edit Participant Type dialog box for the FYI type. This dialog box also includes a **Participants Exclusion List** at the bottom that is not displayed in Figure 29-38.

This participant type is used when a task is sent to a user, but the business process does not wait for a user response; it just continues. FYIs cannot directly impact the outcome of a task, but in some cases can provide comments or add attachments.

For example, a magazine subscription is due for renewal. If the user does not cancel the current subscription before the expiration date, the subscription is renewed. This user is reminded weekly until the request expires or the user acts on it.

Figure 29-38 Edit Participant Type — FYI Type



To configure the FYI participant type:

1. In the **Label** field, enter a recognizable label for this participant. This label must be unique among all the participants in the task definition (for example, `Approval Manager`, `Primary Reviewers`, and so on).

Creating an FYI Task Participant List

Users assigned to the list of participants can act upon tasks. You can create several types of lists:

- Value-based name and expression lists
- Value-based management chain lists
- Rule-based names and expression lists
- Rule-based management chain lists
- Rule-based lists

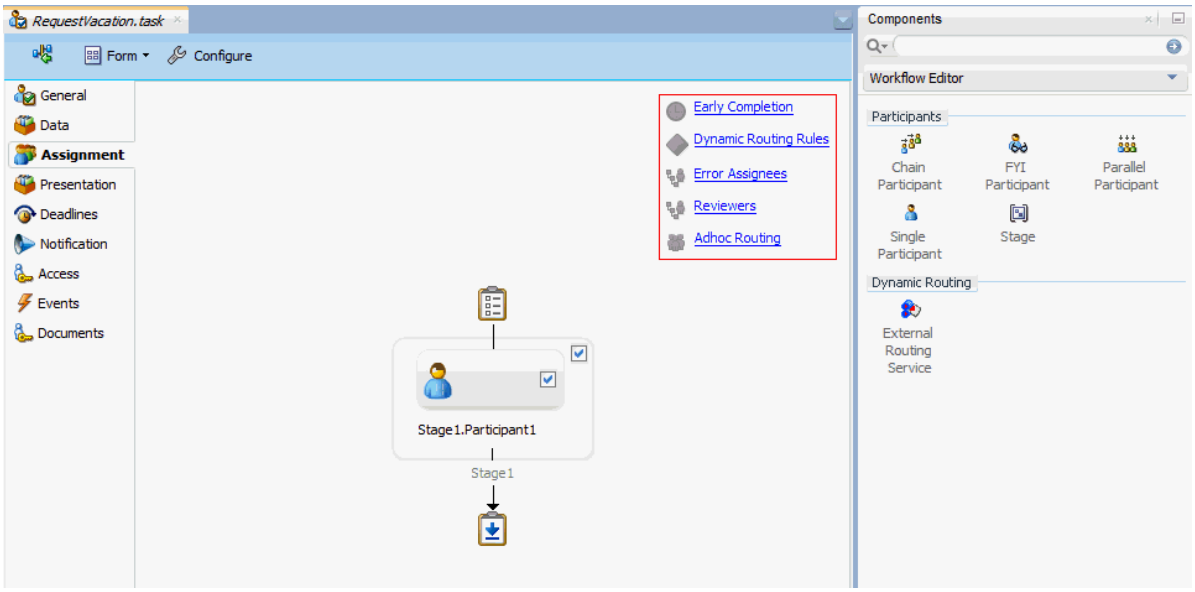
See section Creating a Single Task Participant List for instructions on creating these lists of participants.

Selecting a Routing Policy

This section describes how to route tasks to participants in a certain order and enabling participants to invite others.

After you configure a participant type and are returned to the Human Task Editor, click the **Task will go from starting to final participant** icon, as shown in Figure 29-39.

Figure 29-39 Human Task Editor — Assignment Section



This displays the Configure Assignment dialog box shown in Figure 29-40 for specifying a method for routing your task through the workflow.

Figure 29-40 Configure Assignment

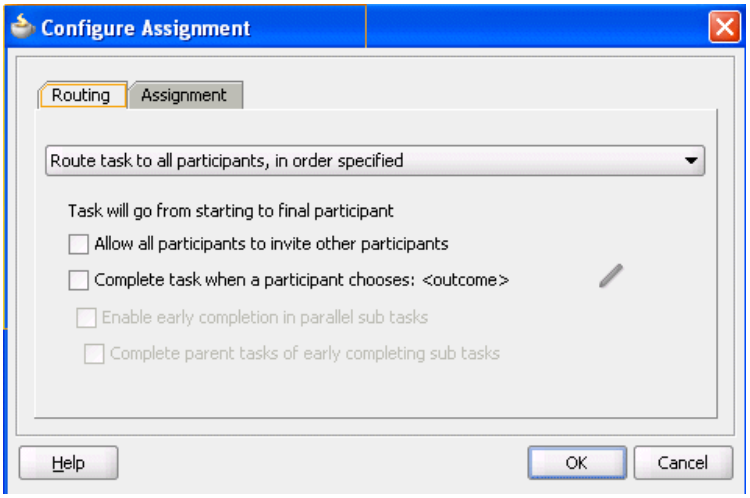


Table 29-10 describes the routing policy methods provided.

Table 29-10 Routing Policy Method

Routing Policy Selection	Use This Policy In Environments Where...	Section
• **Allow all participants to invite other participants**	A participant can select users or groups as the next assignee (ad hoc) when approving the task.	Allow All Participants to Invite Other Participants or Edit New Participants

Table 29-10 (Cont.) Routing Policy Method

Routing Policy Selection	Use This Policy In Environments Where...	Section
• **Complete task when a participant chooses: <outcome>**	A participant in a task can accept or reject it, thus ending the workflow without the task being sent to any other participant. For example, a manager rejects a purchase order, meaning that purchase order is not sent to their manager for review.	Stopping Routing of a Task to Further Participants
• **Enable early completion in parallel subtasks**	**Note:** This option is for environments in which you have multiple stages and participants working in parallel. Participants perform subtasks in parallel, and one group's rejection or approval of a subtask does not cause the other group's subtask to also be rejected or approved.	Enabling Early Completion in Parallel Subtasks
• **Complete parent tasks of early completing subtasks**	**Note:** This option is for environments in which you have multiple stages and participants working in parallel. Participants perform subtasks in parallel, and one group's rejection or approval of a subtask causes the other group's subtask to also be rejected or approved.	Completing Parent Subtasks of Early Completing Subtasks
Use Advanced Rules	The participants to whom the task is routed are determined by the business rule logic that you model. For example, a loan application task is designed to go through a loan agent, their manager, and then the senior manager. If the loan agent approves the loan, but their manager rejects it, the task is returned to the loan agent.	How to Specify Advanced Task Routing Using Business Rules
Use External Routing	The participants in a task are dynamically determined. For example, a company's rules may require the task participants to be determined and then retrieved from a back-end database during runtime.	How to Use External Routing
Assignment tab	A participant is assigned a failed task for the purposes of recovery.	How to Configure the Error Assignee and Reviewers

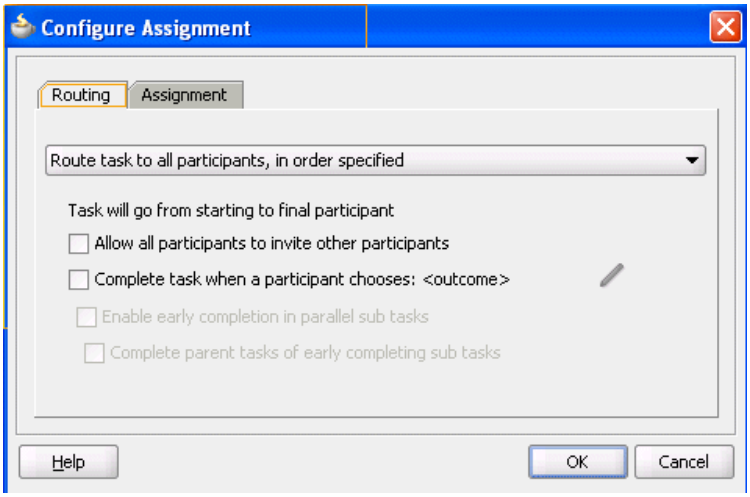
How to Route Tasks to All Participants in the Specified Order

You can select to have a task reviewed by all selected participants. This is known as default routing because the task is routed to each of the participants in the order in which they appear. This type of routing differs from state machine-based routing.

To route tasks to all participants in the specified order:

1. In the **Assignment** section, click the icon to the right of **Task will go from starting to final participant**.
2. Select **Route task to all participants, in order specified** from the list shown in Figure 29-41.

Figure 29-41 Route a Task to All Participants



See the following tasks to define a routing policy:

- Allowing all participants to invite other participants
- Completing a task when a participant chooses
- Enabling early completion in parallel subtasks
- Completing parent subtasks of early completing subtasks

Allow All Participants to Invite Other Participants or Edit New Participants

This check box is the equivalent of the ad hoc workflow pattern of pre-10.1.3 Oracle BPEL Process Manager releases. This applies when there is at least one participant. In this case, each user selects users or groups as the next assignee when approving the task.

To allow all participants to invite other participants:

1. Select **Route task to all participants, in order specified**.
2. Under Adhoc Routing, select the **Allow all participants to invite other participants** check box for this task assignee to invite other participants into the workflow before routing it to the next assignee in this workflow.
3. Select the **Allow participants to edit new participants** check box for this task assignee to edit other adhoc participants that were added to the routing slip.

Note:

Do not add adhoc assignees either above or below an FYI participant.

Allow Initiator to Add Participants

Under **Adhoc Routing**, select the **Allow all initiator to add participants** check box so this task initiator can invite other participants into the workflow before routing to the next assignee in this workflow.

Stopping Routing of a Task to Further Participants

You can specify conditions under which to complete a task early, regardless of the other participants in the workflow.

For example, assume an expense report goes to the manager, and then the director. If the first participant (manager) rejects it, you can end the workflow without sending it to the next participant (director).

To abruptly complete a condition:

1. Under Early Completion, select the **Complete task when a participant chooses: <outcome>** check box.

   The Abrupt Completion Details dialog box appears.

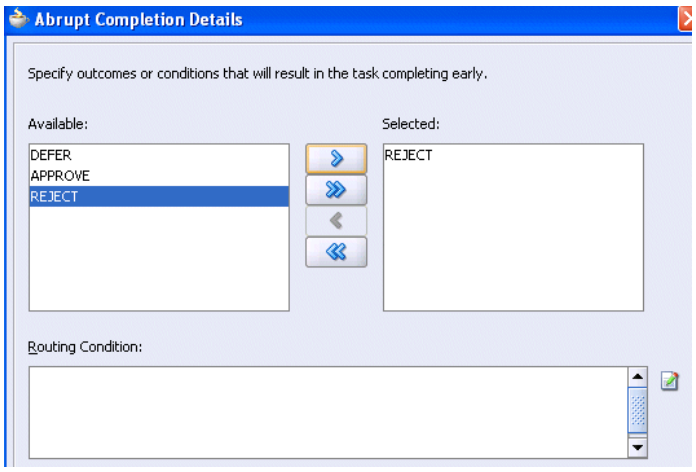
   There are two methods for specifying the abrupt completion of a task:

   - Outcomes
   - XPath expression routing condition

   If outcomes are specified, any time the selected task outcome occurs, the task completes. If both outcome and routing condition are specified, the workflow service performs a logical `OR` operation on the two.

2. Select appropriate outcomes and click the > button, as shown in Figure 29-42. To select all, click the >> button.

   Figure 29-42 Abrupt Completion Details

3. To the right of the **Routing Condition** field, click the icon to display the Expression Builder dialog box for dynamically creating a condition under which to complete this task early. For example, if a user submits a business trip expense report that is under a specific amount, no approval is required by their manager.

An early completion XPath expression is not evaluated until at least one user has acted upon the task.

4. To enable early completion, click **Enable early completion in parallel with subtasks**. For more information, see Enabling Early Completion in Parallel Subtasks.

5. To enable early completion of parent tasks, click **Complete parent tasks of early completing subtasks**. For more information, see Completing Parent Subtasks of Early Completing Subtasks.

6. Click **OK** to return to the Human Task Editor.

   You can click the icon to the right of the **Complete task when a participant chooses: <outcome>** check box to edit this information.

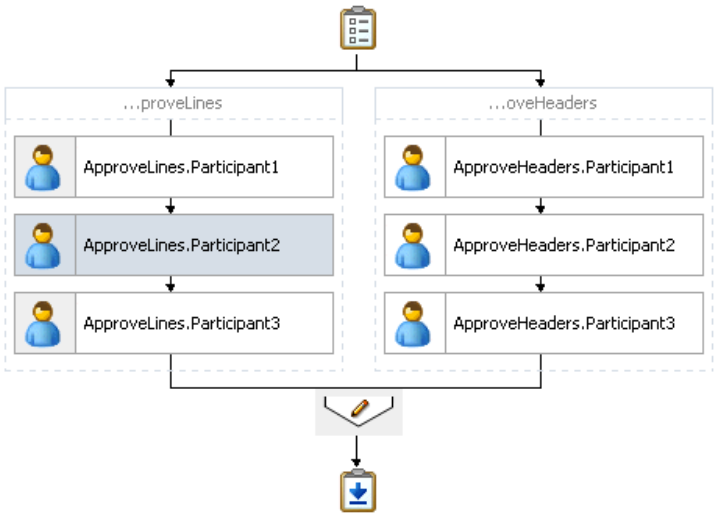
Enabling Early Completion in Parallel Subtasks

You can use this option in the following environments:

- Multiple stages and groups of participants perform subtasks in parallel.
- A participant in one group approves or rejects a subtask, which causes the other participants in that same group to stop acting upon the task. However, this does not cause the other parallel group to stop acting upon subtasks. That group continues taking actions on tasks.

For example, assume there are two parallel subgroups, each in separate stages. One group acts upon lines of a purchase order. The other group acts upon headers of the same purchase order. If participant **ApproveLines.Participant2** of the first group rejects a line, all other task participants in the first group stop acting upon tasks. However, the second parallel group continues to act upon headers in the purchase order. In this scenario, the entire task does not complete early. Figure 29-43 provides details.

Figure 29-43 Early Completion of Parallel Subtasks



Completing Parent Subtasks of Early Completing Subtasks

You can use this option in the following environments:

- Multiple stages and groups of participants perform subtasks in parallel.
- A participant in one group approves or rejects a subtask, which causes the other participants in that same group to stop acting upon the task. This also causes the other parallel group to stop acting upon subtasks.

For example, assume there are two parallel subgroups, each in separate stages, as shown in Figure 29-43. One group acts upon lines of a purchase order. The other group acts upon headers of the same purchase order. If participant **ApproveLines.Participant2** of the first group rejects a line, all other task participants in the first group stop acting upon tasks. In addition, the second parallel group stops acting upon headers in the purchase order. In this scenario, the entire task completes early.

How to Specify Advanced Task Routing Using Business Rules

Use advanced routing rules to create complex workflow routing scenarios. The participant types (single, parallel, serial, and FYI) are used to create a linear flow from one set of users to another with basic conditions such as abrupt termination, skipping assignees, and so on. However, there is often a need to perform more complex back and forth routing between multiple individuals in a workflow. One option is to use the BPEL process as the orchestrator of these tasks. Another option is to specify it declaratively using business rules. This section describes how you can model such complex interactions by using business rules with the Human Task Editor.

Introduction to Advanced Task Routing Using Business Rules

You can define state machine routing rules using Oracle Business Rules. This action enables you to create Oracle Business Rules that are evaluated:

- After a routing slip task participant sets the outcome of the task
- Before the task is assigned to the next routing slip participant

This action enables you to override the standard task routing slip method described in How to Route Tasks to All Participants in the Specified Order and build complex routing behavior into tasks.

Using Oracle Business Rules, you define a set of rules (called a ruleset) that relies on business objects, called facts, to determine which action to take.

Facts

A fact is an object with certain business data. Each time a routing slip assignee sets the outcome of a task, instead of automatically routing the task to the next assignee, the task service performs the following steps:

- Asserts facts into the decision service
- Executes the advanced routing ruleset

Rules can test values in the asserted facts and specify the routing behavior by setting values in a `TaskAction` fact type.

Table 29-11 describes the fact types asserted by the task service.

Table 29-11 Fact Types Asserted By the Task Service

Table 29-11 (Cont.) Fact Types Asserted By the Task Service

Fact Type	Description
`Task`	This fact contains the current state of the workflow task instance. All task attributes can be tested against it. The task fact also contains the current task payload. This fact enables you to construct tests against payload values and task attribute values.
`PreviousOutcome`	This fact describes the previous task outcome and the assignee who set the outcome. The previous outcome fact contains the following attributes: • `actualParticipant`: The name of the participant who set the task outcome (for example, `jstein`) • `logicalParticipant`: The logical name (or label) for the routing slip participant responsible for setting the task outcome (for example, `assignee1`) • `outcome`: The outcome that was set (for example, approve or reject) • `level`: If the previous participant was part of a management chain, then this attribute records their level in the chain, where 1 is the first level in the chain. For other participant types, the value is -1. • `totalNumberOfApprovals`: The total number of users that have now set the outcome of the task.
`TaskAction`	This fact is not intended for writing rule tests against it. Instead, it is updated by the ruleset, and returned to the task service to indicate how the task should be routed. Rules should not directly update the `TaskAction` fact. Instead, they should call one of the RL functions described in Action Types. These functions handle updating the `TaskAction` fact with the appropriate values.

Some fact types can only be used in workflow routing rules, while others can only be used in workflow participant rules. Table 29-12 describes where you can use each type.

Table 29-12 Use of Fact Types

Fact Type	Can Use in Routing Rules?	Can Use in Participant Rules?
`Task`	Yes	Yes
`PreviousOutcome`	Yes	No
`TaskAction`	Yes	No
`Lists`	No	Yes
`RoutingSlipObjectFactory`	No	Yes
`ResourceListType`	No	Yes
`ManagementChainListType`	No	Yes
`ResourceType`	No	Yes
`ParameterType`	No	Yes

Table 29-12 (Cont.) Use of Fact Types

Fact Type	Can Use in Routing Rules?	Can Use in Participant Rules?
`AutoActionType`	No	Yes
`ResponseType`	No	Yes

Action Types

To instruct the task service on how to route the task, rules can specify one of many task actions. This is done by updating the `TaskAction` fact asserted into the rule session. However, rules should not directly update the `TaskAction` fact. Instead, rules should call one of the action RL functions, passing the `TaskAction` fact as a parameter. These functions handle the actual updates to the fact. For example, to specify an action of go forward, you must add a `call GO_FORWARD(TaskAction)` to the action part of the rule.

Each time a state machine routing rule is evaluated, the rule takes one of the actions shown in Table 29-13:

Table 29-13 Business Rule Actions

Action	Description	Parameters
`GO_FORWARD`	Goes to the next participant in the routing slip (default behavior).	None
`PUSHBACK`	Goes back to the previous participant in the routing slip (the participant before the one that just set the task outcome). **Note:** Pushback is designed to work with single approvers and not with group votes. Pushback from a stage with group vote (or parallel) scenario to another stage is not allowed. Similarly, you cannot push back from a single assignee to a group vote (or parallel) scenario.	None
`GOTO`	Goes to a specific participant in the routing slip.	`participant'` A string that identifies the label of the participant (for example, `Approver1`) to which to route the task.
`COMPLETE`	Finishes routing and completes the task. The task is marked as completed, and no further routing is required.	None
`ESCALATE`	Escalates and reassigns the task according to the task escalation policy (usually to the manager of the current assignee).	None

Sample Ruleset

This section describes how to use rules to implement custom routing behavior with a simple example. A human workflow task is created for managing approvals of expense requests. The outcomes for the task are approve and reject. The task definition

includes an `ExpenseRequest` payload element. One of the fields of `ExpenseRequest` is the total amount of the expense request. The routing slip for the task consists of three single participants (`assignee1`, `assignee2`, and `assignee3`).

By default, the task gets routed to each of the assignees, with each assignee choosing to approve or reject the task.

Instead of this behavior, the necessary routing behavior is as follows:

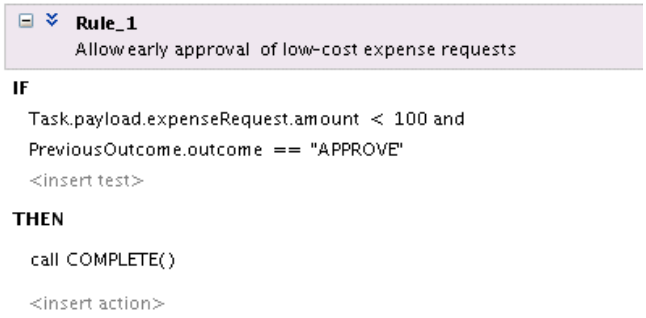
- If the total amount of the expense request is less than $100, approval is only required from one of the participants. Otherwise, it must be approved by all three.
- If an expense request is rejected by any of the participants, it must be returned to the previous participant for re-evaluation. If it is rejected by the first participant, the expense request is rejected and marked as completed.

This behavior is implemented using the following rules. When a rule dictionary is generated for advanced routing rules, it is created with a template rule that implements the default `GO_FORWARD` behavior. You can edit this rule, and make copies of the template rule by right-clicking and selecting **Copy Rule** in the Oracle Business Rules Designer.

If the amount is greater than $100 and the previous assignee approved the task, it is not necessary to provide a rule for routing a task to each of the assignees in turn. This is the default behavior that is reverted to if none of the rules in the ruleset are triggered:

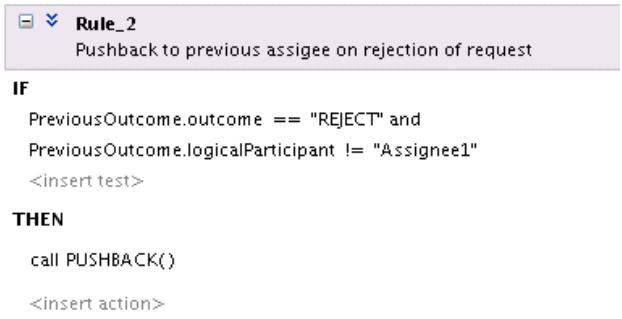
- Early approval rule (Figure 29-44):

Figure 29-44 Early Approval Rule



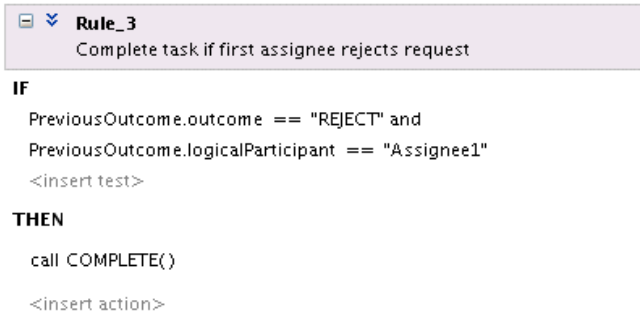
- Push back on the rejected rule (Figure 29-45):

Figure 29-45 Push Back On The Rejected Rule



- Complete the `Assignee1` rejected rule (Figure 29-46):

Figure 29-46 Completion of the Assignee1 Rejected Rule



For information about iterative design, see the `workflow-106-IterativeDesign` sample available with the Oracle SOA Suite samples.

Linked Dictionary Support

For human workflow, business rule artifacts are now stored in two rules dictionaries. This is useful for scenarios in which you must customize your applications. For example, you create and ship version 1 of an application to a customer. The customer then customizes the rulesets in the application with Oracle SOA Composer. Those customizations are now stored in a different rules dictionary than the base rules dictionary. The rules dictionary that stores the customized rulesets links with the rules in the base dictionary. When you later ship version 2 of the application, the base rule dictionary may contain additional changes introduced in the product. The ruleset customization changes previously performed by the customer are preserved and available with the new changes in the base dictionary. When an existing application containing a task using rules is opened, if the rules are in the old format using one dictionary, they are automatically upgraded and divided into two rules dictionaries:

- Base dictionary
- Custom dictionary

For more information about customizations, see the chapter on customizing SOA composite applications in *Developing SOA Applications with Oracle SOA Suite*.

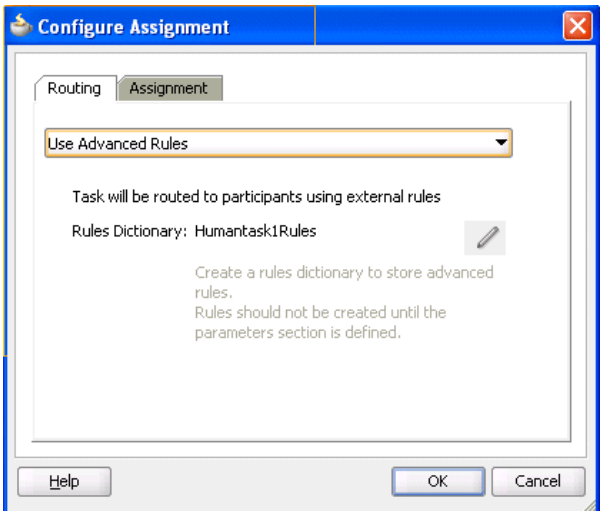
Creating Advanced Routing Rules

To create advanced routing rules:

1. In the **Assignment** section, click the icon to the right of **Task will go from starting to final participant**.
2. Select **Use Advanced Rules** from the list.
3. Select **Dynamic Routing Rules**.

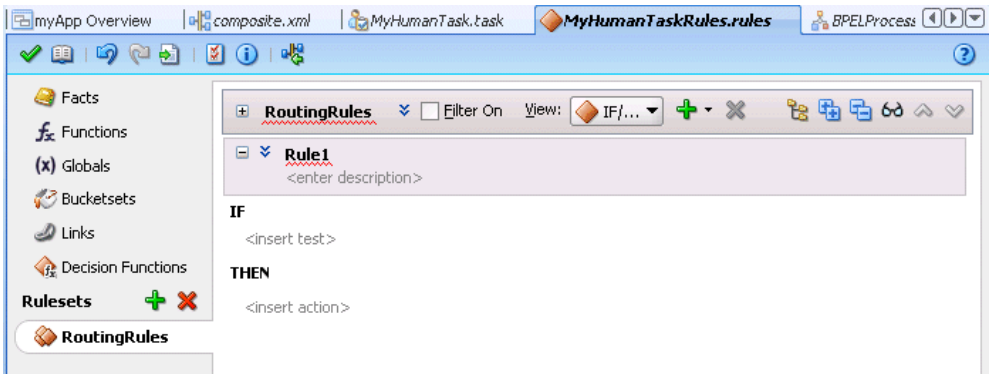
   The Use Advanced Rules edit box displays.
4. To the right of **Rules Dictionary**, click the **Edit** icon, as shown in Figure 29-47.

Figure 29-47 Creating a Rules Dictionary



This starts the Oracle Business Rules Designer with a pre-seeded repository containing all necessary fact definitions, as shown in Figure 29-48. A decision service component is created for the dictionary, and is associated with the task service component.

Figure 29-48 Human Task Rule Dictionary



5. Define state machine routing rules for your task using Oracle Business Rules.

   This automatically creates a fully-wired decision service in the human task and the associated rule repository and data model.

   For more information about business rules, see the following documentation:

   - Sample Ruleset for an example human task ruleset
   - *Designing Business Rules with Oracle Business Process Management*
   - *Rules Language Reference for Oracle Business Process Management*

How to Use External Routing

You configure an external routing service that dynamically determines the participants in the workflow. If this routing policy is specified, all other participant types are ignored. It is assumed that the external routing service provides a list of participant types (single approver, serial approver, parallel approver, and so on) at runtime to determine the routing of the task.

Use this option if you do not want to use any of the routing rules to determine task assignees. In this case, all the logic of task assignment is delegated to the external routing service.

Note:

If you select **Use External Routing** in the Configure Assignment dialog box, specify a Java class, and click **OK** to exit, the next time you open this dialog box, the other two selections (**Route task to all participants, in order specified** and **Use Advanced Rules**) no longer appear in the drop-down list. To access all three selections again, you must delete the entire assignment.

To use external routing

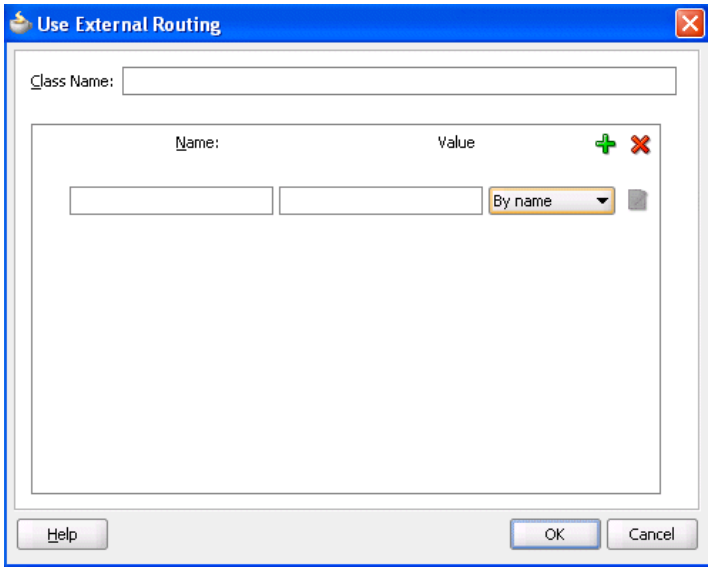
1. Drag and drop **External Routing Service** from the Workflow Editor Components window.

   The Use External Routing edit box displays.

2. Click the **Edit** icon.

   The External Routing dialog box appears, as shown in Figure 29-49.

Figure 29-49 Use External Routing Dialog



3. In the **Class Name** field, enter the fully qualified class file name (for example, the `org.mycompany.tasks.RoutingService` class name). This class must implement the following interface:

   `oracle.bpel.services.workflow.task.IAssignmentService`

4. Add name and pair value parameters by name or XPath expression that can be passed to the external service, as shown in Table 29-14.

Table 29-14 External Routing

Table 29-14 (Cont.) External Routing

Field	Description
By Name	Enter a name in the **Name** field and a value in the **Value** field.
By Expression	Enter a name and dynamically enter a value by clicking the icon to the right of the field to display the Expression Builder dialog box.

5. Click the **Add** icon to add additional name and pair value parameters.

How to Configure the Error Assignee and Reviewers

Tasks can error for reasons such as incorrect assignments. When such errors occur, the task is assigned to the error assignee, who can perform corrective actions. Recoverable errors are as follows:

- Invalid user and group for all participants
- Invalid XPath expressions that are related to assignees and expiration duration
- Escalation on expiration errors
- Evaluating escalation policy
- Evaluating renewal policy
- Computing a management chain
- Evaluating dynamic assignment rules. The task is not currently in error, but is still left as assigned to the current user and is therefore recoverable.
- Dynamic assignment cyclic assignment (for example, user A > user B > user A). The task is not currently in error, but is still left as assigned to the last user in the chain and is therefore recoverable.

The following errors are not recoverable. In these cases, the task is moved to the terminating state `ERRORED`.

- Invalid task metadata
- Unable to read task metadata
- Invalid `GOTO` participant from state machine rules
- Assignment service not found
- Any errors from assignment service
- Evaluating custom escalate functions
- Invalid XPath and values for parallel default outcome and percentage values

During modeling of workflow tasks, you can specify error assignees for the workflow. If error assignees are specified, they are evaluated and the task is assigned to them. If no error assignee is specified at runtime, an administration user is discovered and is assigned the alerted task. The error assignee can perform one of the following actions:

- Ad hoc route

Route the task to the actual users assigned to the task. Ad hoc routing allows the task to be routed to users in sequence, parallel, and so on. **Note:** Do not add adhoc assignees either above or below a FYI participant.

- Reassign

  Reassign the task to the actual users assigned to this task

- Error task

  Indicate that this task cannot be rectified.

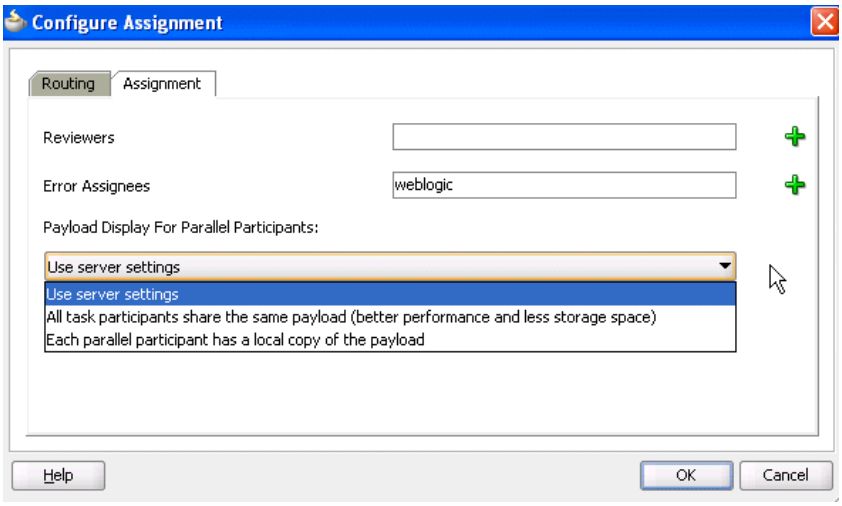
If there are any errors in evaluating the error assignees, the task is marked as being in error.

This dialog box enables you to specify the users or groups to whom the task is assigned if an error in assignment has occurred.

To configure the error assignee:

1. Click the **Add** icon to assign reviewers or error assignees, as shown in Figure 29-50.

   Figure 29-50 Error Assignment Details

2. Click the **Add** icon and select a user, group, or application role to participate in this task.

   The **Identification Type** column of the **Starting Participant** table displays your selection of user, group, or application role.

3. See Step 5 through 7 of Creating a Single Task Participant List for instructions on selecting a user, group, or application role.

4. If you are using parallel participant types, you can specify where to store the subtask payload with the following options.

   - **Use server settings**

     The **SharePayloadAcrossAllParallelApprovers** System MBean Browser boolean property in Oracle Enterprise Manager Fusion Middleware Control determines whether to share the payload of subtasks in the root task. By default, this property is set to **true**. If set to **true**, the All task participants share

the same payload (better performance and less storage space) option is used. If this property is set to **false**, the **Each parallel participant has a local copy of the payload** option is used. To change this property, perform the following steps:

a. Right-click **soa-infra** and select **Administration** > **System MBean Browser**.

b. Expand **Application Defined MBeans** > **oracle.as.soainfra.config** > **Server:** *server_name* > **WorkflowConfig** > **human-workflow**.

c. Click **SharePayloadAcrossAllParallelApprovers**.

d. Change this property in the list, and click **Apply**.

- **All task participants share the same payload (better performance and less storage space)**

  The payload for the subtasks is stored in their root task. This situation means that the payload of the root task is shared across all its subtasks. Internally, this option provides better performance and storage space consumption. Less storage space is consumed because the payload of the root task is shared across all its subtasks.

- **Each parallel participant has a local copy of the payload**

  Each subtask has its own copy of the payload. Internally, this option provides lesser performance and storage space consumption because more storage space is consumed.

5. Click **OK**.

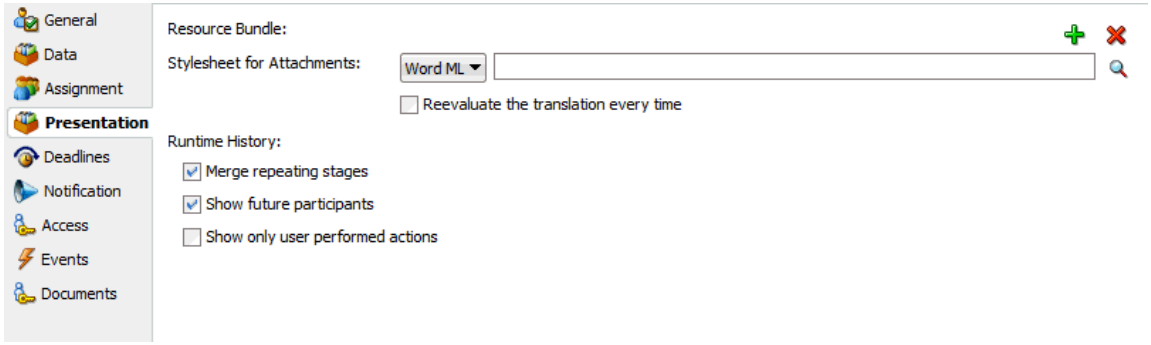
For more information about users, groups, or application roles, see Participant Assignment.

Specifying Multilingual Settings and Style Sheets

You can specify resource bundle for displaying task details in different languages.

The **Presentation** section shown in Figure 29-51 enables you to specify resource bundles for displaying task details in different languages in Oracle BPM Worklist and WordML and custom style sheets for attachments.

Figure 29-51 Presentation Section



How to Specify WordML and Other Style Sheets for Attachments

To specify WordML style sheets for attachments:

1. In the **Stylesheet for Attachments** list of the **Presentation** section, select one of the following options:
   - **Word ML**: This option dynamically creates Microsoft Word documents for sending as email attachments using a WordML XSLT style sheet. The XSLT style sheet is applied on the task document.
   - **Other**: This option creates email attachments using an XSLT style sheet. The XSLT style sheet is applied on the task document.
2. Click the **Search** icon to select the style sheet as an attachment.

How to Specify Multilingual Settings

You can specify resource bundles for displaying task details in different languages in Oracle BPM Worklist. Resource bundles are supported for the following task details:

- Displaying the value for task outcomes in plain text or with the `message(key)` format.
- Making email notification messages available in different languages. At runtime, you specify the `hwf:getTaskResourceBundleString(taskId, key, locale?)` XPath extension function to obtain the internationalized string from the specified resource bundle. The locale of the notification recipient can be retrieved with the function `hwf:getNotificationProperty(propertyName)`.

Resource bundles can also simply be property files. For example, a resource bundle that configures a display name for task outcomes can look as follows:

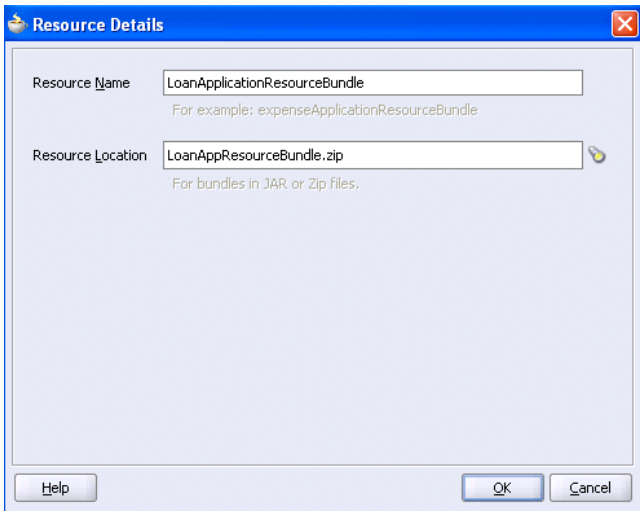
- `APPROVE=Approve`
- `REJECT=Reject`

To specify multilingual settings:

1. In the **Presentation** section, click the **Add** icon across from **Resource Bundle**.

   The Resource Details dialog box shown in Figure 29-52 appears.

Figure 29-52 Resource Details Dialog



2. In the **Resource Name** field, enter the name of the resource used in the resource bundle. This should be a `.properties`-based resource bundle file.

3. In the **Resource Location** field, click the **Search** icon to select the JAR or ZIP resource bundle file to use. The resource bundle is part of your system archive (SAR) file.

   If the resource bundle is outside of the composite project, you are prompted to place a local copy in `SCA-INF/lib`.

   If the resource bundle file is not in the composite class loader (directly under `SCA-INF/classes` or in a JAR file in `SCA-INF/lib`), you must specify its location. For example, if the resource bundle is accessible from a location outside of the composite class loader (for example, an HTTP location such as `http://`*`host`*`:`*`port`*`/bundleApp/taskBundles.jar`), then this location must be specified in this field.

4. Click **OK** to return to the Human Task Editor.

   For more information, see the section on configuring notification messages in different languages in *Developing SOA Applications with Oracle SOA Suite*.

Specify What to Show in Task Details in the Worklist

The Presentation section enables you to specify the records in the runtime history section of the task details form in `worklistapp`.

Merge repeating stages: Select this option to view one aggregated entry for all repeating stages. The Worklist UI also provides an option to set or unset this option.

Show future participants: Select this option to see details about all future participants in the task.

Show only user performed actions: By default, task history details contain records for Admin and system actions, such as root task updates. Select this option to not see only user-performed action updates in the task details.

Escalating, Renewing, or Ending the Task

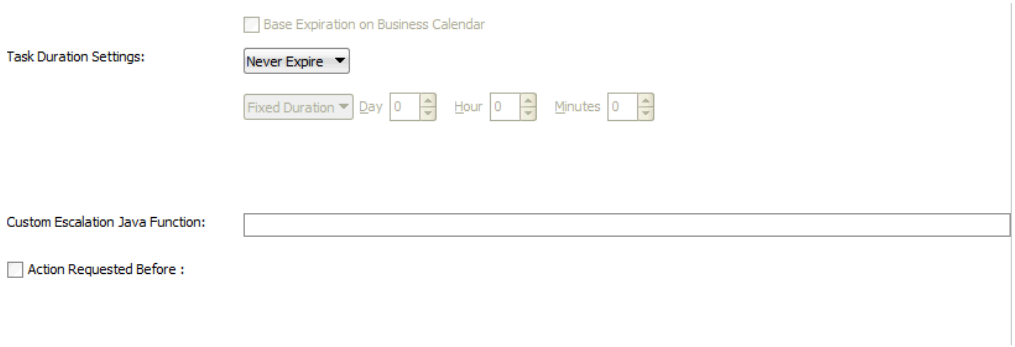
You can specify the expiration duration of a task in this global policy section (also known as the routing slip level).

If the expiration duration is specified at the routing slip level instead of at the participant type level, then this duration is the expiration duration of the task across all the participants. However, if you specify expiration duration at the participant type level (through the **Limit allocated duration to** check box), then those settings take precedence over settings specified in the **Deadlines** section (routing slip level).

Figure 29-53 shows the **Deadlines** section of the Human Task Editor.

You can also specify that a task be escalated to a user's manager after a specified time period. For more information, see Specifying a Time Limit for Acting on a Task.

Figure 29-53 Human Task Editor — Deadlines Section



Introduction to Escalation and Expiration Policy

This section provides an overview of how specifying the expiration duration at this level makes this setting the expiration duration of the task across all the participants.

For example, participant **LoanAgentGroup** and participant **Supervisor** have three days to act on the task between them, as shown in Figure 29-54:

Figure 29-54 Expire After Policy

If there is no expiration specified at either the participant level or this routing slip level, then that task has no expiration duration.

If expiration duration is specified at any level of the participants, then for that participant, the participant expiration duration is used. However, the global expiration duration is still used for the participants that do not have participant level expiration duration. The global expiration duration is always decremented by the time elapsed in the task.

The policy for interpreting the participant level expiration for the participants is described as follows:

- Serial

  Each assignment in the management chain gets the same expiration duration as the one specified in the serial. The duration is not for all the assignments resulting from this assignment. If the task expires at any of the assignments in the management chain, the escalation and renewal policy is applied.

- Parallel:

  – In a parallel workflow, if the parallel participants are specified as a resource, a routing slip is created for each of the resources. The expiration duration of each created routing slip follows these rules:

    ◆ The expiration duration equals the expiration duration of the parallel participant if it has an expiration duration specified.

    ◆ The expiration duration that is left on the task if it was specified at the routing slip level.

    ◆ Otherwise, there is no expiration duration.

  – If parallel participants are specified as routing slips, then the expiration duration for the parallel participants is determined by the routing slip.

Note:

When the parent task expires in a parallel task, the subtasks are withdrawn if those tasks have not expired or completed.

How to Specify a Policy to Never Expire

You can specify for a task to never expire.

To specify a policy to never expire:

1. In the drop-down list in the **Deadlines** section, as shown in Figure 29-53, select **Never Expire**.

How to Specify a Policy to Expire

You can specify for a task to expire. When the task expires, either the escalation policy or the renewal policy at the routing slip level is applied. If neither is specified, the task expires. The expiration policy at the routing slip level is common to all the participants.

To specify for a task to expire:

1. In the drop-down list of the **Deadlines** section, select **Expire after**, as shown in Figure 29-55.
2. Specify the maximum time period for the task to remain open.

   The expiration policy for parallel participants is interpreted as follows:

- If parallel participants are specified as resources in parallel elements, there is no expiration policy for each of those participants.
- If parallel participants are specified as routing slips, then the expiration policy for the routing slip applies to the parallel participants.

Figure 29-55 indicates that the task expires in three days.

Figure 29-55 Expire After Policy

How to Extend an Expiration Policy Period

You can extend the expiration period when the user does not respond within the allotted time. You do this by specifying the number of times the task can be renewed upon expiration (for example, renew it an additional three times) and the duration of each renewal (for example, three days for each renewal period).

To extend an expiration policy period:

1. In the drop-down list of the **Deadlines** section, select **Renew after**, as shown in Figure 29-56.
2. Specify the maximum number of times to continue renewing this task.

   In Figure 29-56, when the task expires, it is renewed at most three times. It does not matter if the task expired at the **LoanAgentGroup** participant or the **Supervisor** participant.

Figure 29-56 Renew After Policy

How to Escalate a Task Policy

You can escalate a task if a user does not respond within the allotted time. For example, if you are using the escalation hierarchy configured in your user directory, the task can be escalated to the user's manager. If you are using escalation callbacks, the task is escalated to whoever you have defined. When a task has been escalated the

maximum number of times, it stops escalating. An escalated task can remain in a user inbox even after the task has expired.

To escalate a task policy:

1. In the drop-down list of the **Deadlines** section, select **Escalate after**, as shown in Figure 29-57.
2. Specify the following additional values. When both are set, the escalation policy is more restrictive.

   - **Maximum Escalation Levels**

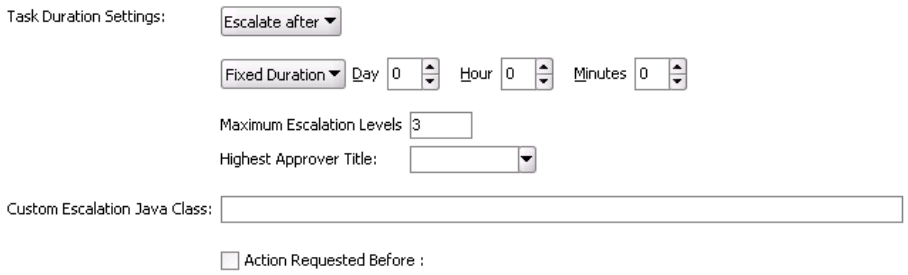
     Number of management levels to which to escalate the task. This field is required.

   - **Highest Approver Title**

     The title of the highest approver (for example, self, manager, director, or CEO). These titles are compared against the title of the task assignee in the corresponding user repository. This field is optional.

   The escalation policy specifies the number of times the task can be escalated on expiration and the renewal duration. In Figure 29-57, when the task expires, it is escalated at most three times. It does not matter if the task expired at the **LoanAgentGroup** participant or the **Supervisor** participant.

Figure 29-57 Escalate After Policy



How to Specify Escalation Rules

This option allows a custom escalation rule to be plugged in for a particular workflow. For example, to assign the task to a current user's department manager on task expiration, you can write a custom task escalation function, register it with the workflow service, and use that function in task definitions.

The default escalation rule is to assign a task to the manager of the current user. To add a new escalation rule, follow these steps.

To specify escalation rules:

1. Implement the following interface:

   ```
   oracle.bpel.services.workflow.assignment.dynamic.IDynamicTaskEscalationFunction
   ```

   This implementation must be available in the class path for the server.

2. Log in to Oracle Enterprise Manager Fusion Middleware Control.

3. Expand the **SOA** folder in the navigator.
4. Right-click **soa-infra**, and select **SOA Administration** > **Workflow Config** > **Task** tab.

   The Workflow Task Service Properties page appears.
5. Add a new function. For example:
   - Function name: `DepartmentSupervisor`
   - Classpath: `oracle.bpel.services.workflow.assignment.dynamic.patterns.DepartmentSupervisor`
   - Function parameter name
   - Function parameter value
6. In the **Custom Escalation Java Class** field of the **Deadlines** section, enter the function name as defined in the Workflow Task Service Properties page for the escalation rule.

   For more information, see the section on custom escalation functions in *Developing SOA Applications with Oracle SOA Suite*.

How to Specify a Due Date

A due date indicates the date by which the task should be completed. The due date is different from the expiration date. When a task expires it is either marked expired or automatically escalated or renewed based on the escalation policy. The due date is generally a date earlier than the expiration date and an indication to the user that the task is about to expire.

You can enter a due date for a task, as shown in Figure 29-53. A task is considered overdue after it is past the specified due date. This date is in addition to the expiration policy. A due date can be specified irrespective of whether an expiration policy has been specified. The due date enables Oracle BPM Worklist to display a due date, list overdue tasks, filter overdue tasks in the inbox, and so on. Overdue tasks can be queried using a predicate on the `TaskQueryService.queryTask(...)` API.

To specify a due date:

1. In the **Deadlines** section, select the **Action Requested Before** check box.
2. Select **By Duration** to enter a time duration or select **By Expression** to dynamically enter a value as an XPath expression.

   Note the following details:
   - The due date can be set on both the task (using the Create ToDo Task dialog box in Oracle BPM Worklist) and in the `.task` file (using the Human Task Editor). This is to allow to-do tasks without task definitions to set a due date during initiation of the task. A due date that is set in the task (a runtime object) overrides a due date that is set in the `.task` file.
   - In the task definition, the due date can only be specified at the global level, and not for each participant.

- If the due date is set on the task, the due date in the `.task` file is ignored.
- If the due date is not set on the task, the due date in the `.task` file is evaluated and set on the task.
- If there is no due date on either the task or in the `.task` file, there is no due date on the task.

Note:

You cannot specify business rules for to-do tasks.

For more information on how to create a ToDo task, see *Developing SOA Applications with Oracle SOA Suite*.

Specifying Participant Notification Preferences

Notifications indicate when a user or group is assigned a task or informed that the status of the task has changed. Notifications can be sent through email, voice message, instant message, or SMS.

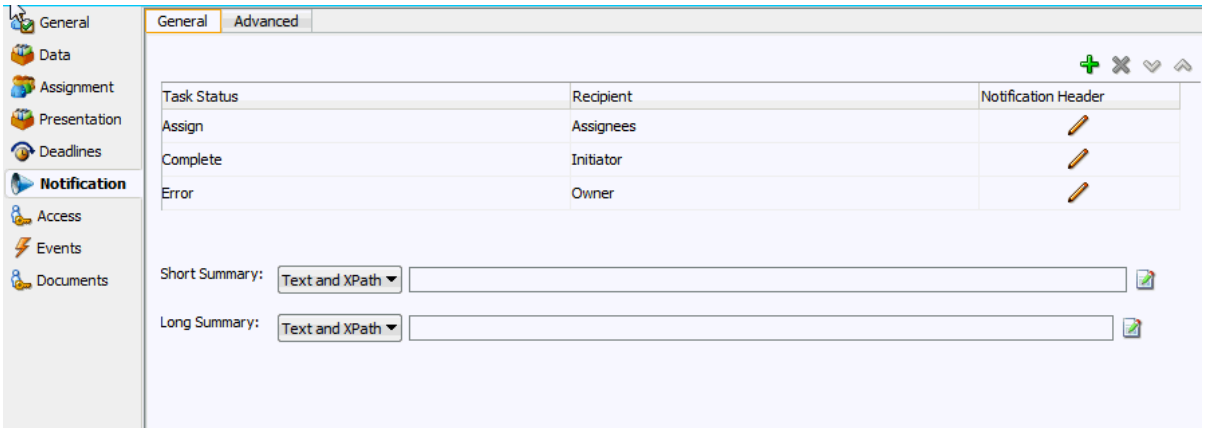
Figure 29-58 shows the **General** tab of the **Notification** section of the Human Task Editor (when fully expanded).

Notifications are sent to different types of participants for different actions. Notifications are configured by default with default messages. For example, a notification message is sent to indicate that a task has completed and closed. You can create your own or modify existing configurations.

Note:

Embedded LDAP does not support group email addresses. Therefore, when a task is assigned to a group ID, emails are sent to all of its members instead of to the group email address.

Figure 29-58 Human Task Editor — General Tab of Notification Section



To specify participant notification preferences:

1. Click the **Notification** tab (displays as shown in Figure 29-58).

Instructions for configuring the following subsections of the **General** tab of the **Notification** section are listed in Table 29-15.

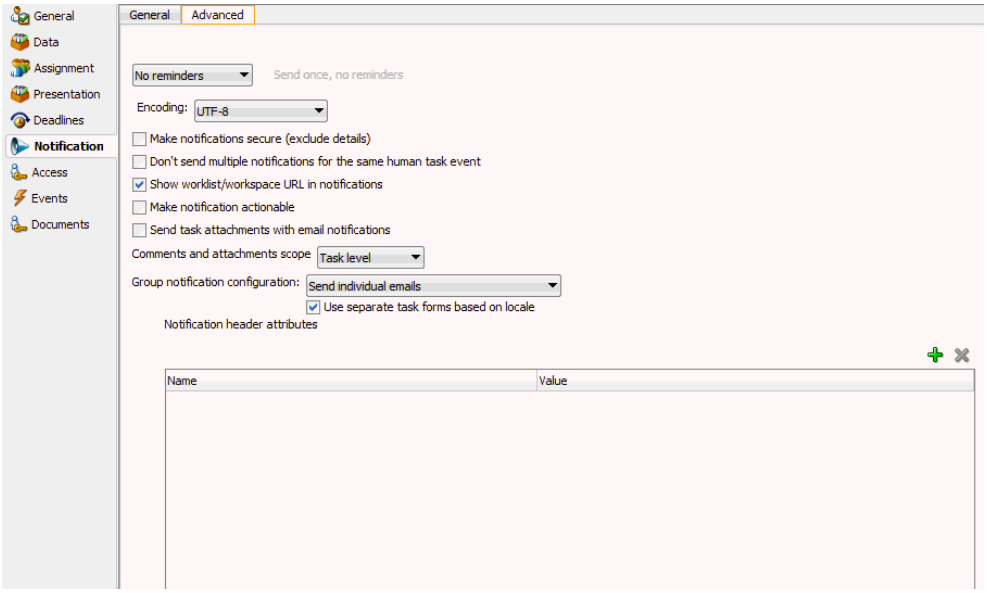
Table 29-15 Human Task Editor — General Tab of Notification Section

For This Subsection...	See...
Task Status **Recipient**	How to Notify Recipients of Changes to Task Status
Notification Header	How to Edit the Notification Message

For information about the notification service, see the section on notifications in *Developing SOA Applications with Oracle SOA Suite*.

2. In the **Notification** section, click the **Advanced** tab. Figure 29-59 provides details.

Figure 29-59 Notification Section - Advanced Tab



Instructions for configuring the following subsections of the **Advanced** tab of the **Notification** section are listed in Table 29-16.

Table 29-16 Human Task Editor — Advanced Tab of Notification Section

For This Subsection...	See...
Reminders	How to Set Up Reminders
Encoding	How to Change the Character Set Encoding
Make notifications secure (exclude details)	How to Secure Notifications to Exclude Details
Show worklist URL in notifications	How to Display the Oracle BPM Worklist URL in Notifications
Make notifications actionable	How to Make Email Messages Actionable

Table 29-16 (Cont.) Human Task Editor — Advanced Tab of Notification Section

For This Subsection...	See...
Send task attachments with email notifications	How to Send Task Attachments with Email Notifications
Group notification configuration	How to Send Email Notifications to Groups and Application Roles
Notification header attributes	How to Customize Notification Headers

How to Notify Recipients of Changes to Task Status

Three default status types display in the **Task Status** column: **Assign**, **Complete**, and **Error**. You can select other status types for which to receive notification messages.

To notify recipients of changes to task status:

1. In the **Notification** section, click the **General** tab.
2. In the **Task Status** column, click a type to display the complete list of task types:
   - **Alerted**

     When a task is in an alerted state, you can notify recipients. However, none of the notification recipients (assignees, approvers, owner, initiator, or reviewer) can move the task from an alerted state to an error state; they only receive an FYI notification of the alerted state. The owner can reassign, withdraw, delete, or purge the task, or ask the error assignee to move the task to an error state if the error cannot be resolved. Only the error assignee can move a task from an alerted state to an error state.

     You configure the error assignee on the **Assignment** tab of the Configure Assignment dialog box under the **Task will go from starting to final participant** icon in the **Assignment** section. For more information, see How to Configure the Error Assignee and Reviewers.

   - **Assign**

     When the task is assigned to users or a group. This captures the following actions:

     – Task is assigned to a user

     – Task is assigned to a new user in a serial workflow

     – Task is renewed

     – Task is delegated

     – Task is reassigned

     – Task is escalated

     – Information for a task is submitted

   - **Complete**
   - **Error**

- **Expire**
- **Request Info**
- **Resume**
- **Suspend**
- **Update**
  - Task payload is updated
  - Task is updated
  - Comments are added
  - Attachments are added and updated
- **Update Outcome**
- **Withdraw**
- **All Other Actions**
  - Any action not covered in the above task types. This includes acquiring a task.

3. Select a task status type.

   Notifications can be sent to users involved in the task in various capacities. This includes when the task is assigned to a group, each user in the group is sent a notification if there is no notification endpoint available for the group.

4. In the **Recipient** column, click an entry to display a list of possible recipients for the notification message:

   - **Assignees**

     The users or groups to whom the task is currently assigned.

   - **Initiator**

     The user who created the task.

   - **Approvers**

     The users who have acted on the task up to this point. This applies in a serial participant type in which multiple users have approved the task and a notification must be sent to all of them.

   - **Owner**

     The task owner

   - **Reviewer**

     The user who can add comments and attachments to a task.

   For more information, see the section on notifications in *Developing SOA Applications with Oracle SOA Suite*.

How to Edit the Notification Message

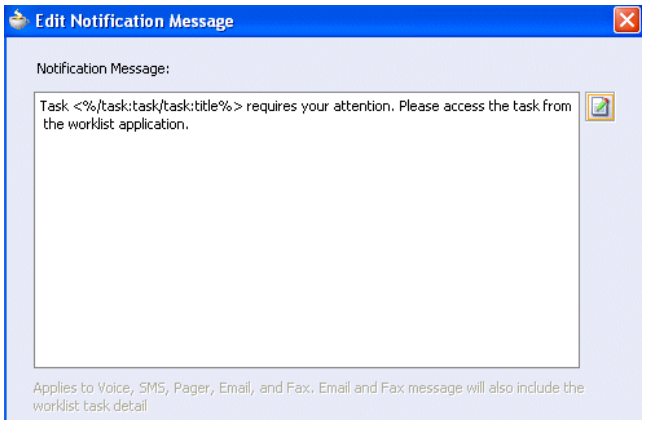
A default notification message is available for delivery to the selected recipient. If you want, you can modify the default message text.

To edit the notification message:

1. In the **Notification** section, click the **General** tab.
2. In the **Notification Header** column, click the **Edit** icon to modify the default notification message.

   The Edit Notification Message dialog box shown in Figure 29-60 appears.

   Figure 29-60 Edit Notification Message Dialog

   This message applies to all the supported notification channels: email, voice, instant messaging, and SMS. Email messages can also include the worklist task detail defined in this message. The channel by which the message is delivered is based upon the notification preferences you specify.
3. Modify the message wording as necessary.
4. Click **OK** to return to the Human Task Editor.

For more information about notification preference details, see the section on notifications in *Developing SOA Applications with Oracle SOA Suite*.

How to Set Up Reminders

You can send task reminders, which can be based on the time the task was assigned to a user or the expiration time of a task. The number of reminders and the interval between the reminders can also be configured.

To set up reminders:

1. In the **Notification** section, click the **Advanced** tab.
2. From the list, select the number of reminders to send.
3. If you selected to remind the assignee one, two, or three times, select the interval between reminders, and whether to send the reminder before or after the assignment.

For more information, see the section on reminders in *Developing SOA Applications with Oracle SOA Suite*.

How to Change the Character Set Encoding

Unicode is a universally-encoded character set that enables information from any language to be stored using a single character set. Unicode provides a unique code value for every character, regardless of the platform, program, or language. You can use the default setting of UTF-8 or you can specify a character set with a Java class.

To change the character set encoding

1. In the **Notification** section, click the **Advanced** tab.
2. From the **Encoding** list, select **Specify by Java Class**.
3. Enter the Java class to use.

How to Secure Notifications to Exclude Details

To secure notifications, make messages actionable, and send attachments:

1. In the **Notification** section, click the **Advanced** tab.
2. Select **Make notifications secure (exclude details)**.

   If selected, a default notification message is used. There are no HTML worklist task details, attachments, or actionable links in the email. Only the task number is in the message.

   For more information, see the section on notifications in *Developing SOA Applications with Oracle SOA Suite*.

How to Display the Oracle BPM Worklist URL in Notifications

You can configure whether to display the Oracle BPM Worklist URL in email notification messages.

To display the Oracle BPM Worklist URL in notifications:

1. In the **Notification** section, click the **Advanced** tab.
2. Select the **Show worklist URL in notifications** check box to display the Oracle BPM Worklist URL in email notification messages. If this check box is not selected, the URL is not displayed.

How to Make Email Messages Actionable

To make email messages actionable:

1. In the **Notification** section, click the **Advanced** tab.
2. Select **Make notification actionable**. This action enables you to perform task actions through email.

Note:

FYI tasks are not actionable and cannot be acknowledged from email messages.

For more information about additional configuration details, see the section on actionable messages in *Developing SOA Applications with Oracle SOA Suite*.

For more information about configuring outbound and inbound emails, see *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

How to Send Task Attachments with Email Notifications

You can send task attachments with email notifications.

To send task attachments with email notifications:

1. In the **Notification** section, click the **Advanced** tab.
2. Select **Send task attachments with email notifications**.

How to Send Email Notifications to Groups and Application Roles

You can send email notifications to groups and application roles to which tasks are assigned.

To send email notifications to groups and application roles:

1. In the **Notification** section, click the **Advanced** tab.
2. From the **Group notification configuration** list, select one of the following options.
   - **Send individual emails**

     Each user in the group or application role receives an individual email notification. This is the default selection.

     In addition, the **Use separate task forms based on locale** check box is automatically selected.

     – When selected, this sends individual emails with a separate task form based on the language locale.

     – When not selected, this sends individual emails and reuses (shares) the task form.

   - **Send one email containing all user addresses**

     A shared notification email is generated once for a user locale in a group or application role, thereby saving time in notification email content generation. The email is sent to all users in the group or application role.

Note:

– Since all (or a subset of) users receive the same email, the users in the group or application role are expected to have the same privilege. This ensures that the user does not see task details to which they are not entitled.

– When sending one email to all users, the maximum number of characters allowed in the address field is 2000. If the limit is exceeded, email is sent to only those user addresses contained within the maximum limit.

How to Customize Notification Headers

Custom notification headers are used to specify name and value pairs to identify key fields within the notification. These entries can be used by users to define delivery preferences for their notifications. For example, you can set **Name** to **ApprovalType** and **value** to **Expense** or **Name** to **Priority** and **value** to **High**.Users can then specify delivery preferences in Oracle BPM Worklist. These preferences can be based on the contents of the notification.

The rule-based notification service is *only* used to identify the preferred notification channel to use. The address for the preferred channel is still obtained from the identity service.

To customize notification headers:

1. In the **Notification** section, click the **Advanced** tab.
2. Expand **Notification Header Attributes**.
3. Add name and pair value parameters by name or XPath expression.

   For more information about preferences, see the following sections:

   - the sections on sending inbound and outbound attachments, and creating custom notification headers in *Developing SOA Applications with Oracle SOA Suite*.
   - *Developing Applications with Oracle User Messaging Service*

Specifying Access Policies and Task Actions on Task Content

You can specify access rules on task content and actions to perform on that content.

This includes specifying access policies on task content and how to specify a workflow digital signature policy.

How to Specify Access Policies on Task Content

You can specify access rules that determine the parts of a task that participants can view and update. Access rules are enforced by the workflow service by applying rules on the task object during the retrieval and update of the task.

Note:

Task content access rules and task actions access rules exist independently of one another.

Introduction to Access Rules

Access rules are computed based on the following details:

- Any attribute configured with access rules declines any permissions for roles not configured against it. For example, assume you configure the payload to be read by assignees. This action enables *only* assignees and nobody else to have read permissions. No one, including assignees, has write permissions.
- Any attribute not configured with access rules has *all* permissions.
- If any payload message attribute is configured with access rules, any configurations for the payload itself are ignored due to potential conflicts. In this case, the returned map by the API does not contain any entry for the payload. Write permissions automatically provide read permissions.
- If only a subset of message attributes is configured with access rules, all message attributes not involved have all permissions.
- Only comments and attachments have add permissions.
- Write permissions on certain attributes are meaningless. For example, write permissions on history do not grant or decline any privileges on history.
- The following date attributes are configured as one in the Human Task Editor. The map returned by `TaskMetadataService.getVisibilityRules()` contains one key for each. Similarly, if the participant does not have read permissions on `DATES`, the task does not contain any of the following task attributes:
  - `START_DATE`
  - `END_DATE`
  - `ASSIGNED_DATE`
  - `SYSTEM_END_DATE`
  - `CREATED_DATE`
  - `EXPIRATION_DATE`
  - `ALL_UPDATED_DATE`
- The following assignee attributes are configured as one in the Human Task Editor. The map returned by `TaskMetadataService.getVisibilityRules()` contains one key for each of the following. Similarly, if the participant does not have read permissions on `ASSIGNEES`, the task does not contain any of the following task attributes:
  - `ASSIGNEES`
  - `ASSIGNEE_USERS`
  - `ASSIGNEE_GROUPS`

  - ACQUIRED_BY

- Mapped attributes do not have individual representation in the map returned by `TaskMetadataService.getVisibilityRules()`.
- All message attributes in the map returned by `TaskMetadataService.getVisibilityRules()` are prefixed by `ITaskMetadataService.TASK_VISIBILITY_ATTRIBUTE_PAYLOAD_MESSAGE_ATTR_PREFIX (PAYLOAD)`.

An application can also create pages to display or not display task attributes based on the access rules. This can be achieved by retrieving a participant's access rules by calling the API on `oracle.bpel.services.workflow.metadata.ITaskMetadataService`. Example 29-1 provides details.

For more information about this method, see *Oracle Fusion Middleware Workflow Services Java API Reference for Oracle SOA Suite*.

Example 29-1 API Call

```
public Map<String, IPrivilege> getTaskVisibilityRules(IWorkflowContext context,
                                    String taskId)
   throws TaskMetadataServiceException;
```

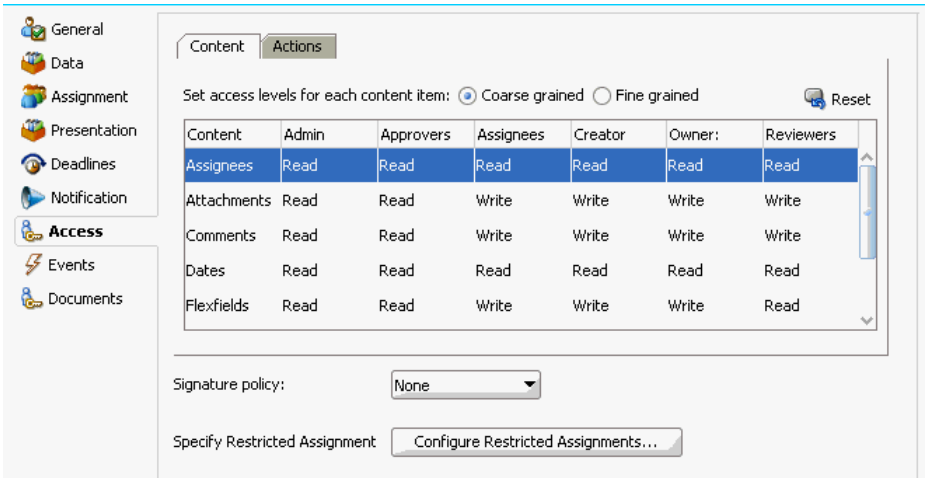
Specifying User Privileges for Acting on Task Content

You can specify the privileges that specific users (such as the task creator or owner) have for acting on specific task content (such as a payload).

To specify user privileges for acting on task content:

1. Click the **Access** tab.
2. Click the **Content** tab.
3. Select the task content for which to specify access privileges, as shown in Figure 29-61.

Figure 29-61 Configure Task Content Access



4. Assign privileges (read, write, or no access) to users to act upon task content. A user cannot be assigned a privilege above their highest level. For example, an

ADMIN user cannot be assigned write access on the **PAYLOAD** task content. Table 29-17 shows the maximum privilege each user has on task content.

Table 29-17 Highest Privilege Levels for Users of Task Content

Task Content	Individual with Read Access	Individual with Write Access
Assignees	Admin, Approvers, Assignees, Creator, Owner, Reviewers	--
Attachments	Admin, Approvers	Assignees, Creator, Owner, Reviewers
Comments	Admin, Approvers	Assignees, Creator, Owner, Reviewers
Dates	Admin, Approvers, Assignees, Creator, Owner, Reviewers	--
Flexfields	Admin, Approvers, Reviewers	Assignees, Creator, Owner
History	Admin, Approvers, Assignees, Creator, Owner, Reviewers	--
Payload	Admin, Approvers, Reviewers	Assignees, Creator, Owner
Reviewers	Admin, Approvers, Assignees, Creator, Owner, Reviewers	--
Payload elements	Inherited from payload	Inherited from payload

For example, if you accept the default setting of **ASSIGNEES**, **CREATOR**, and **OWNER** with write access, **ADMIN**, **APPROVERS**, and **REVIEWERS** with read access, and **PUBLIC** with no access to the **PAYLOAD** task content, the dialog box appears as shown in Figure 29-61.

5. Select the method for displaying task content in this dialog box. Choosing the currently unselected option causes all settings to reset to their default values.

   - **Coarse grained** (default)

     Displays the task content as a whole (for example, displays only one payload or reviewer).

   - **Fine grained**

     Displays the content as individual elements (for example, displays all payloads (such as **p1**, **p2**, and **p3**) and all reviewers assigned to this task (such as **jstein**, **wfaulk**, and **cdickens**).

     Note:

     Access rules are always applied on top of what the system permits, depending on who is performing the action and the current state of the task.

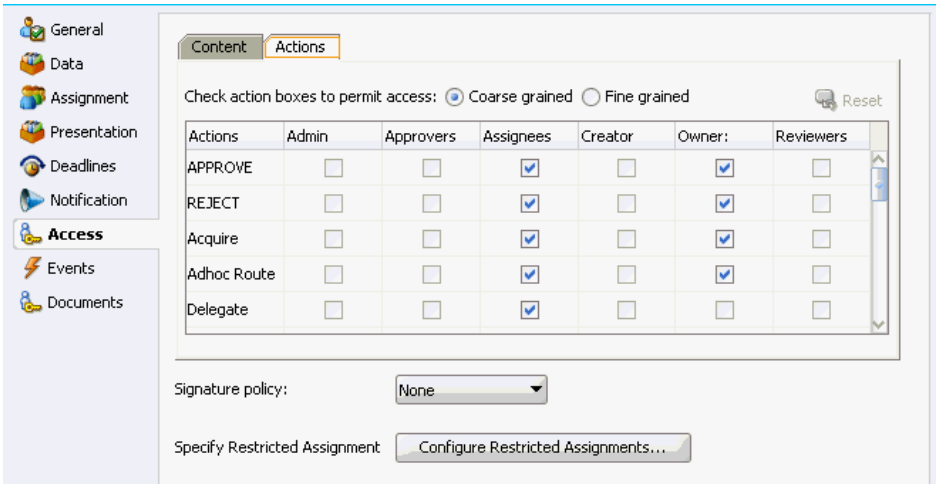
Specifying Actions for Acting Upon Tasks

You can specify the actions (either access or no access) that specific users (such as the task creator or owner) have for acting on the task content (such as a payload) that you specified in the Configure Task Content Access dialog box.

To specify actions for acting upon tasks:

1. Click the **Access** tab.
2. Click the **Actions** tab.
3. Select the task action for which to specify users, as shown in Figure 29-62.

Figure 29-62 Selection of Add Action Access Rule



4. Select if participants can or cannot perform the selected actions.
5. Select the method for displaying task actions in this dialog box. Choosing the currently unselected option causes all settings to reset to their default values.

   - **Coarse grained** (default)

     Displays the task actions as a whole (for example, displays only one approval or rejection).

   - **Fine grained**

     Displays the content actions as individual elements. (for example, displays all approvals or rejections).

Creating and Implementing Digital Certificates

This section discusses how to create and implement digital certificates.

Note: Signing tasks using digital signature is not supported on Google Chrome and Apple Safari browsers.

Tasks for creating and implementing digital certificates

- How to Create a Digital Certificate Authority
- How to Create Digital User Certificates
- How to Generate Digital Certificate Revocation List
- How to Specify a Certificate Authority
- How to Specify a Workflow Digital Signature Policy

How to Create a Digital Certificate Authority

You must create a digital certificate authority that issues individual user certificates.

Create a digital certificate authority that issues individual user certificates. To create a digital certificate authority, login as user root, open a Linux terminal window, and then enter the following commands sequentially.

1. Create a Certificate Authority's Certificate Signing Request (CA's CSR) file.

   ```
   openssl req -passout pass:<password> -subj "/C=<Country Code>/ST=<State Code>/
   L=<Company Location>/O=<Company Name>/OU=<IT Security Division>/CN=<Name of
   security Authority>/emailAddress=<Email ID of the certificate Authority>" -new >
   <Certificate Authority Name>.cert.csr
   ```

   For example, `openssl req -passout pass:welcome1 -subj "/C=AU/ ST=QLD/L=Brisbane/O=Ozayr Syed Security Incorporated/OU=Oz Security Division/CN=Ozayr Syed Certificate Authority/ emailAddress=ozayr.syed@oz.com" -new > OzayrSyedCA.cert.csr`

2. Create a key file to store the private key.

   ```
   openssl rsa -passin pass:<password> -in privkey.pem -out <Key Name>.ca.key
   ```

   For example, `openssl rsa -passin pass:welcome1 -in privkey.pem -out OzCertificateAuthority.ca.key`

3. Create an X.509 digital certificate from the above-mentioned certificate request.

   ```
   openssl x509 -in < Certificate Authority Name>.cert.csr -out
   <CertificateAuthority>.ca.cert -req -signkey <Key Name>.ca.key -days 365
   ```

   For example, `openssl x509 -in OzayrSyedCA.cert.csr -out OzCertificateAuthority.ca.cert -req -signkey OzCertificateAuthority.ca.key -days 365`

4. Create a PKCS#12-encoded file with the certificate and private key.

   ```
   openssl pkcs12 -passout pass:<password> -export -nokeys -cacerts -
   <CertificateAuthority>.ca.cert -out <CertificateAuthority>.ca.cert.p12 -inkey
   <Key Name>.ca.key
   ```

   For example, `openssl pkcs12 -passout pass:welcome1 -export -nokeys -cacerts -in OzCertificateAuthority.ca.cert -out OzCertificateAuthority.ca.cert.p12 -inkey OzCertificateAuthority.ca.key`

Certificate Authority certificate is created (OzCertificateAuthority.ca.cert). Use the Certificate Authority to create digital certificates for users.

How to Create Digital User Certificates

Use the Certificate Authority to create digital certificates for users.

Create individual user certificates. To create a digital user certificate, login as user root, open a Linux terminal window, and then enter the following commands sequentially.

1. Create a Certificate Signing Request (CSR) file for the user.

   ```
   openssl req -passout pass:<password> -subj "/C=<Country Code>/ST=<State Code>/
   L=<Company Location>/O=<Company Name>/OU=<IT Security Division>/CN=<approver>/
   emailAddress=<Email ID of the approver>" -new > <approver ID>.cert.csr
   ```

   For example, `openssl req -passout pass:welcome1 -subj "/C=AU/ ST=QLD/L=Brisbane/O=Oz Motor Corporation/OU=ICT Approvals/ CN=approver/emailAddress=approver@oz.com" -new > approver.cert.csr`

2. Create a private key file for the user.

   ```
   openssl rsa -passin pass:<password> -in privkey.pem -out <approver ID>.key
   ```

   For example, `openssl rsa -passin pass:welcome1 -in privkey.pem -out approver.key`

3. Create a X.509 certificate for the user.

   ```
   openssl x509 -req -in <approver ID>.cert.csr -out approver.cert -signkey
   approver.key -CA <certificate authority>.ca.cert -CAkey <Key ID>.ca.key -
   CAcreateserial -days 365
   ```

   For example, `openssl x509 -req -in approver.cert.csr -out approver.cert -signkey approver.key -CA OzCertificateAuthority.ca.cert -CAkey OzCertificateAuthority.ca.key -CAcreateserial -days 365`

4. Create a PKCS#12-encoded file.

   ```
   openssl pkcs12 -passout pass:<password> -export -in approver.cert -out
   approver.cert.p12 -inkey approver.key
   ```

How to Generate Digital Certificate Revocation List

To generate digital certificate revocation list:

1. Run the following command.

   ```
   run, openssl ca -gencrl -out <certificate authority name>.crl.pem.crl
   ```

   For example, `openssl ca -gencrl -out OzCertificateAuthority.crl.pem.crl`

2. Correct errors, if any. (Optional step)

   If the following error occurs.

```
Using configuration from /etc/pki/tls/openssl.cnf
Error opening CA private key /etc/pki/CA/private/cakey.pem
139903336728392:error:02001002:system library:fopen:No such file or
directory:bss_file.c:398:fopen('/etc/pki/CA/private/cakey.pem','r')
139903336728392:error:20074002:BIO routines:FILE_CTRL:system lib:bss_file.c:400:
unable to load CA private key
```

Fix for the error by inserting the following content in `crlnumber`.

```
cp <Key ID>.ca.key  /etc/pki/CA/private/cakey.pem
cp <certificate authority>.ca.cert /etc/pki/CA/cacert.pem
touch /etc/pki/CA/index.txt
touch /etc/pki/CA/crlnumber
```

Note: Navigate to `/etc/pki/CA/crlnumber` and insert **01** in first line and press the enter key (for a separate line), and only then edit the file.

3. Convert PEM to DER format by running the following commands..

```
openssl crl -inform PEM -in <certificate authority name>.crl.pem.crl -outform DER
-out <certificate authority name>.crl.der.crl
openssl ca -gencrl -out <certificate authority name>.crl.pem.crl
```

(Optional) Enter the result of the procedure here.

How to Specify a Certificate Authority

To use digital signatures, you must specify CAs you consider trustworthy in the System MBean Browser in Oracle Enterprise Manager Fusion Middleware Control. Only certificates issued from such CAs are considered valid by human workflow. To specify a certificate authority:

1. From the **SOA Infrastructure** menu, select **Administration** > **System MBean Browser**.
2. Select **Application Defined MBeans** > **oracle.as.soainfra.config** > **Server:** *server_name* > **WorkflowConfig** > **human.workflow**.
3. Click the **Operations** tab.
4. Click **AddTrustedCA**.
5. In the **Value** fields for **CaName** and **CaURL**, specify appropriate values.
6. Click **Invoke**.
7. Click **Return**.

   You must validate these values before using them.

How to Specify a Workflow Digital Signature Policy

Digital signatures provide a mechanism for the nonrepudiation of digitally-signed human tasks. This ability to mandate that a participant acting on a task signs the details and their action before the task is updated ensures that they cannot repudiate it later.

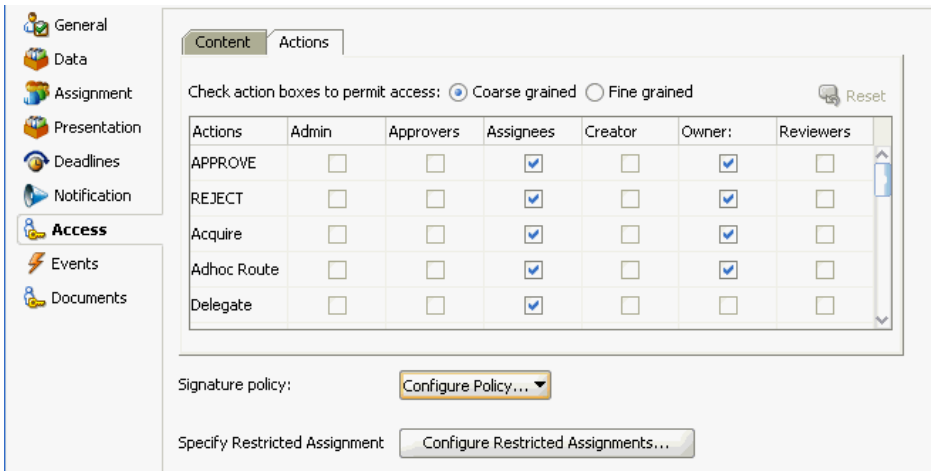
Note:

If digital signatures are enabled for a task, actionable emails are not sent during runtime. This is the case even if actionable emails are enabled during design time.

To specify a workflow digital signature policy:

1. Click the **Access** tab.
2. From the **Signature Policy** list, select **Configure Policy**, as shown in Figure 29-63.

Figure 29-63 Digital Signatures



3. Specify the signature policy for task participants to use:

   - **No signature required**

     Participants can send and act upon tasks without providing a signature. This is the default policy.

   - **Password required**

     Participants specify a signature before sending tasks to the next participant. Participants must reenter their password while acting on a task. The password is used to generate the digital signature. A digital signature authenticates the identity of the message sender or document signer. This ensures that the original content of the sent message is unchanged.

   - **Digital certificate required**

     Participants must possess a digital certificate for the nonrepudiation of digitally-signed human tasks. A digital certificate establishes the participant's credentials. It is issued by a certification authority (CA). It contains the following:

     – Your name

     – A serial number

     – Expiration dates

     – A copy of the certificate holder's public key (used for encrypting messages and digital signatures)

– Digital signature of the certificate-issuing authority so that message authenticity can be established

The CA names and CA CRL and URLs of the issuing authorities must be configured separately.

4. Click **OK**.

For more information, see the section on the evidence store in *Developing SOA Applications with Oracle SOA Suite*.

Specifying Restrictions on Task Assignments

You can restrict the users to which a task can be reassigned or routed by using a callback class.

The user community seeded in a typical LDAP directory can represent the whole company or division. However, it may be necessary at times to limit the potential list of users to associate with a task based on the scope or importance of the task or associated data. For example, in a large company with thousands of users, only a few people have the ability to approve and create purchase orders. Specifically for such tasks, the users that can be chosen for ad hoc routing and reassignment should not be the whole company. Instead, only a few users who are relevant or have the right privilege should be chosen. This can be achieved by the restricted assignment functionality. This is implemented as a callback class that can implement the logic to choose the right set of users dynamically based on the task object that is passed containing the instance data.

Note:

Certain functions, such as restricted task reassignment, are available only when a single task is selected. If multiple tasks that use restricted reassignment are selected, then the restricted reassignment algorithm is not invoked. In that case, the complete list of users gets returned as though restricted reassignment had not been specified.

How to Specify Restrictions on Task Assignments

To specify restrictions on task assignments:

1. In the **Access** section, click **Configure Restricted Assignments**.

   The Configure Restricted Assignment dialog box appears.

2. Enter the class name. The class must implement the `oracle.bpel.services.workflow.task.IRestrictedAssignmentCallback` interface.

3. Click the **Add** icon to add name and value pairs for the property map passed to invoke the callback.

4. Click **OK**.

Specifying Java or Business Event Callbacks

You can specify Java or business event callbacks.

Note:

If you implemented a callback, then the user callback implementation overrides any other form of restricted assignment. When you perform a search, the result only shows the users that the user callback returns.

How to Specify Callback Classes on Task Status

You can register callbacks for the workflow service to call when a particular stage is reached during the lifecycle of a task. Two types of callbacks are supported:

- Java callbacks: The callback class must implement the interface `oracle.bpel.services.workflow.task.IRoutingSlipCallback`. Make the callback class available in the class path of the server.
- Business event callbacks: You can have business events raised when the state of a human task changes. You do not need to develop and register a Java class. The caller implements the callback using an Oracle Mediator service component to subscribe to the applicable business event to be informed of the current state of an approval transaction.

To specify callback classes on task status:

1. Click the **Events** tab.

   The following state change callbacks are available for selection:

   - **OnAssigned**

     Select if the callback class must be called on any assignment change, including standard routing, reassignment, delegation, escalation, and so on. If a callback is required when a task has an outcome update (that is, one of the approvers in a chain approves or rejects the task), this option must be selected.

   - **OnUpdated**

     Select if the callback class must be called on any update (including payload, comments, attachments, priority, and so on).

   - **OnCompleted**

     Select if the callback class must finally be called when the task is completed and control is about to be passed to the initiator (such as the BPEL process initiating the task).

   - **OnStageCompleted**

     Select if the callback class must be called to enable business event callbacks in a human workflow task. When the event is raised, it contains the name of the completed stage, the outcome for the completed stage, and a snapshot of the task when the callback is invoked.

   - **OnSubtaskUpdated**

Select if the callback class must be called on any update (including payload, comments, attachments, priority, and so on) on a subtask (one of the tasks in a parallel-and-parallel scenario).

If your Oracle JDeveloper installation is updated to include both the BPEL and BPM extensions, then the following content callbacks are also available for selection:

- **Comments Callback**

  Select if the callback class must be called to store the comments in a schema other than the `WFCOMMENTS` column. The callback class must implement the `oracle.bpel.services.workflow.callback.NotesStore` interface.

- **Attachment Call Back**

  Select if the callback class must be called to store the attachments in a schema other than the `WFATTACHMENT` table in the soa-infra schema. The callback class must implement the `oracle.bpel.services.workflow.callback.AttachmentStore` interface.

- **Validation Callback**

  Select if the callback class must be called to validate either the task or payload before updating, approving, and so on. The callback class must implement the `oracle.bpel.services.workflow.task.ITaskValidationCallback` interface.

2. See the following section based on the type of callback to perform.

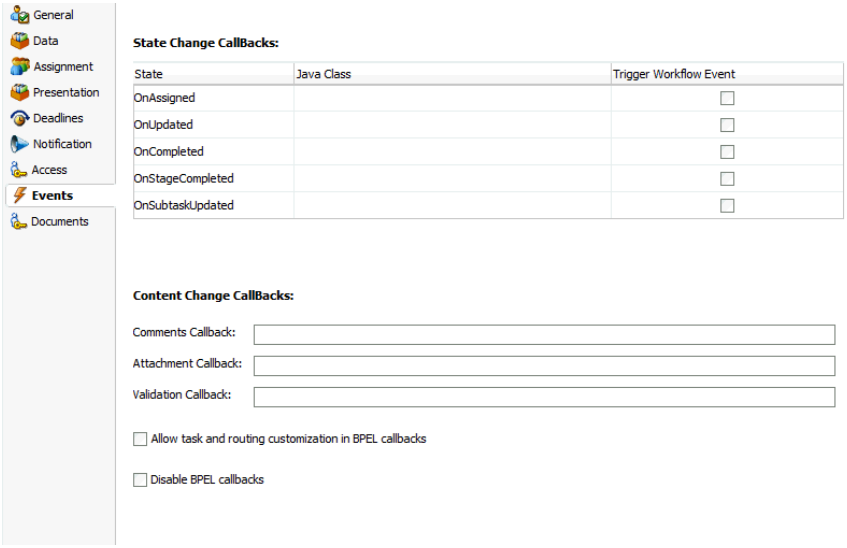
   - Specifying Java Callbacks
   - Specifying Business Event Callbacks

Specifying Java Callbacks

To specify Java callbacks:

1. In the **State** column of the **Events** section, select a task state.
2. In the **Java Class** column, click the empty field to enter a value. This value is the complete class name of the Java class that implements `oracle.bpel.services.workflow.task.IRoutingSlipCallback`. Figure 29-64 provides details.

Figure 29-64 CallBack Details Dialog with Java Selected



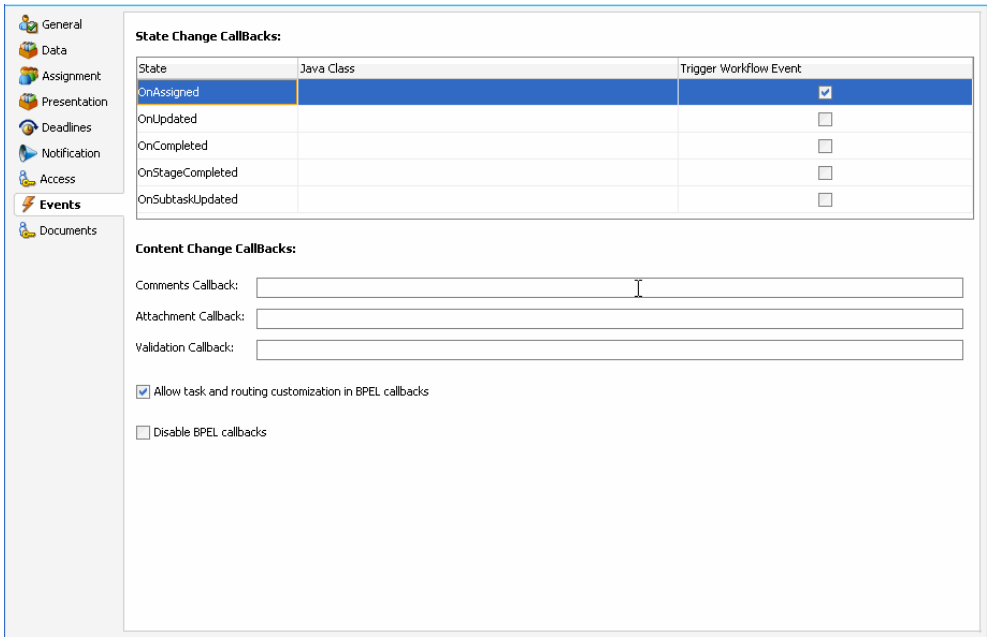
3. Click **OK**.

Specifying Business Event Callbacks

To specify business event callbacks:

1. In the **State** column of the **Events** section, select a task state.
2. Leave the **Java Class** field empty.
3. Select the **Trigger Workflow Event** check box. This action disables the **Java Class** column, as shown in Figure 29-65. Each callback, such as **OnAssigned**, corresponds to a business event point. When a business event is fired, the event details contain the task object and a set of properties that are populated based on the context of the event being fired.

Figure 29-65 CallBack Details Dialog with Business Events Selected



A pre-seeded, static event definition language (EDL) file (*JDev_Home* `\jdeveloper\integration\seed\soa\shared\workflow\HumanTaskEvent.edl`) provides the list of available business events to which to subscribe. These business events correspond to the callbacks you select in the Callback Details dialog box. You must now create an Oracle Mediator service component in which you reference the EDL file and subscribe to the appropriate business event.

Note:

A file-based MDS connection is required so that the EDL file can be located. The location for the file-based MDS is *JDev_Home*`\jdeveloper\integration\seed`.

---

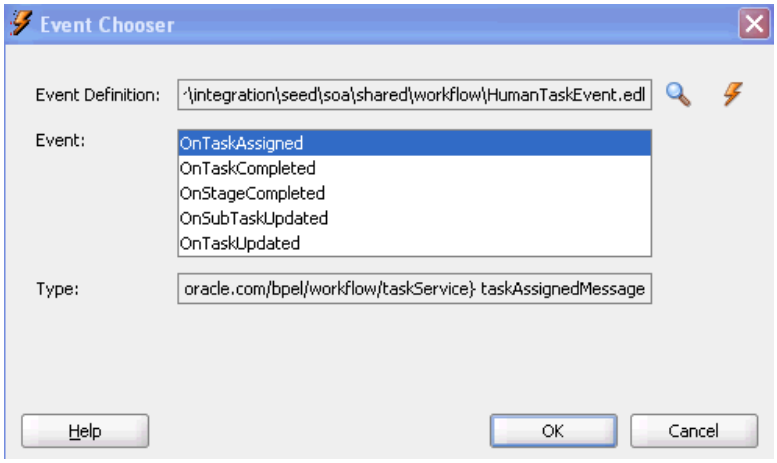
4. Create an Oracle Mediator service component in the same or a different SOA composite application that can subscribe to the event.
5. In the **Template** list during Oracle Mediator creation, select **Subscribe to Events**.
6. Click the **Add** icon to subscribe to a new event.
7. To the right of the **Event Definition** field, click the **Browse** icon to select the EDL file.

   The SOA Resource Browser dialog box appears.
8. Select the previously created file-based MDS connection.
9. From the list at the top, select **Resource Palette**.
10. Select **SOA** > **Shared** > **Workflow** > **HumanTaskEvent.edl**.
11. Click **OK**.

The Event Chooser is now populated with EDL file business events available for selection.

12. In the **Event** field, select the event to which to subscribe. Figure 29-66 provides details.

Figure 29-66 Event Callbacks



You can have multiple human tasks available for subscribing to the event. For example, assume you performed the following:

- Configured a human task named TaskA to subscribe to the event (for example, **OnAssigned**)
- Configured a human task named TaskB to subscribe to the same event

To distinguish between events for TaskA and TaskB and ensure that an event is processed only by the intended Oracle Mediator, you can add a static routing filter:

```
xpath20:compare(med:getComponentName(), 'TaskA')
```

This only invokes this routing when the sending component is TaskA.

13. If the EDL file was *not* selected from the file-based MDS connection, accept to import the dependent XSD files when prompted, and click **OK**. If the EDL file was selected from the file-based MDS connection, you are not prompted.

    The Oracle Mediator service component is now populated with the business event to which to subscribe. You can also subscribe to other business events defined in the same EDL file now or at a later time.

See the following documentation for additional details about business events and callbacks:

- *Developing SOA Applications with Oracle SOA Suite*.
- Sample workflow-116-WorkflowEventCallback, which is available with the Oracle SOA Suite samples.

How to Specify Task and Routing Customizations in BPEL Callbacks

In general, the BPEL process calls into the workflow component to assign tasks to users. When the workflow is complete, the human workflow service calls back into the BPEL process. However, if you want fine-grained callbacks (for example,

`onTaskUpdate` or `onTaskEscalated`) to be sent to the BPEL process, you can use the **Allow task and routing customization in BPEL callbacks** option.

Make sure to manually refresh the BPEL diagram for this callback setting.

To specify task and routing customizations in BPEL callbacks:

1. In the **Events** section, select the **Allow task and routing customization in BPEL callbacks** check box.
2. Return to Oracle BPEL Designer.
3. Open the task activity dialog box.
4. Click **OK**.

This creates the while, pick, and onMessage branch of a pick activity for BPEL callback customizations inside the task scope activity.

For more information about specifying task and routing customizations, see *Developing SOA Applications with Oracle SOA Suite*.

How to Disable BPEL Callbacks

A user talk activity (in Oracle BPEL Designer) has an invoke activity followed by a receive or pick activity. Deselecting the **Disable BPEL callbacks** check box enables you to invoke the task service without waiting for a reply.

To disable BPEL callbacks:

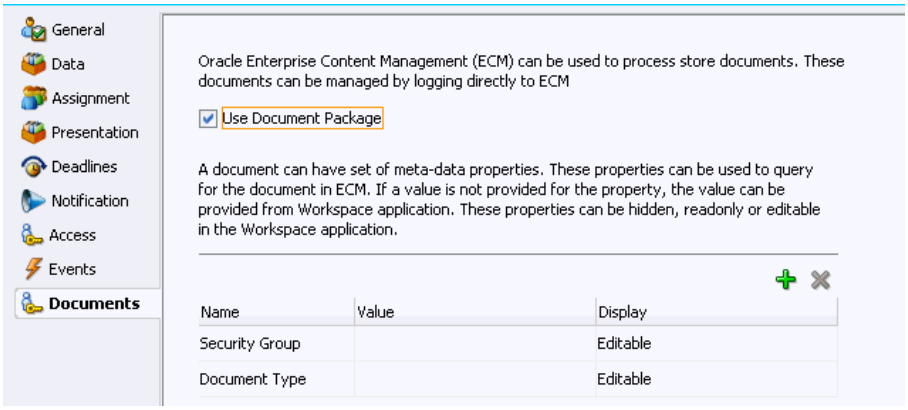
1. In the **Events** section, deselect the **Disable BPEL callbacks** check box.
2. Click **OK**.

Storing Documents in Oracle Enterprise Content Management

Oracle Enterprise Content Management can be used to process store documents.

Figure 29-67 shows the **Documents** section of the Human Task Editor.

Figure 29-67 Human Task Editor — Documents Section



How to Configure Oracle UCM Repository to Store Task Attachments

You can configure Human Tasks to store attachments in the UCM repository. These attachments may contain one or more metadata properties. You can assign values to these properties or configure them to allow the user to provide the value.

Note: When a file is attached from the Process Tracking page, it goes to the BPM repository and not to the UCM repository. Hence, even if the UCM repository is unavailable, the Process Tracking page allows you to upload files. However, when the UCM repository is configured and Human Tasks are designed to upload attachment to the UCM repository, the Tasks page uploads files to the UCM repository. However, the Process Tracking page always uploads the files to BPM repository.

To configure Oracle UCM Repository for task attachments:

1. In the **Project Navigator** tree, expand the Business Catalog node.
2. Expand the **Human Tasks** node.
3. Double-click the Human Task you want to configure.

   The Human Task Editor appears.
4. Click the **Documents** tab.
5. Select **Use Document Package**.

   A section to configure metadata properties appears. The table already contains the mandatory standard metadata: Security Group and Document Type.
6. Optionally add new standard or custom metadata properties:
   a. Click the **Add** button.
   b. Click the **Name** column to select a standard property from the list or to enter a custom name.
   c. Click the **Value** column to assign a value to the property. Select **By Name** to provide the text to assign the value to the property, or **By Expression** to provide an expression.
   d. Click the Display column to select a display mode:

      Editable: the user can provide a value in the task form when uploading the attachment.

      Hidden: the value does not appear in the task form

      Read-Only: the value appears in the task form but the user cannot modify it

---

Note:

Custom metadata does not appear in the task form, so you must map the value to a task payload or provide a static value.

30

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.

- Introduction to Guided Business Processes
- Guided Business Process Use Cases
- Standards and Guidelines for Working with Guided Business Processes
- The Typical Flow of Developing a Guided Business Process
- Introduction to Developing a Guided Business Process
- Developing a BPMN Guided Business Process
- Configuring Activity Guide Properties
- Deploying an Guided Business Process to Oracle WebLogic Server
- Testing Guided Business Processes

Introduction to Guided Business Processes

Guided Business Processes 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 30-1 An Activity Guide

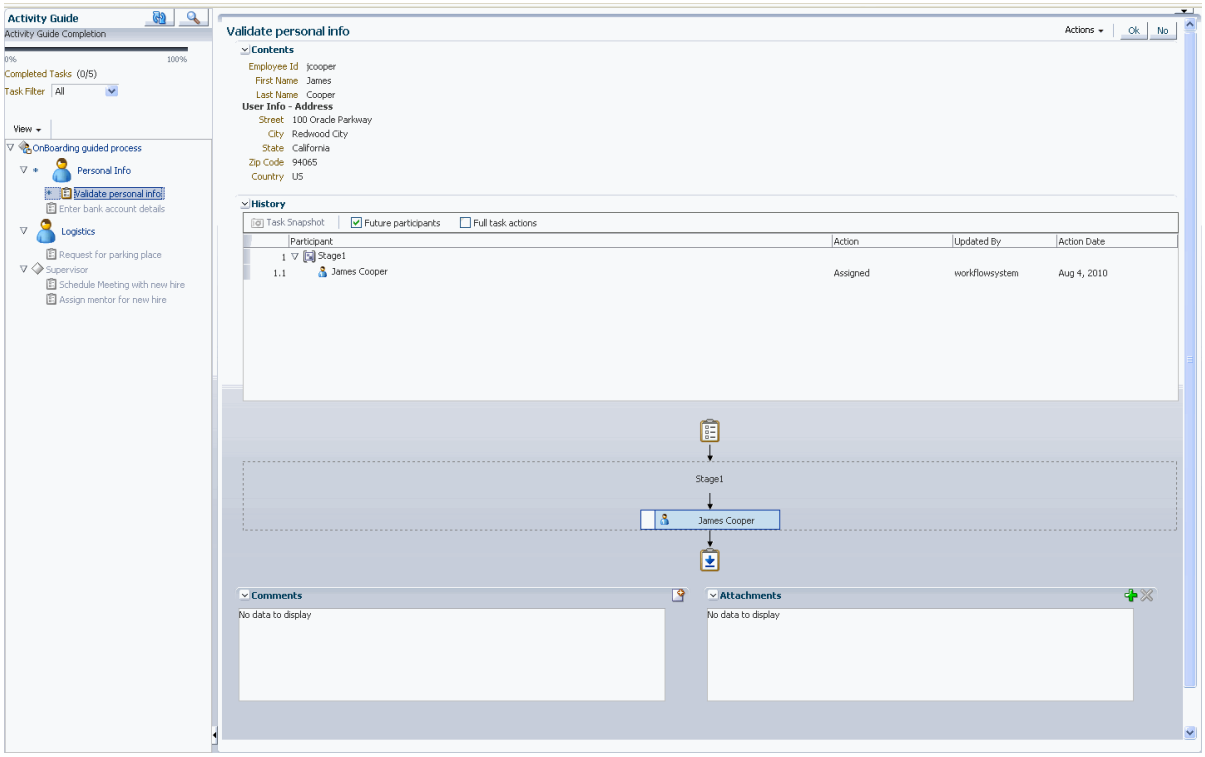
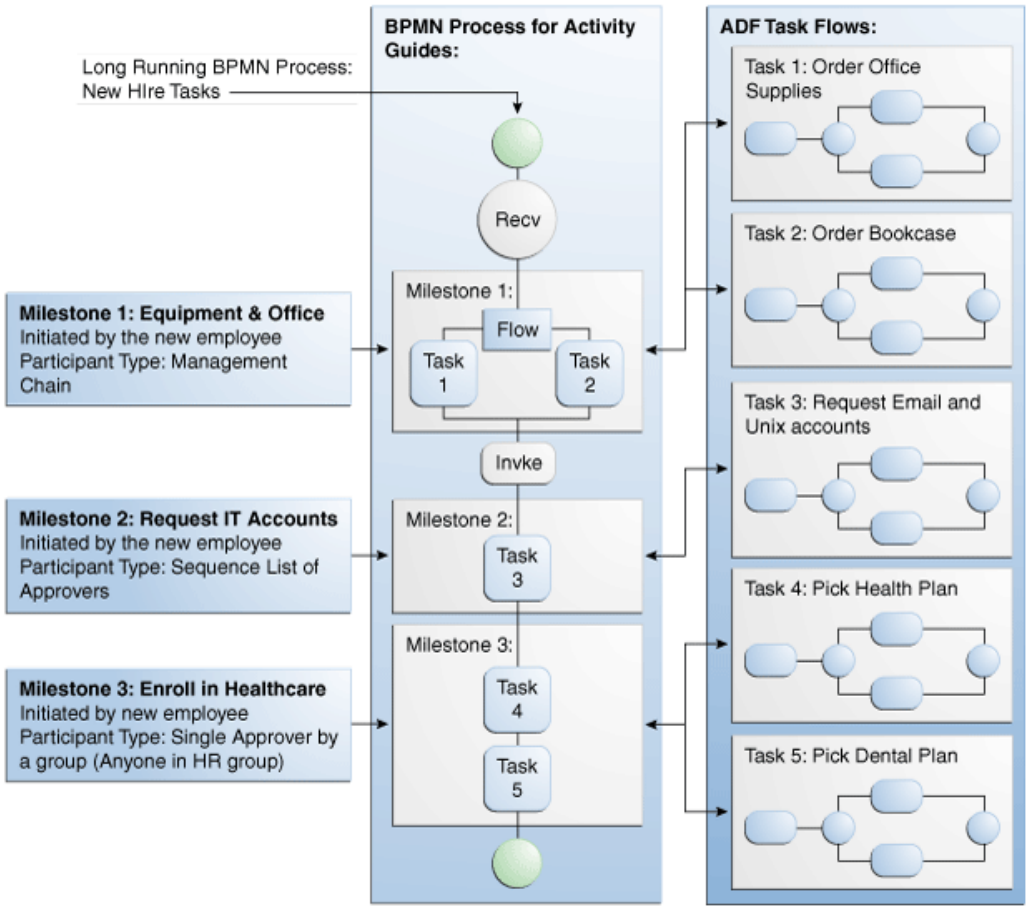


Figure 30-2 An Example of a Guided Business Process



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.

Note:

Guided Business Processes are not available in Adaptive Case Management projects. Case milestones provide a similar structure to Guided Business Processes. See Working with Adaptive Case Management.

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 Runtime

Guided Business Processes consist of both design time components and runtime interfaces.

Guided Business Process Design Time Architecture

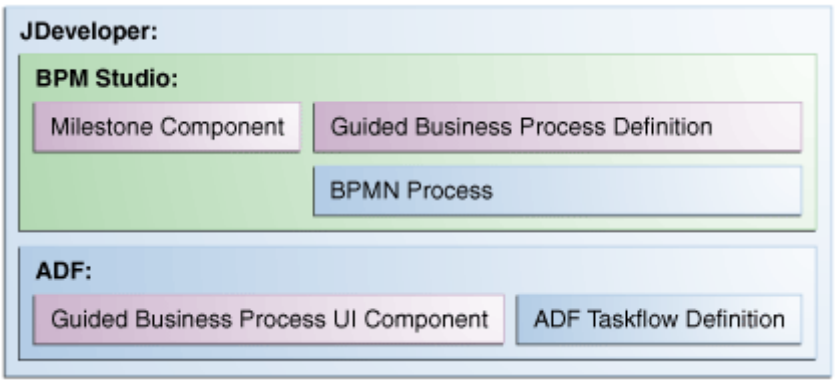
Guided Business Processes use Oracle Business Process Management to provide a comprehensive, standards-based, easy-to-use solution for creating, deploying, and managing composite application business processes with both automated and human workflow steps—all in a service-oriented architecture.

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 30-3 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 Portal: Framework
- Guided Business Process Runtime Services.

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 runtime.

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.

Guided Business Process Runtime Architecture

Guided Business Processes rely on Oracle Business Process Management to orchestrate tasks, combining Oracle Business Process Management, worklist

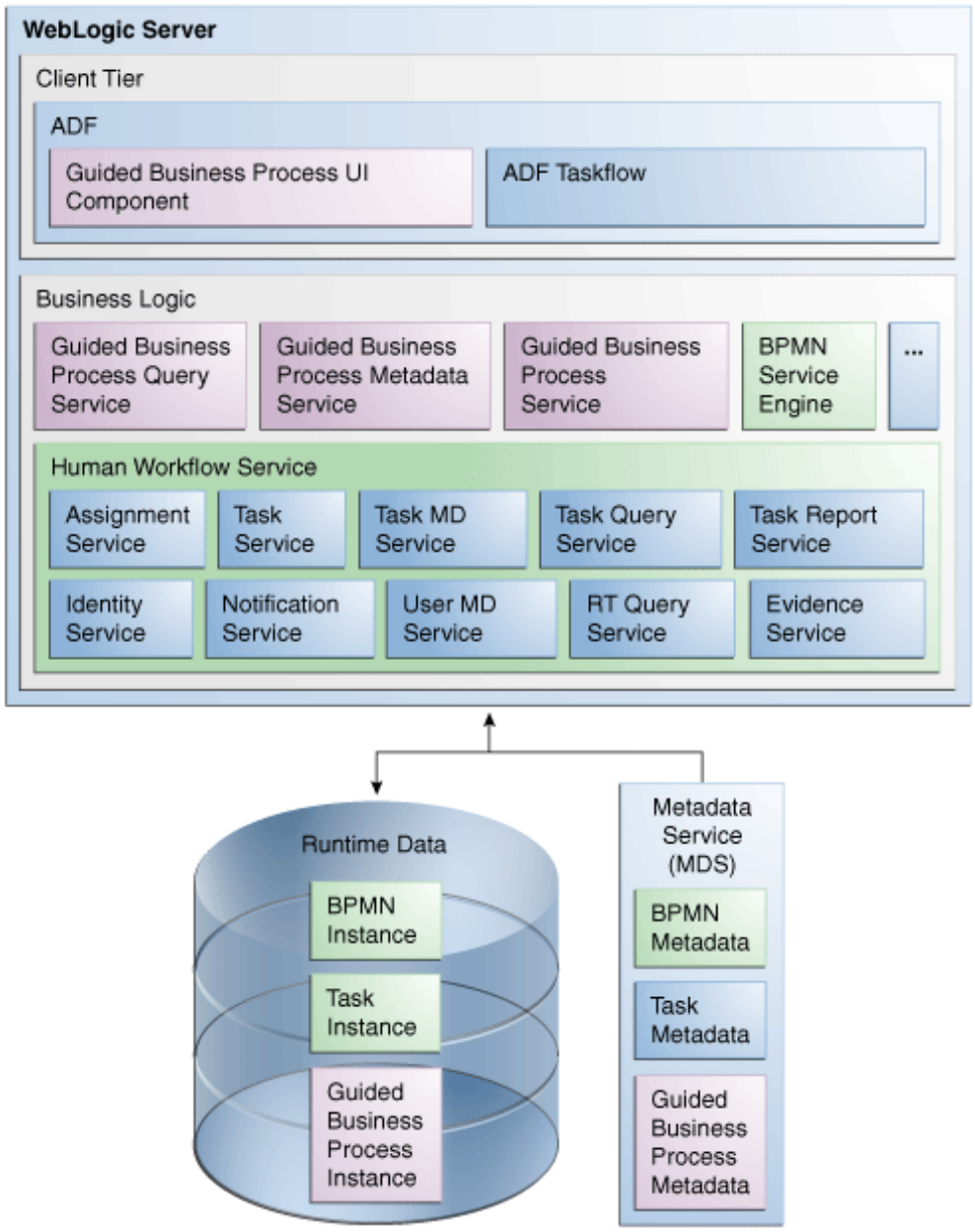
applications, and human task flows to link disparate human tasks to a greater long-running process.

At runtime, Guided Business Processes manifest as BPMN process instances orchestrating ADF task flows within Milestones. The runtime 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 Portal UI or Guided Business Process access APIs. The process instances that you view in an Guided Business Process client also appear in the Oracle BPM Worklist application, but they are not organized into milestones.

As shown in Figure 30-4, the Guided Business Process runtime architecture is composed of the client, business logic and data tiers.

Figure 30-4 Guided Business Process Runtime Architecture



Runtime support includes the Guided Business Process Query, Guided Business Process Metadata and Guided Business Process Instance Management services. The runtime 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 BPM Worklist application
- Oracle ADF
- Oracle WebCenter Portal: Framework
- Guided Business Process Runtime Services.

Client Tier

The Guided Business Process runtime 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 Portal UI or Guided Business Process APIs.
- **Oracle Business Process Management Workspace**: An out-of-the-box interface for displaying the tasks in a Guided Business Process, to users completing the guided tasks.
- Oracle Enterprise Manager Fusion Middleware Control: 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 Runtime User Interface

Following are some features of the runtime 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 Process 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 Configuring Activity Guide Properties.

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 runtime. As such, no runtime 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 runtime 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 runtime 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 runtime 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 runtime.

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 BPM Worklist application, Process Workspace, email, portals, or custom applications.

For more information about the Human Workflow Service, see the chapter "Introduction to Human Workflow" in *Developing SOA Applications with Oracle SOA Suite*.

Data Tier

Oracle Business Process Management persists Guided Business Processes, BPMN process and task instances to the database at runtime. 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.

Guided Business Process Use Cases

Guided business processes can be used for online public sector form processing and anonline loan application procedure.

The following use cases show you different situations where to use Guided Business Processes:

- Online Public Sector Form Processing
- Online Loan Application Procedure

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.

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 30-5.

Figure 30-5 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

Standards and Guidelines for Working with Guided Business Processes

Certain standards and guidelines apply to Guided Business Processes:

- Guided Business Processes must be deployed to a standalone Oracle WebLogic Server.

The Typical Flow of Developing a Guided Business Process

This section 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 runtime. To develop a front end, you can:
    - Develop a client application with Oracle ADF UI or Oracle WebCenter Portal UI.
    - Use Oracle Business Process Management Workspace application as a client application.

- Develop a custom UI with Guided Business Process runtime services.

3. Deploy the Guided Business Process client application.
4. Monitor Guided Business Process instances using the Oracle Enterprise Manager Application Server Control console.

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 runtime.

- **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 How to Localize a BPMN Guided Business Process.

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.

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 Applications window, 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**.

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.

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 Applications window, select the **Activity Guide** node.
2. In the Structure window, right-click the **Activity Guide** node.
3. Select **New Milestone**.

   The New Milestone dialog box appears.
4. Enter a title for the milestone.
5. Enter the number of tasks the user has to complete to consider this milestone completed in the Tasks Remaining field.

   At runtime, 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**.

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.

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 box 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**.

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.

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 Applications window, 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 box appears.
4. From the Milestones list, select the milestone where you want to move the user task.
5. Click **OK**.

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.

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 Applications window, select the **Activity Guide** node.
2. In the Structure window, expand the **Activity Guide** node.
3. Move each milestone to the right position:
   a. Right-click the milestone.
   b. Select **Move-Up** or **Move-Down** according to where you want to move the milestone.

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.

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 Applications window, 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**.

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.

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 Applications window, 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**.

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.

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 Applications window, 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 box appears.
6. Select **Show Task as Optional**.
7. Click **OK**.

What Happens When You Configure an Optional Task

By default all tasks 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.

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. “BPMN Flow Object Reference" in the *Developing Business Processes with Oracle Business Process Composer* for more information on how to use gateways.

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 "BPMN Flow Object Reference" in the *Developing Business Processes with Oracle Business Process Composer* for more information on how to use gateways and conditional sequence flows.

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 Applications window, 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 box 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**.

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.

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 Applications window, right-click the **Activity Guide** node.
2. Select **Configure**.
3. Click the **Browse** button, next to the Icon Location field.

   The Browse Icons dialog box appears.
4. Select an icon from your file system.
5. Click **Open**.

   The icon path appears in the Edit Activity Guide dialog box.

6. Click **OK**.

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.

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 Applications window, 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 box appears.
6. Select an icon from your file system.
7. Click **Open**.

   The icon path appears in the Edit Milestone dialog box.
8. Click **OK**.

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.

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 Applications window, 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.

Display Mode	Description
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**.

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.

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 Applications window, 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**.

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.

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 Applications window, 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**.

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.

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 Applications window, 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**.

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.

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 Applications window, 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 box appears.
6. Click **Create Resource Bundle**.

   The Create Resource Bundle dialog box appears.
7. Enter a name to identify the resource bundle.
8. Click **OK**.

   The Edit Translatable Strings dialog box 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 box 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 box appears.
15. Click the **Add** icon next to the key list to add a new translation key.

    The Create a New Key dialog box 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 box, click **OK**.
20. Localize the milestones that compose the Guided Business Process.

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 Applications window, 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 box appears.
6. Click the **Add** icon next to the key list to add a new translation key.

   The Create a New Key dialog box 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 box appears.
12. Click the **Add** icon next to the key list to add a new translation key.

    The Create a New Key dialog box 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 box, click **OK**.
17. If the milestone contains user tasks configured to display blocked icon and text, localize the user tasks that compose the milestone.

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 Applications window, 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 box appears.
8. Click the **Add** icon next to the key list to add a new translation key.

   The Create a New Key dialog box 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**.

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.

Configuring Activity Guide Properties

You can customize Activity Guides behavior by configuring their properties. To configure these properties you must edit the Activity Guide properties file.

Generally you name this file `activityguide.properties`. If you choose another name then you must provide its value to the ag-tasktree-task-flow using the parameter `AGPropsBeanName`.

Table 30-1 shows the properties you can specify in this file.

Example 30-1 shows a typical Activity Guide properties file:

Table 30-1 Activity Guide Properties

Property	Description	Possible Values
ServerConnectionMode	Specifies the mode for the transmission of data.	• SOAP • REMOTE

Table 30-1 (Cont.) Activity Guide Properties

Property	Description	Possible Values
WorklistHttpURL	Only required when using digital signatures. Specifies the URL to access the Oracle BPM Worklist application.	http://*host:port*/integration/worklisapp
SelectionFilter	Specifies the filter used to filter the processes in an activity guide.	• MY • PREVIOUS • REPORTEES • ADMIN
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
ShowCustomBlockedIcon	Specifies if the Guided Business Process shows the custom task blocked icon.	• true • false Default value: false

Example 30-1 An Activity Guide Properties File

```
#ActivityGuide Properties
ServerConnectionMode=SOAP
SelectionFilter=MY
ShowRefreshButton=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
```

Deploying a Guided Business Process to Oracle WebLogic Server

Guided Business Process are deployed to the application server in the same way as a 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.

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 30-2:

For more information about deploying an SOA composite to the application server, see "Deploying SOA Applications with Enterprise Manager" in *Developing SOA Applications with Oracle SOA Suite*.

Example 30-2 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>
```

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 *Developing SOA Applications with Oracle SOA Suite*.

Testing Guided Business Processes

You can create an instance of the deployed Guided Business Process in the Oracle Enterprise Manager Fusion Middleware 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:
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.

Example 30-3 Oracle Enterprise Manager Fusion Middleware Control Console URL

```
http://<hostname of Weblogic standalone server>:<port>/em
```

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.

31

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:

- Introduction to Building a Guided Business Process Client Application
- Developing a Guided Business Process Client Application with Oracle ADF
- Securing the Guided Business Process Client Application
- Localizing a Guided Business Process Client Application
- Guided Business Process Runtime APIs
- Developing an Example of a User Interface for Guided Business Process Tasks Using Guided Business Process Runtime Services
- Using Guided Business Process Logging

Introduction to Building a Guided Business Process Client Application

Guided Business Processes provide you with predefined ADF taskflows 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.

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.

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 runtime 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 with the Activity Guide properties.

    Note:

    Generally you name this file `activityguide.properties`. If you choose another name then you must provide its value to the ag-tasktree-task-flow using the parameter `AGPropsBeanName`.

    For more information on Activity Guide properties, see 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 Activity Guide 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 Activity Guide properties file, 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 `Activity Guide 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 31-1 illustrates adding a `ShowRefreshButton` parameter to the tree task flow.

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 31-2.

14. Create a Workflow Service client configuration file. An example is shown in Example 31-3.

Example 31-1 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>
```

Example 31-2 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>
```

Example 31-3 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>
```

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.

What Happens at Runtime: How a Guided Business Process Application Is Developed with Oracle ADF

At runtime, 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.

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 and policy enforcement.

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 How to Configure the Supported Locales for a Guided Business Process Client Application for more information on how to localize a Guided Business Process.

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 31-4 shows how the `faces-config.xml` file looks after adding a set of supported locales.

9. Set the browser locale to a supported locale.

10. Run the client page.

Example 31-4 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>
```

Guided Business Process Runtime 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.

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 *Developing SOA Applications with Oracle SOA Suite*.

Table 31-1 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 runtime data of the Guided Business Process instances involved, which you can use to render the Guided Business Process instance views in Process Workspace 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 31-1 (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 runtime 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`

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`

Developing an Example of a User Interface for Guided Business Process Tasks Using Guided Business Process Runtime 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 runtime services:

Example 31-5 Accessing the Guided Business Process Runtime 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 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.RE
MOTE_CLIENT);

    sJCooperCtx =
        wfSvcClient.getTaskQueryService().authenticate(USERNAME, PASSWORD, 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.

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.

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.

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.

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 31-2 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.

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.

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 31-3 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.
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 31-2.

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 31-6 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.

32

Using Approval Management

This chapter describes the approval management extensions available for 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 Oracle BPM Worklist, email, portals, or custom applications. Approval management extensions enable 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, approval management extensions let 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 *Developing SOA Applications with Oracle SOA Suite*:

- Designing Human Tasks
- Understanding Human Task Services

This chapter includes the following sections:

- Introduction to Approval Management
- Understanding Approval Management Concepts
- Designing Approval Management Tasks in Oracle JDeveloper
- Using the End-to-End Approval Management Samples
- Using the User Metadata Migration Utility

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 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

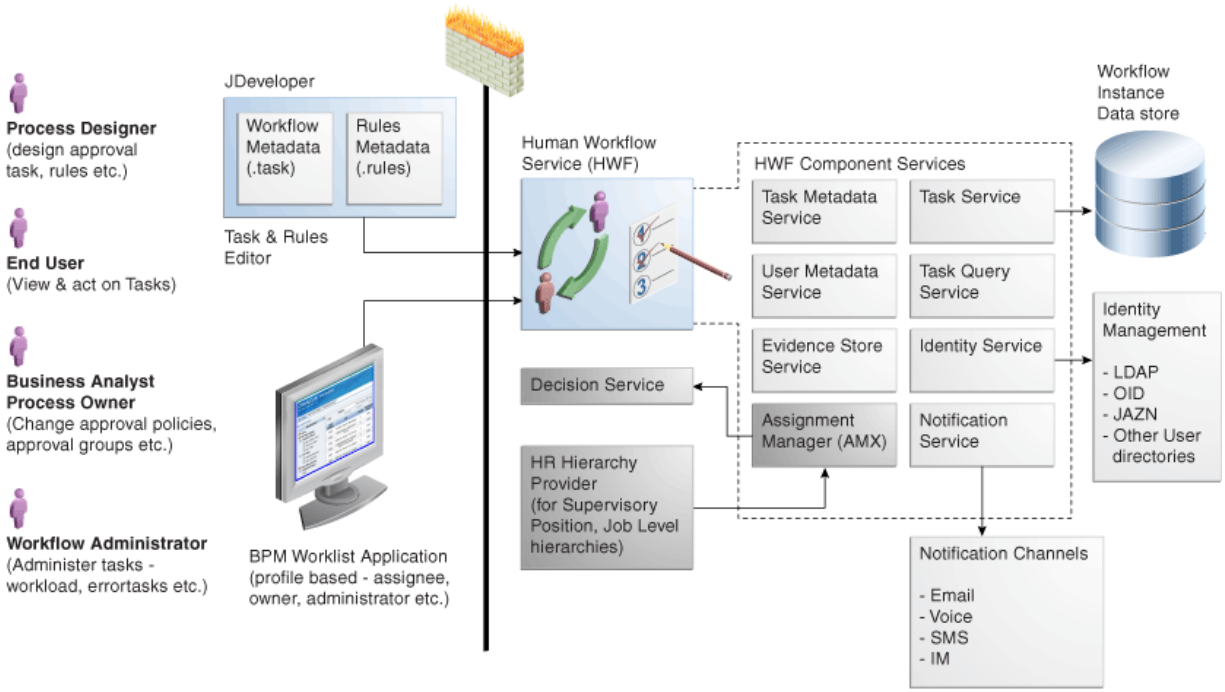
AMX provides the following additional features:

- 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

AMX Components

The following figure shows the key AMX and human task integration components. These components are described in subsequent sections of this chapter.

Figure 32-1 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 *Developing SOA Applications with 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

- **Oracle BPM Worklist**

  Oracle BPM 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. Oracle BPM 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.

Understanding Approval Management Concepts

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
- Oracle BPM Worklist

These concepts are described in the following chapters in *Developing SOA Applications with Oracle SOA Suite*:

- "Designing Human Tasks"
- "Designing Task Display Forms for Human Tasks"
- "Using Oracle BPM Worklist Application

The sections that follow describe new concepts introduced to handle complex approvals.

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.

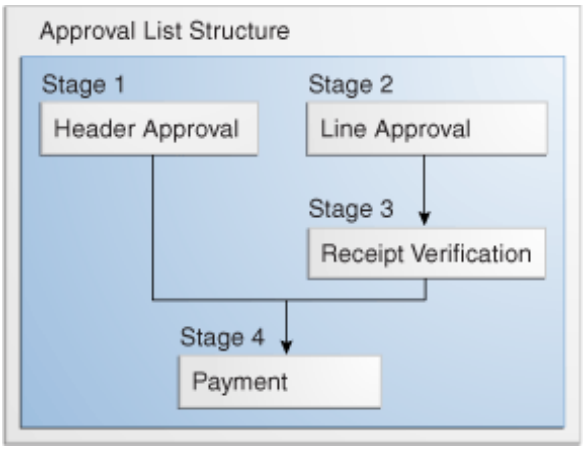
Some of the standard metadata for a task include the following:

- 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 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 List Builders.

- *Approval task configurations*, including policies for substitution and modification of approvers, configuration of self-approval, and repeated approvers. For more information, see Task.

  The following figure shows the various stages in a sample approval pattern.

Figure 32-2 Approval List Structure



These are described in more detail in *Developing SOA Applications with Oracle SOA Suite*.

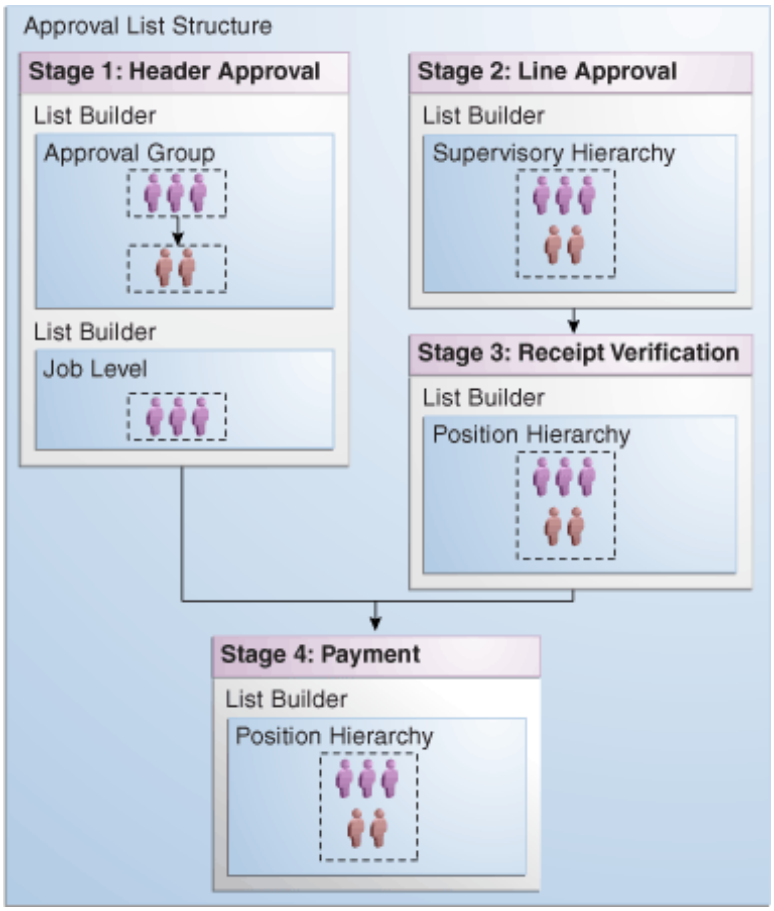
The approval pattern consists of four stages:

- 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. The following figure illustrates these concepts.

Figure 32-3 Stages and Their List Builders



These concepts are described in the sections that follow.

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.

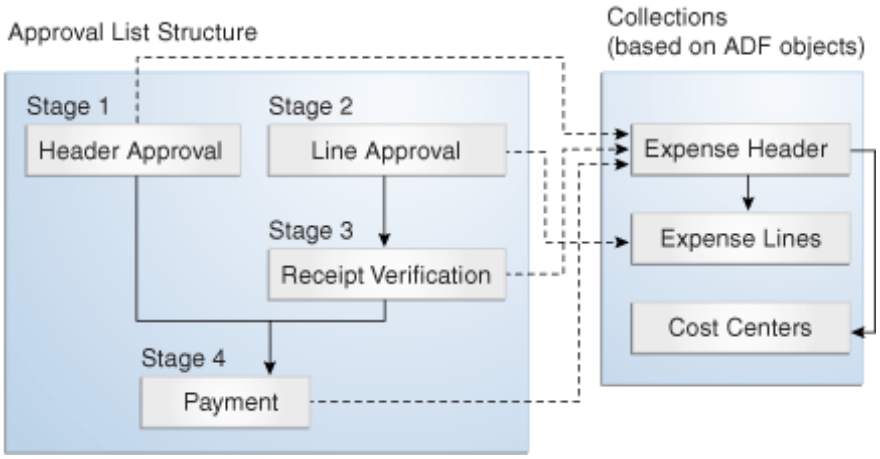
For more information, see "Implementing Business Services with Application Modules" and "Integrating Service-Enabled Application Modules" in *Developing Fusion Web Applications with Oracle Application Development Framework*.

Stages

A stage is a set of approvals related to a collection. The same collection can be associated with multiple approval stages.

The following figure illustrates the mapping of stages and collections.

Figure 32-4 Mapping of Stages and Collections



Each approval stage is associated with a collection. In the figure, 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 runtime, the appropriate set of approvals are returned based on the list builders used within the stage and on the associated policies.

List Builders

As described in Stages, 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 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 *Developing SOA Applications with Oracle SOA Suite*.

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 BPM Worklist Application" in *Developing SOA Applications with Oracle SOA Suite* for a complete list of actions.

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 *Developing SOA Applications with Oracle SOA Suite*.

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.

For more information, see How to Create Lists.

Example 32-1 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 32-2 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)
```

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.

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 How to Make Approver Substitutions.

Example 32-3 Approver-Substitution Usage

```
IF
ExpenseItems.ReceiptAmount < new BigDecimal(4000)
THEN
call Substitute(fromId:"jcooper", toId:"jstein", ruleName:"Substituted",
substitutionRules: SubstitutionRules)
```

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.

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 How to Make List Modifications.

Example 32-4 List-Modification Usage

```
IF
ExpenseItems.ReceiptAmount > new BigDecimal(3000)
THEN
Call Extend(ifFinalApproverLevel:3, extendBy:2,ruleName:"Modified",lists:Lists)
```

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.

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 *Developing SOA Applications with 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 *Developing SOA Applications with Oracle SOA Suite*.

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.

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 *Developing SOA Applications with Oracle SOA Suite*.

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 runtime. 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.

The following figure shows the globalization icon in the Human Task Editor.

Figure 32-5 Title Globalization Icon

The following procedure explains how to add translatable strings. It assumes that a resource bundle has been specified.

1. Select Translation from the drop-down list.

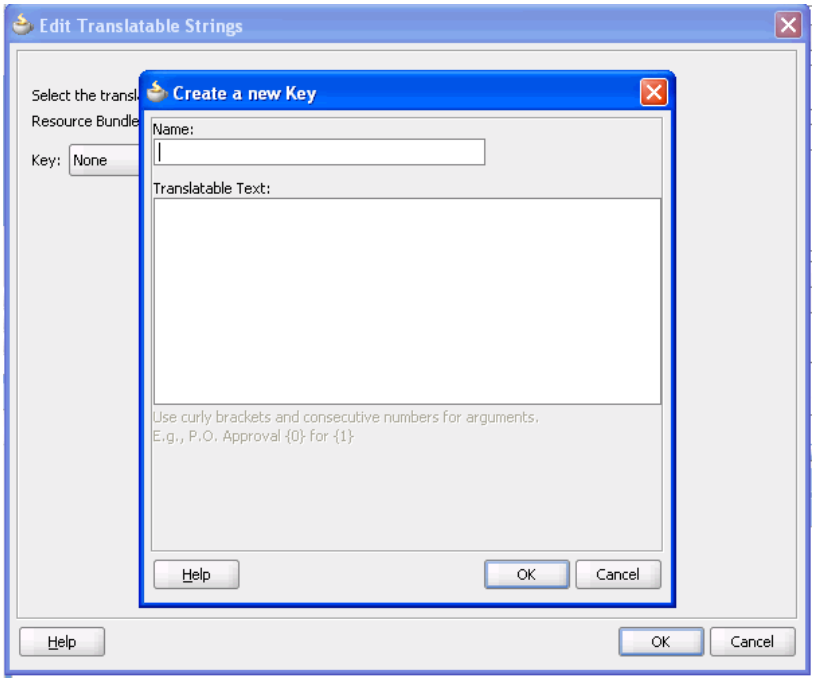
   The Global icon displays.

2. Click the icon to display the Edit Translatable Strings dialog box.

3. Select a key from the drop-down list or click the plus sign (+) to create one.

   The following Create a New Key dialog box, displays when you click the plus sign (+) on the Edit Translatable Strings dialog box.

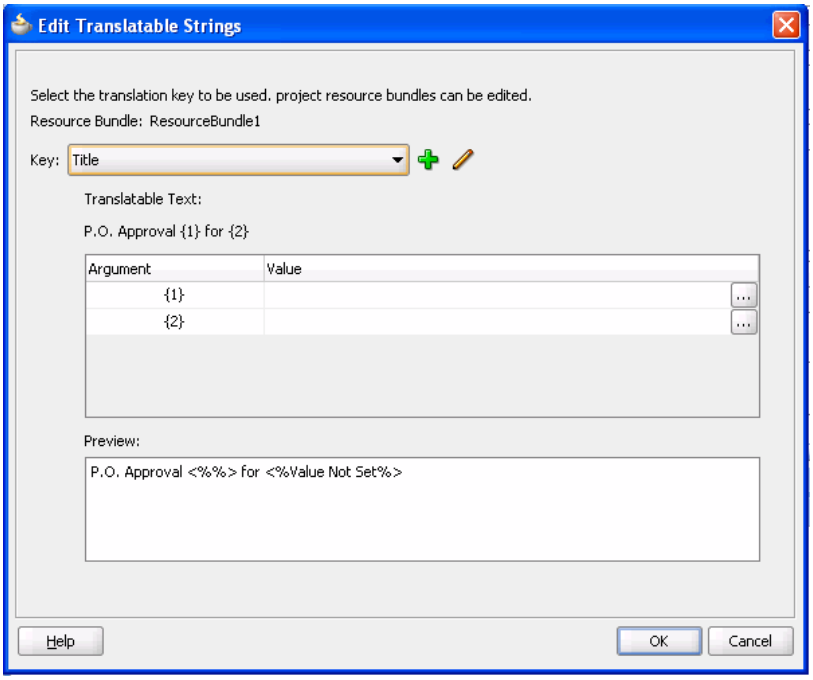
Figure 32-6 Create a New Key Dialog



4. Enter a name, the translatable text, and click **OK**.

   The New Key added dialog box shows the Edit Translatable Strings dialog box after a new key has been added.

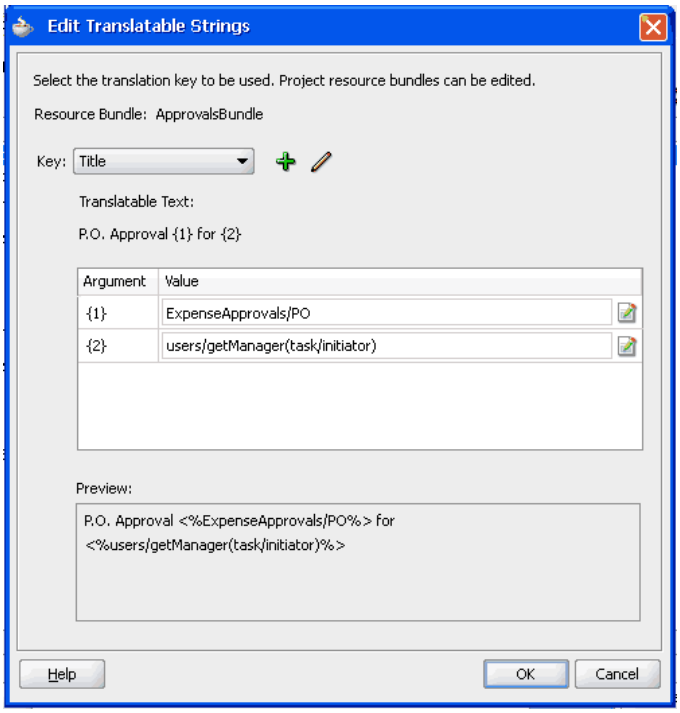
Figure 32-7 New Key Added



5. Use the Expression Builder to add values.

   The Translatable Text and Values dialog box shows the completed Edit Translatable Strings dialog box.

Figure 32-8 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.

6. Click **OK** to close the dialog box.

Specifying Task Parameters

Specifying task parameters includes the following tasks:

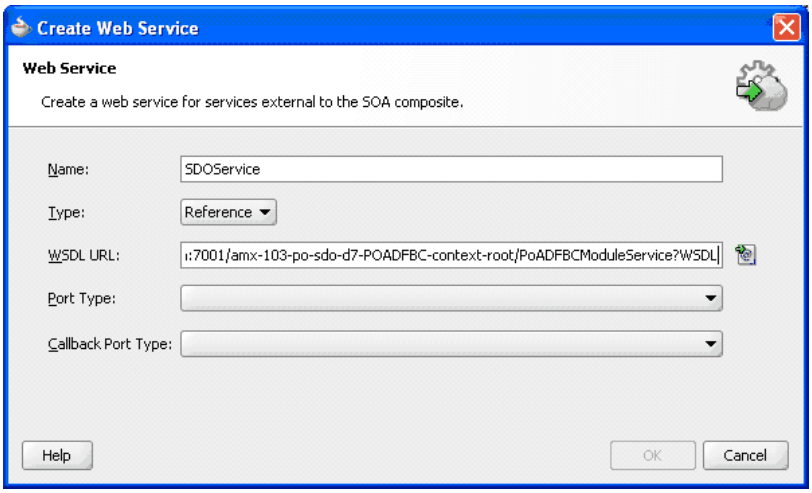
- Creating SDO references
- Defining entity parameters
- Defining collections

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 box.

To create a reference, enter the WSDL URL and select the port type from the available port types, as shown in the following figure.

Figure 32-9 Web Service Reference



For information about creating SDOs, see "Introduction to References" in the section "Introduction to the SOA Composite Editor" in *Developing SOA Applications with Oracle SOA Suite*.

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. Select a reference.

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.

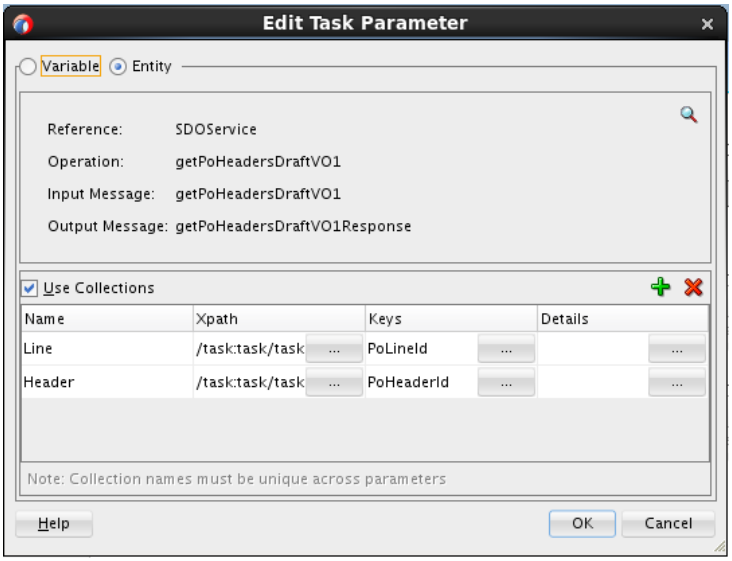
5. Click **OK**.

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. The following figure shows how collections are defined.

Figure 32-10 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 runtime. For information about how the collection information is used in a stage, see How to Model and Configure Stages.

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 runtime. 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 *Developing SOA Applications with 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 runtime, and are deployed along with the task component.

  Table 32-1 summarizes the 60 available protected flexfield attributes.

Table 32-1 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 Oracle BPM Worklist 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 Oracle BPM Worklist.
ProtectedURLAttribute1 - ProtectedURLAttribute10	Stores text data, up to 200 characters. The content in these fields is **not** checked during keyword searches in the Oracle BPM Worklist.
ProtectedDateAttribute1 - ProtectedDateAttribute10	Stores date information.
ProtectedNumberAttribute1 - ProtectedNumberAttribute10	Stores number information.

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.

How to Define Attribute-Label Mappings

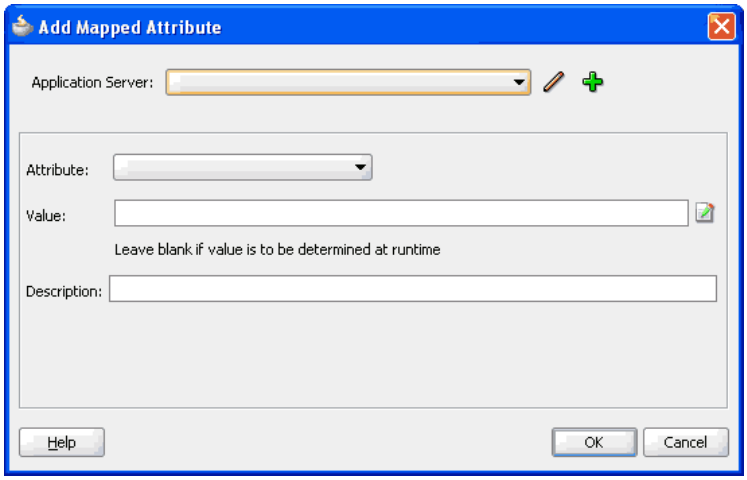
You define attribute-label mappings in the **Mapped Attributes** section of the Human Task Editor, as shown in the following figure.

Figure 32-11 Mapped Attributes Section

Use the following procedure to define attribute-label mappings:

1. Click the **Add** icon to display the Add Mapped Attribute dialog box.

   Figure 32-12 Add Mapped Attribute Dialog

2. Perform one of these options:

   - From the drop-down 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** drop-down list populates with the available attribute labels from the specified server.

3. From the drop-down 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 runtime.

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**.

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

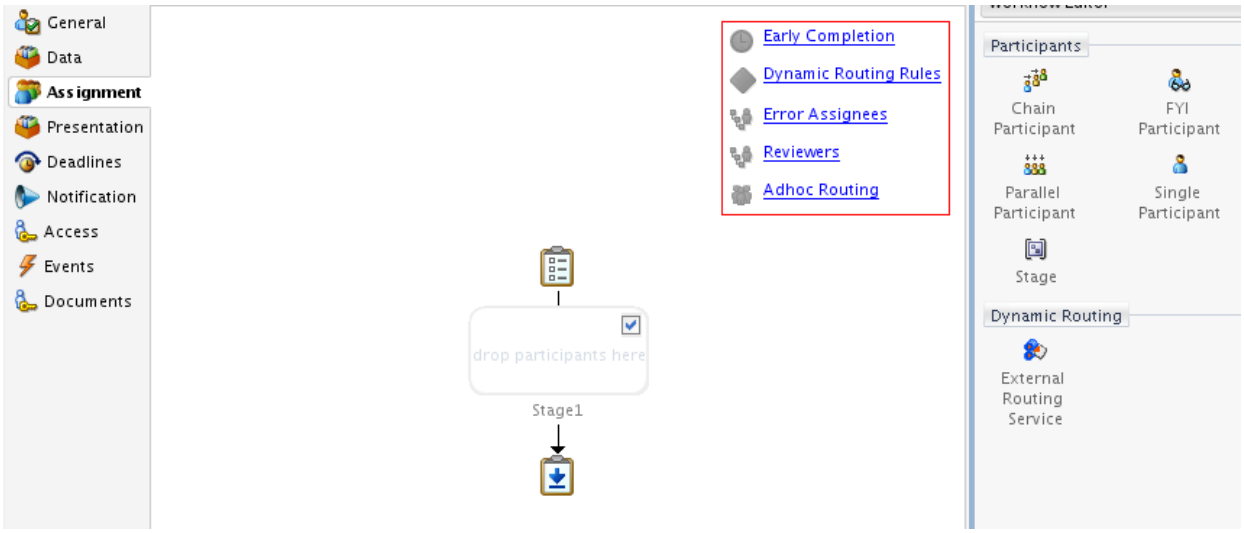
How to Model and Configure Stages

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. Drag the stage from the palette on the right side to a specific location on the canvass.

Figure 32-13 Create Stage

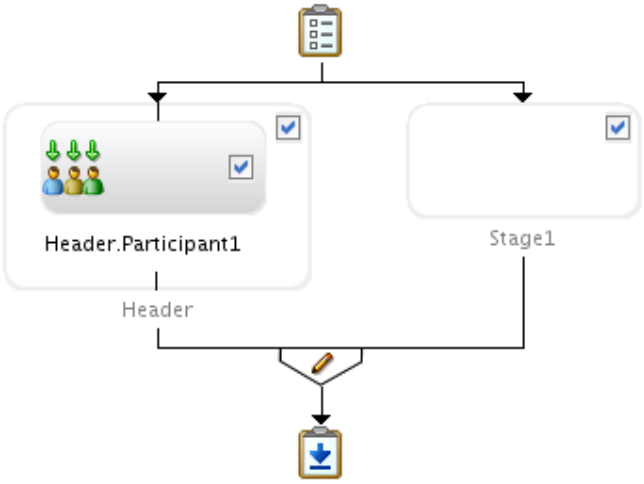


If you chose to create a sequential stage, the **Assignment and Routing** section looks like the following figure.

Figure 32-14 Add Sequential Stage

If you chose to create a parallel stage, the **Assignment and Routing** section looks like the following figure.

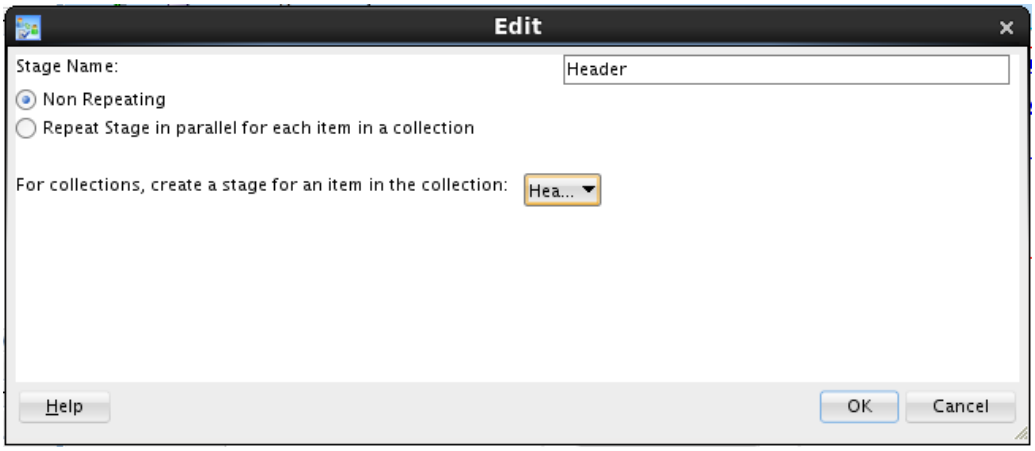
Figure 32-15 Add Parallel Stage



3. Double-click the stage you just created.

   The Edit dialog box displays, as shown in the following figure.

Figure 32-16 Edit Stage Dialog



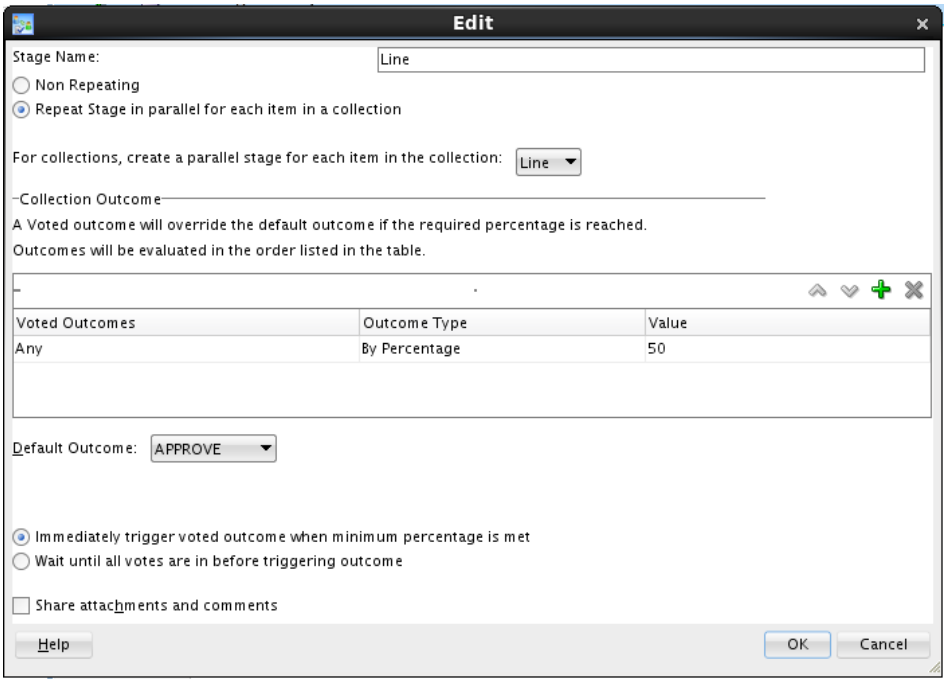
4. Enter a name for the stage.
5. Choose one of these options:

   - **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 drop-down 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 box.
   - If you selected **Repeat Stage in parallel for each item in a collection**, additional options display, as shown in the following figure.

Figure 32-17 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.

– Check the **Share attachments and comments** check box.

– Click **OK** to close the Edit dialog box.

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 *Developing SOA Applications with 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 How to Model and Configure List Builders.

How to Model and Configure List Builders

Stage uses a combination of list builders to generate the approver list. For more information, see Stages and 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 *Developing SOA Applications with Oracle SOA Suite*.

Table 32-2 describes the AMX-specific list builders and the options available to them.

Table 32-2 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
Top Participant	The last participant in the approval. Approval does not go beyond this participant in a hierarchy.	Job Level Position Supervisory

Table 32-2 (Cont.) List-Builder Options

Option Name	Description	List Builder
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 32-5 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 32-5 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>
```

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 runtime. 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 Oracle BPM Worklist'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

Use dynamic approval groups when you need to calculate the approval group dynamically based on the task payload. Specially in line level approval where each line may require different approval group. For example, each cost center may require the approval of a different cost center owner. Each line may have different cost centers that require the approval of different cost center owners. When the number of cost centers is greater than one hundred, this may become difficult to manage with business rules.

Two views of the Approval Groups list builder are shown in the following figures.

Figure 32-18 Value-Based Approval Groups List Builder Dialog

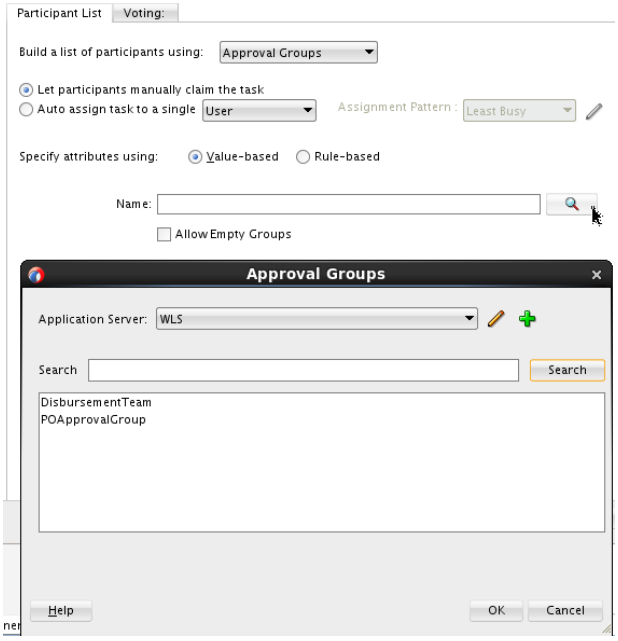
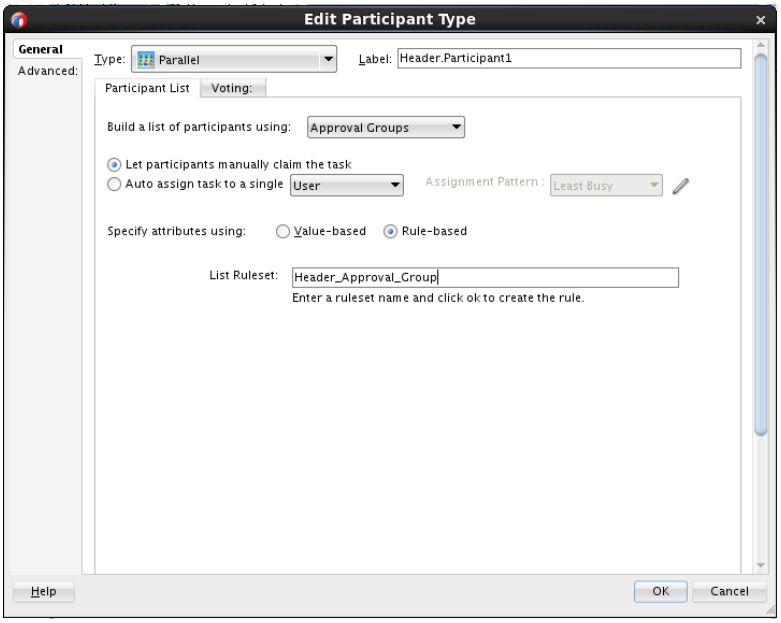


Figure 32-19 Rule-Based Approval Groups List Builder Dialog



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 box. For information about the options, see Table 32-2.

Note:

If you configure the resource list with a group, then it behaves as a single type participant regardless of the serial or parallel type configuration.

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 the following figures.

Figure 32-20 Value-Based Job Level List Builder Dialog

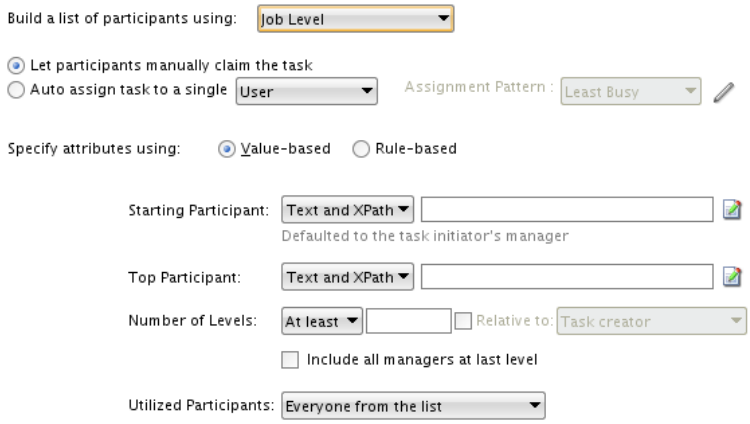
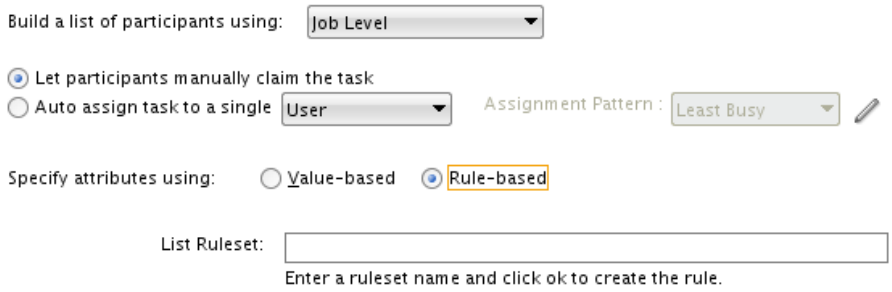


Figure 32-21 Rule-Based Job Level List Builder Dialog



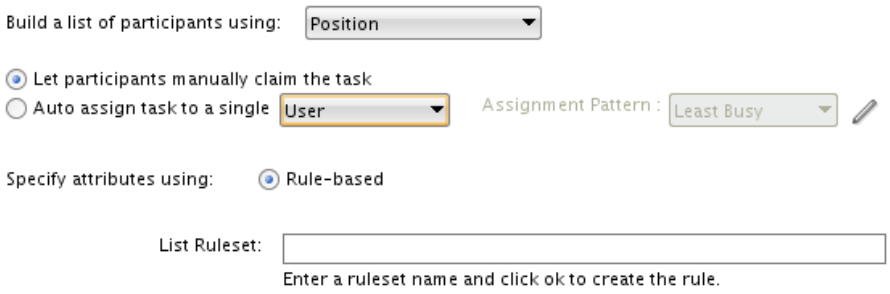
To model a Job Level list builder, first specify if the list builder's attributes are to be value-based or rule-based, and then select the options on the corresponding dialog box. For information about the options, see Table 32-2.

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.

The following figure shows a view of the Position list builder.

Figure 32-22 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 box. For information about the options, see Table 32-2.

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 the following figures.

Figure 32-23 Value-Based Supervisory List Builder Dialog

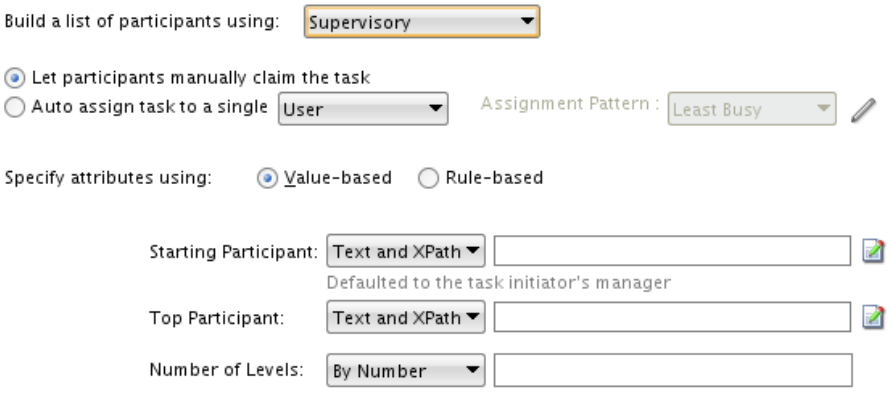
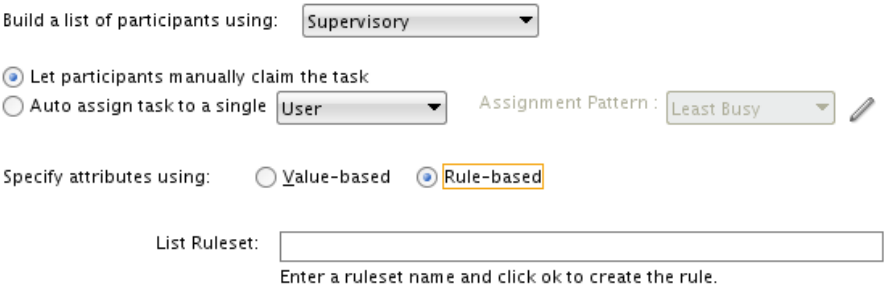


Figure 32-24 Rule-Based Supervisory List Builder Dialog



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 box. For information about the options, see Table 32-2.

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 *Developing SOA Applications with Oracle SOA Suite*.

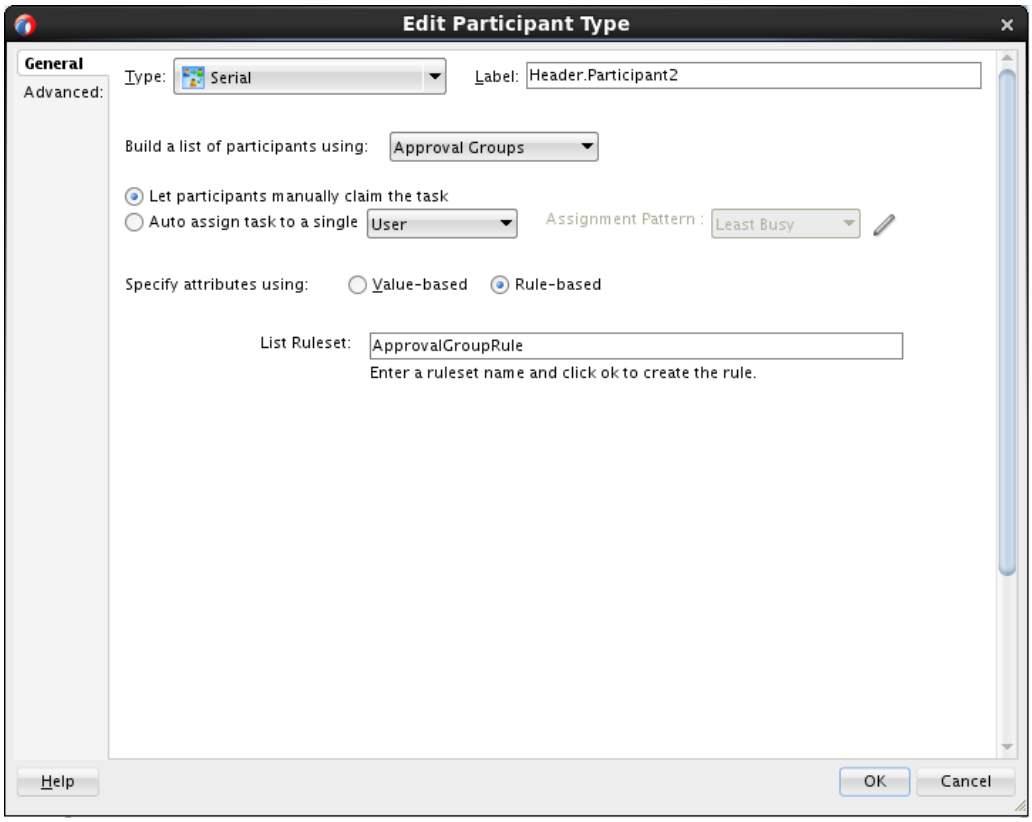
The sections that follow explain list creation, approver substitution, list modification, and repeating node attributes using Oracle Business Rules.

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:

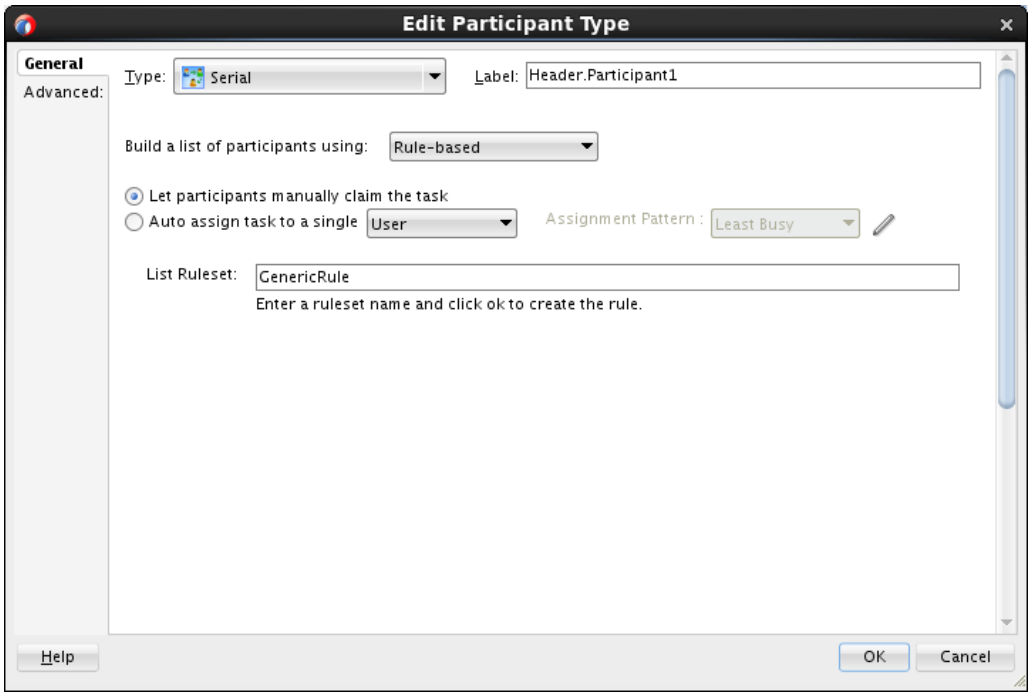
- Rules that define the parameters of a specific list builder. In this case, the task routing pattern dialog box is modeled to use a specific list builder. The parameters in the list builder come from rules. With this option, rules should return a list builder of the same type as the one modeled in JDeveloper. The following figure shows a sample configuration.

Figure 32-25 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. The following figure shows a sample configuration.

Figure 32-26 List Builder and Parameters Configuration

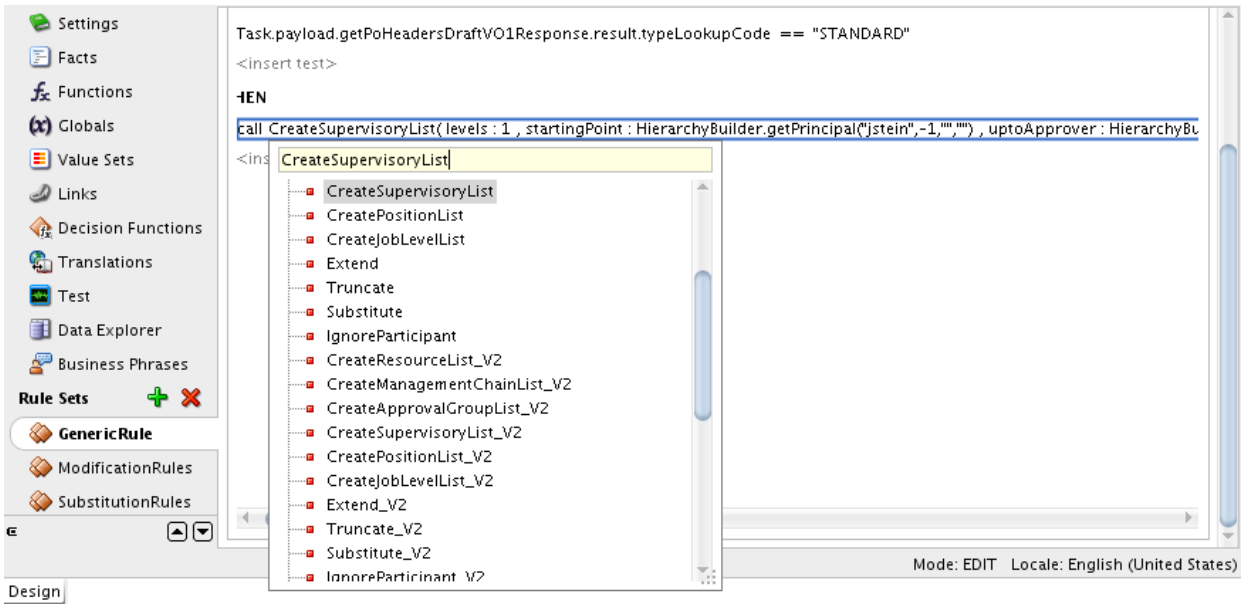


In the rule dictionary, rule functions are seeded to facilitate the creation of list builders. These functions are the following:

- 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 the following figure.

Figure 32-27 Modeling Conditions in Rules Designer



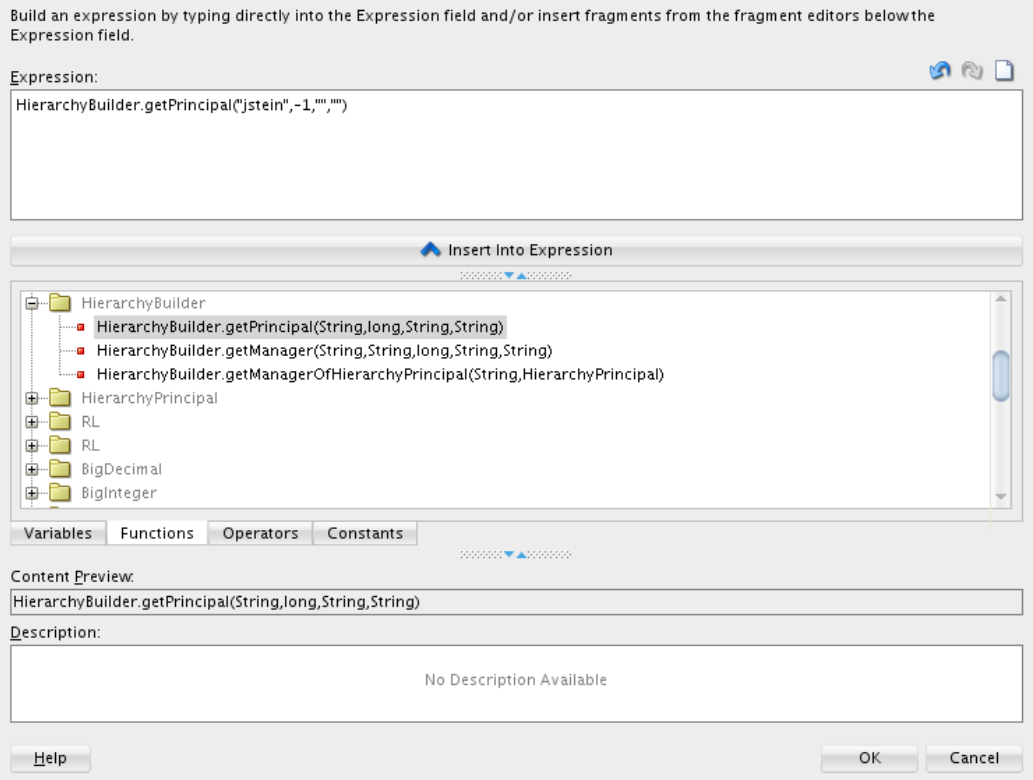
The parameters for the rule functions are similar to the ones in JDeveloper modeling. In addition to the configurations in JDeveloper, some additional options are available in Rules Designer for the following attributes:

- 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 the following figure.

Note:

If you want to leave the job level attribute undefined when using the HierarchyBuilder function, then you must set its value to a negative integer.

Figure 32-28 HierarchyBuilder Function



HierarchyBuilder has a number of functions including getManager, getPrincipal, and getManagerOfHierarchyPrincipal.

HierarchyBuilder.getManager builds an approval list and takes the following parameters:

ListbuilderType - String - can be "supervisory"," joblevel" or "position"

ReferenceUser - String - for example "Task.creator"

AssignmentID - long - the default value is -1, otherwise it is set to the user

EffectiveDate - String - for example, "2015–07–31"

HierarchyType - String - specifies the type of manager to look for when the list is built. Example values are "LINE_MANAGER", "RESOURCE_MANAGER", "CORPORATE_MANAGER", "PROJECT_MANAGER"

An example HierarchyBuild.getManager call is
`HierarchyBuilder.getManager("supervisory",Task.creator,-1,"2015-07-31","LINE_MANAGER")`

HierarchyBuilder.getPrincipal locates an approval list member and can be used, for example, to identify the top approver in an approval list. It takes the following parameters:

PrincipalName - String - can be "supervisory"," joblevel" or "position"

AssignmentID - long - the default value is -1, otherwise it is set to the user

EffectiveDate - String - for example, "2015–07–31"

HierarchyType - String - specifies the type of manager to look for when the list is built. default is "LINE_MANAGER". Other possible values are "RESOURCE_MANAGER", "CORPORATE_MANAGER", "PROJECT_MANAGER"

- 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.

The following figures show examples of rules.

Figure 32-29 Example Rules (1)

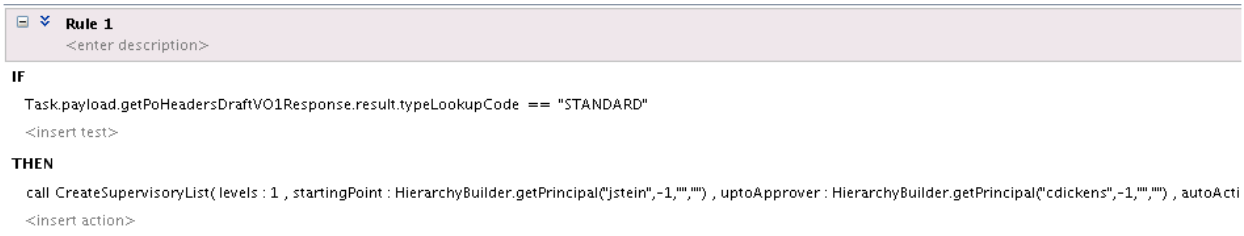


Figure 32-30 Example Rules (2)

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 runtime 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.

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 32-3 lists the "Substitute" functions and their parameters.

Table 32-3 "Substitute" Function Parameters

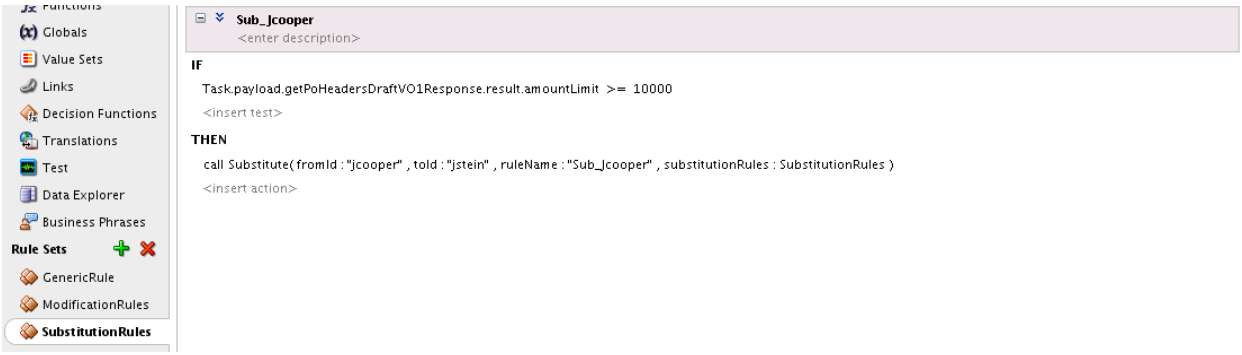
Parameter	Description
fromId	The ID of the user/group/application role from which to substitute.
toId	The ID of the user/group/application role which to substitute to.
ruleName	Used to create an assignment reason. Rule set name + "_" + rule name is used as a key to look up the resource bundle for a translatable 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.

Note:

In a Human Task with a substitution rule, the resulting approval list might have a duplicate participant. It is not possible to edit the duplicate approvers in the Future Participants list.

The following figure shows a sample approver-substitution action.

Figure 32-31 Sample Approver-Substitution Action



How to Make List Modifications

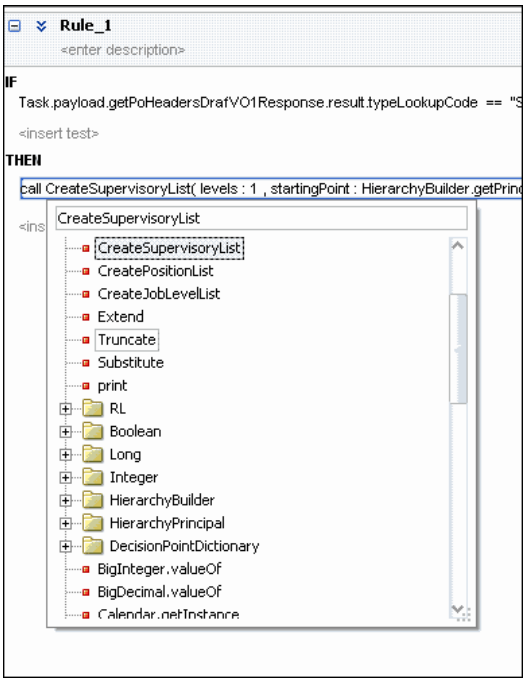
List modification enables you to extend or truncate the Job Level and Position list builders from rules. List modification is applied after the list is created. This feature does not require any configuration from JDeveloper. In each rule dictionary there is a pre-seeded rule set named "ModificationRules." 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 the following figure.

Figure 32-32 Rule Functions



Extend and truncate parameters are listed in Table 32-4 and Table 32-5.

Table 32-4 "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 32-5 "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.

The following figure shows a sample list-modification action.

Figure 32-33 Sample List-Modification Action

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

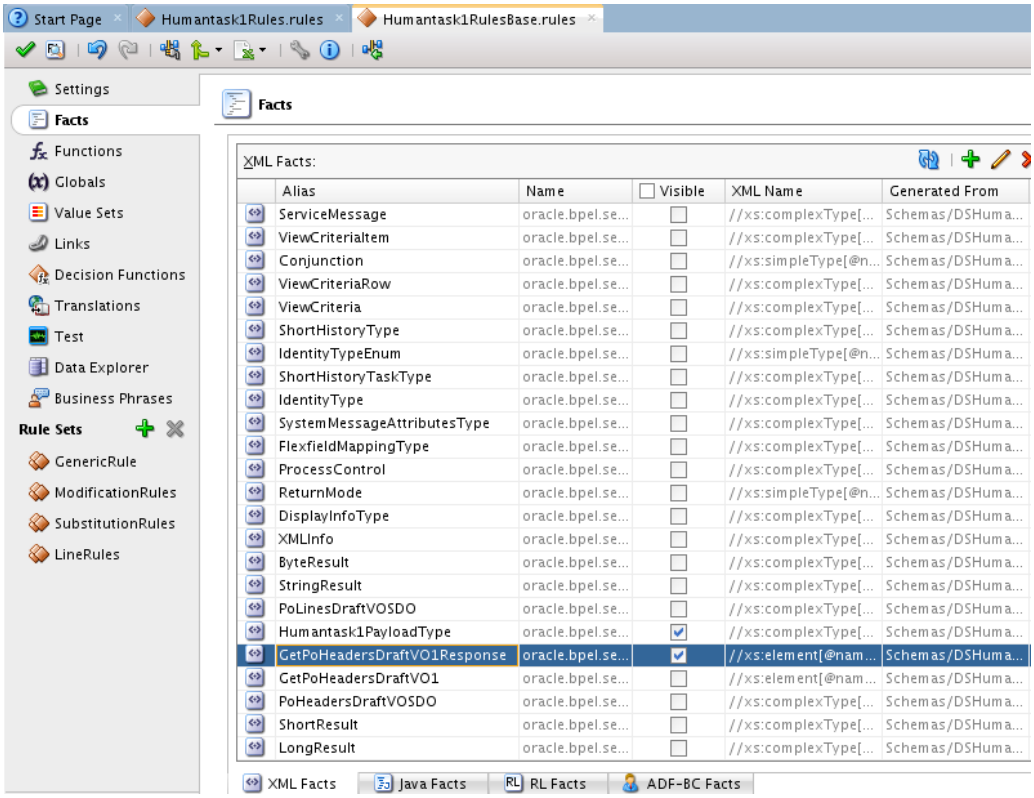
the amount is an attribute of *line*, that is, it is an attribute of a repeating node.

Use the following procedure to define repeating-node attributes:

1. In Base Dictionary, select **Facts**.

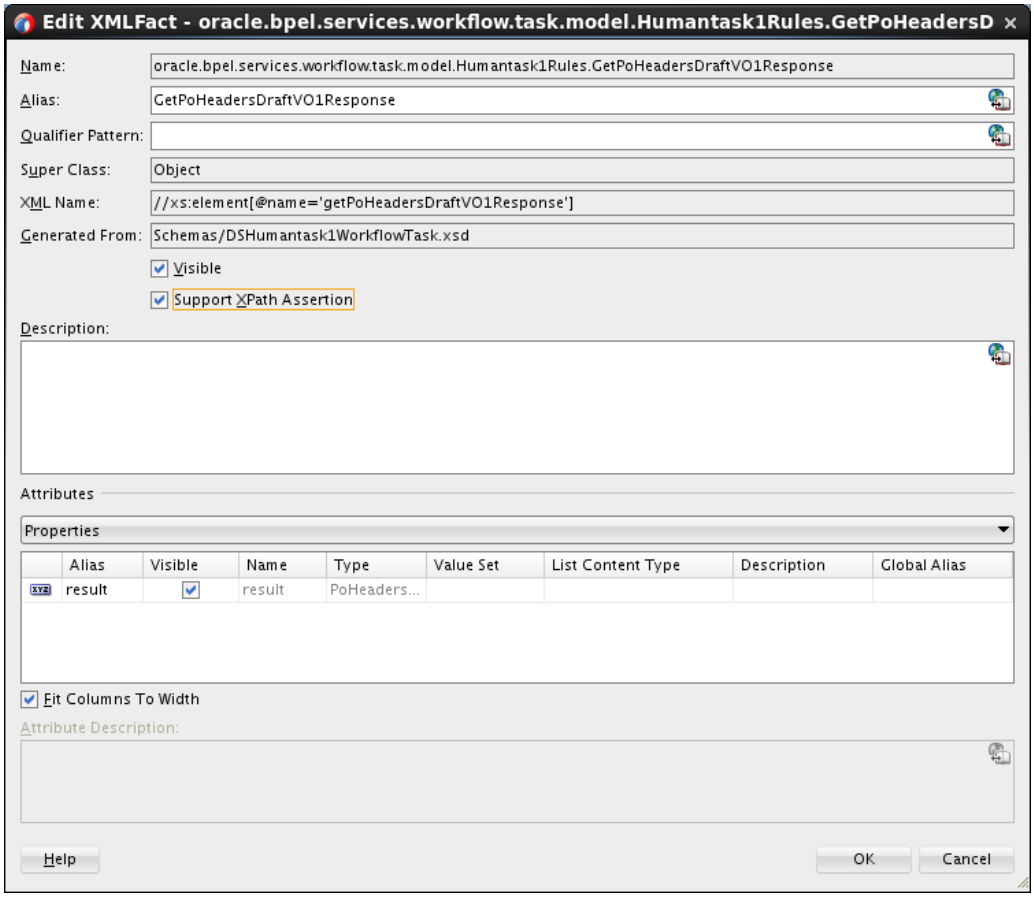
   In the Humantask1RulesBase rules tab, a list of facts displays as follows.

Figure 32-34 Facts List



2. Edit each appropriate fact to ensure that it is visible, as shown in the following figure.

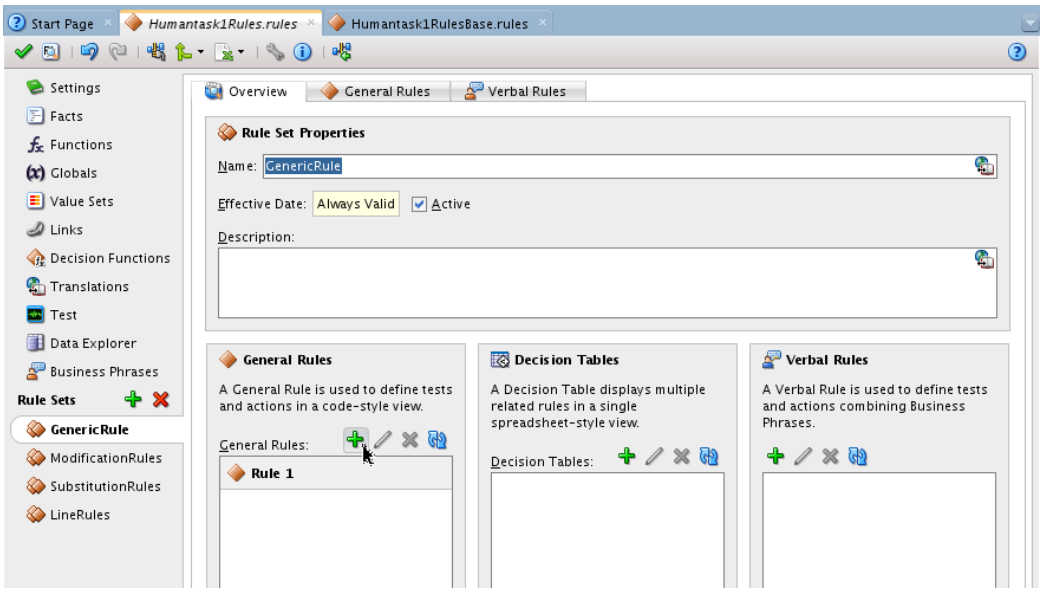
Figure 32-35 Edit XML Fact Dialog



3. Decide whether you want to add a generic rule, a decision table, or a verbal rule. Once you decide, click the Add (+) button. In Rules Designer, select a rule and click **Add** icon (+).

   The following rule-definition section displays.

Figure 32-36 Rule-Definition Section



4. Click the double down arrows to the left of the rule name to show advanced settings, as shown in the following figure.

Figure 32-37 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 the following figure.

Figure 32-38 ROOT Options

6. Define the rule conditions.

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 Oracle BPM Worklist uses assignment context when it displays task history.

Configuring Assignment Context

You configure assignment context in the Add (or Edit) Participant Type dialog box 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.

  Assignment context can also be assigned in value-based context. See Assigning Task Participants for more information.

- Select the **Advanced** finger tab to configure any number of assignment contexts.

  In this case, you can customize assignment contexts by entering your own information into the Assignment Context fields. the following figure shows the fields.

Figure 32-39 Assignment Context Section

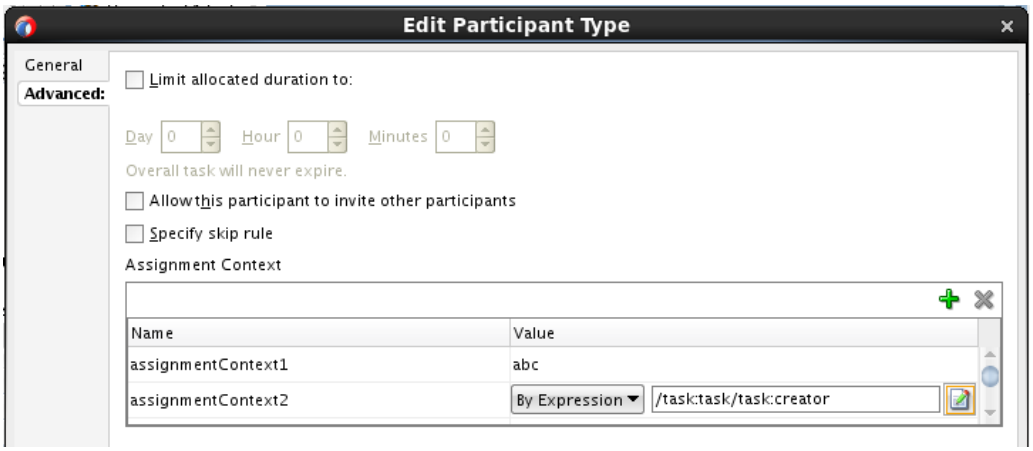


Table 32-6 contains field descriptions.

Table 32-6 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.

Table 32-6 (Cont.) Assignment-Context Field Descriptions

Field	Description
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.

How to Aggregate Task Approvals

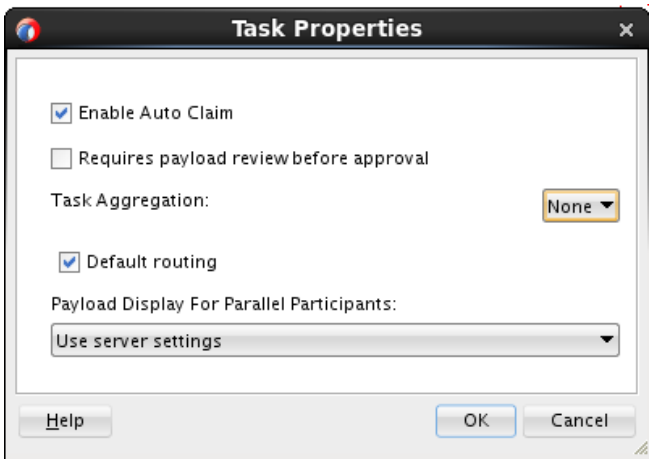
A task can be assigned multiple times to one user during the task life cycle. The Human Task Editor enables you to configure how often a user sees the task.

The following procedure explains how to configure task-approval aggregation.

1. Click **Configure** in the top.

   The Task Properties window displays as follows.

   Figure 32-40 Task Properties



2. Select a task-aggregation option from the drop-down 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.

3. Click **OK**.

When the task is aggregated and assigned to a user, the task has a collection table in the Oracle BPM Worklist that displays 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.

Aggregated tasks are business tasks and show the actions approve and reject. If you can aggregate FYI tasks, then they show the approve and reject actions. In this case the approve and reject actions are treated as an acknowledgement.

Note:

Aggregation is available only when the assignees are from the same set. For example, if you assign a task to user A and another to both user A and user B; then user A sees two separate tasks. The two assignments are not aggregated because the assignees are not exactly the same.

Defining Escalation and Renewal Policies

This feature is not specific to AMX. For more information, see "Designing Human Tasks" in *Developing SOA Applications with Oracle SOA Suite*.

Note:

Escalation is only applicable to management chain.

Specifying Notification Settings

This feature is not specific to AMX. For more information, see "Designing Human Tasks" in *Developing SOA Applications with Oracle SOA Suite*.

Using Advanced Settings

Using advanced settings includes the following tasks:

- Specifying callbacks for notes, attachments, and validation
- Defining security access rules

How to Add Callbacks for Notes, Attachments, and Validation

Callbacks are mechanisms that allow you to do the following:

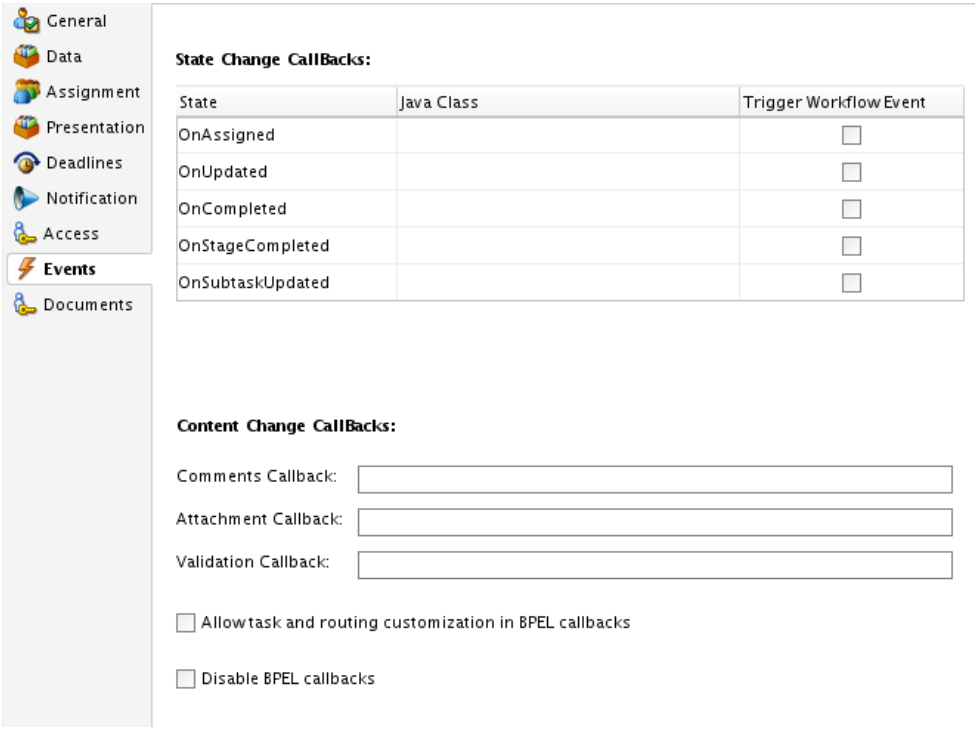
- Access notes and attachments associated with business objects from external content-management systems or custom schemas
- Perform custom validation of workflow tasks at various points in a task life cycle by defining validation logic for each task action

Use the following procedure to add callbacks:

1. From the Task Editor, select the 4 finger tab to configure the callbacks.

   The Callback Details dialog box opens as follows.

Figure 32-41 Callback Details Dialog



2. 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.

3. Click **OK**.

How to Define Security Access Rules

Access rules restrict the actions that a user can perform by overriding default actions and permissions. At runtime, 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.

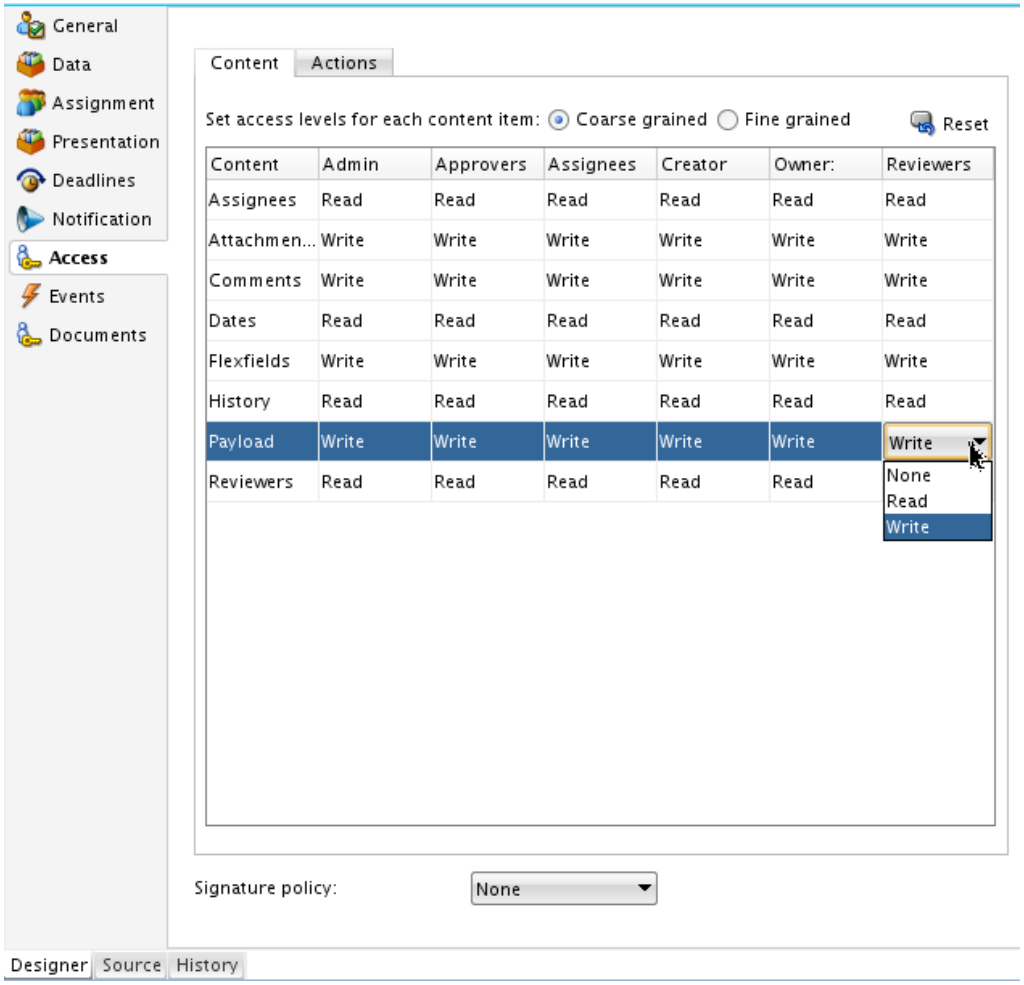
To define a security access rule:

1. Select the **Access** finger tab to display security access rules.

2. Click **Configure Visibility...**.

   The Configure Task Content Access dialog box displays, as shown in the following figure.

Figure 32-42 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. Look up the appropriate content and role in the grid.

5. From the drop-down list, select the appropriate privilege or action.

6. Click **OK** to close the dialog box.

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 drop-down list.
- Select application roles to include in the access rule from the Select an Application Role dialog box, as shown in the following figure.

See Figure 29-62 in Specifying Access Policies and Task Actions on Task Content for more information.

For more information, see the section "Specifying Access Rules on Task Content" in the chapter "Designing Human Tasks" in *Developing SOA Applications with Oracle SOA Suite*.

Using the End-to-End Approval Management Samples

You can use samples of end-to-end approval management.

Table 32-7 shows the end-to-end workflow examples included in the `ORACLE_HOME\samples\soa-infra\workflow\amx` directory.

In addition to the demonstration features listed in the table, all samples show the use of worklist applications and workflow notifications.

Table 32-7 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`

Using the User Metadata Migration Utility

The user metadata migration utility, **hwfMigrator**, 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 *Administering Oracle SOA Suite and Oracle Business Process Management Suite*.

33

Working with Adaptive Case Management

This chapter describes how to create and configure a case management definition.

This chapter includes the following sections:

- Introduction to Adaptive Case Management
- Creating a Case
- Configuring a Case
- Configuring Case General Properties
- Configuring Case Data and Documents
- Configuring Case User Events
- Defining Case Stakeholders and Permissions
- Defining Case Tag Permissions
- Localizing a Case
- Case Activities and Sub Cases
- Defining Input Parameters for Case Activities
- Defining Output Parameters for Case Activities
- Configuring Case Activities
- Creating a Global Case Activity
- Using Business Rules with Cases
- Closing Cases
- Integrating with Oracle BPM
- Schema Reference

Introduction to Adaptive Case Management

Adaptive case management is a way of modeling very flexible and data intensive business processes.

Use adaptive case management to model a pattern of work with the following characteristics:

- Complex interaction of people, content and policies
- Complex decision making and judgments

- The progress of the case depends on user decisions, actions, events, and policies
- Changes at runtime, for example, adding new stakeholders enables new actions
- Context-driven assignments, for example, assignments based on the number of cases resolved by a certain analyst and the time it took them to resolve them

Case management enables you to handle unstructured ad-hoc processes. It relies on the content and information of the process so that the user can make informed business decisions. It focuses on unpredictable business processes which rely on worker knowledge and involve human participants.

Case management involves:

- People, often referred to as case workers or knowledge workers
- Data
- Documents
- Collaboration
- Reporting
- History
- Events
- Policies
- Processes

A case is a collection of information, processes, tasks, rules, services. It requires the worker's knowledge, their involvement and active collaboration to move the case forward.

Adaptive case management enables you to define only the activities a user performs to achieve a goal without defining the workflow process. However, it does supports dynamic workflows, structured processes and a combination of both.

A case definition contains various case activities that represent the different work that users can perform in the context of a case. Oracle BPM allows you to define case activities based on:

- a Human Task
- a BPMN process
- a custom Java class

Differences Between Adaptive Case Management and Business Processes

Adaptive case management allows the end user to define the case flow at runtime, while business processes require you to define the flow at design time. Adaptive case management uses documents and contextual information to determine the flow of the case at runtime.

Table 33-1 illustrates the differences between adaptive case management and business processes.

Table 33-1 Differences between adaptive case management and business processes

Adaptive Case Management	Business Process
Data centric	Process centric
Adhoc, unstructured progress of a case from creation to its final state	The process instance follows a predefined workflow
Non-deterministic - the case flow is dynamically determined at runtime	The process flow logic is defined at design time
The case consists of a collection of processes and isolated tasks	The flow logic is expressed in a process model
The flow is determined by objectives, the case workers choose actions to meet a certain goal	The flow is designed to automate and improve processes to increase efficiency
Knowledge work based	Routine work based
Collaborative environment	Collaboration requirements are not a priority
Strongly relies on documents	Uses structure data

Adaptive Case Management Artifacts

The key artifacts in adaptive case management are:

- **Case:** a collection of structured, semi-structured, un-structured processes, information and interactions used to make a business decision.
- **Case Model:** models the definition of the case. The case has multiple attributes such as name, type, various milestones. It also has associations like the behavior container and root folder. The definition of the case is the collection of these attributes and associations.
- **Case Instance:** a collection of documents, data and case activities that are used to process the case and audit the progress of the case.
- **Case Folder:** the folder(s) in the Content Management System where the case information is stored.
- **Case Data:** the data for the case stored in the BPM database.
- **Case Lifecycle**: the case lifecycle is reflected by the case state, which can be one of active, stale, suspended, aborted, or closed.
- **Milestones:** checkpoints that indicate the progress of a case and represent the completion of a deliverable or a set of related deliverables. Stakeholders can use milestones to obtain a high level view of the status of the case.
- **Case Activity:** the work that can be performed in the context of a case. Case activities have various properties that define their behavior. Case activities can be mandatory, conditional, or optional. Case activities can be manual or automatic. Manual case activities require a case worker to initiate them while automatic case activities are initiated by the case runtime. You implement case activities using a BPMN process, a Human Task or a custom java class.

- **Sub Case:** a child task of a case, used when additional activities must be spawned as part of the processing of the parent case. Sub cases are instantiated at run time and are similar to case activities, except that they only inherit data from the parent case.
- **Case Event:** include case lifecycle changes, case milestone changes, case activity changes and other manual case events. Manual case events are events modeled in the case corresponding to various manual actions that can occur during the case processing.

Use Cases

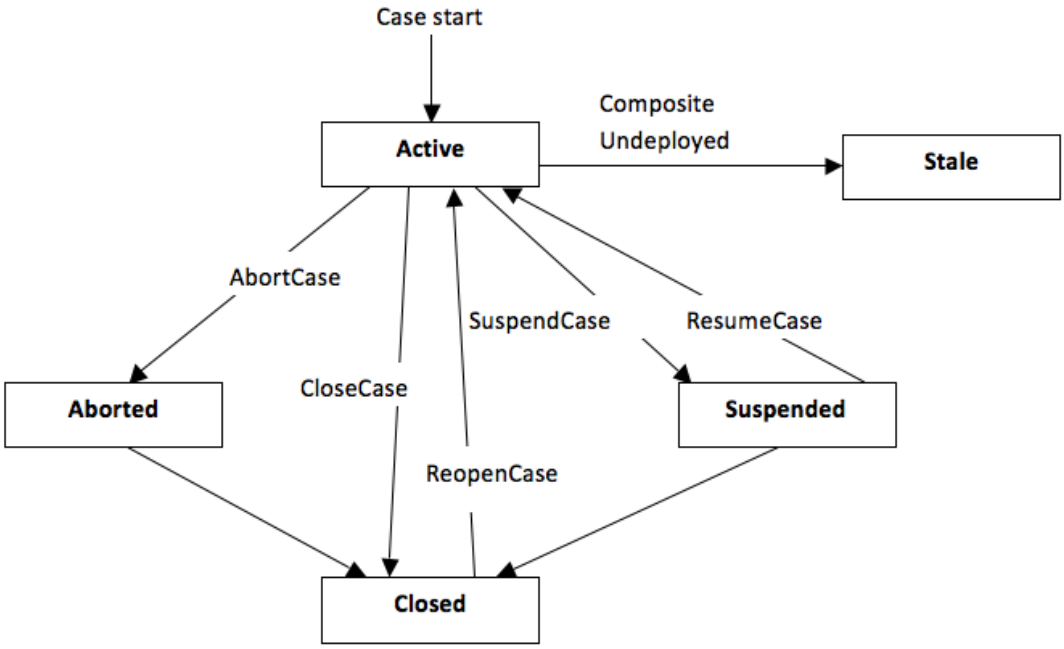
Adaptive case management is suitable to model business processes in many different industries and scenarios:

- **Financial Services:** loan origination, credit and debit card dispute management, financial crime management (suspicious activity reporting), wealth management, brokerage, trading, new business account opening, e-bank account opening, accounts payable, accounts receivable, and B2B order management.
- **Insurance:** P&C claims processing, policy management, policy servicing, underwriting, fraud prevention, customer on-boarding.
- **Health Care:** payer claims processing, policy and procedure management, virtual patient records management, member service management, provider service management, group sales management, health plan insight, clinical and operational insight.
- **Energy & Utilities:** process safety management, FERC e-tariff, transmittal process, SOP processing.
- **Public Sector**: citizen benefits eligibility and benefit's enrollment, grant management, public safety, tax and custom filling, court solution and judicial matters.
- **Human Resources:** employee on-boarding, employee off-boarding, employee performance review, employee benefits administration.
- **Legal:** contract management, legal matter management, auditing and compliance.
- **Customer Service**: customer correspondence, call center, constituent services.

Case State Model

The following diagram shows the lifecycle of a case through its various states, from creation to closure.

Figure 33-1 Case State Model



Creating a Case

A case can be added to an already existing BPM project that doesn't already include one, or you can create a new BPM project for a case.

You can define only one case per BPM project.

How to Create a Case

To create a case you must open or create a BPM project and then create the case. For more information on how to create a BPM project, see Creating and Working with Projects.

To create a case:

1. Select **File**.
2. Select **New**.
3. Select **BPM Tier**.
4. Select **Create Case Management**.

   The **Create Case Management** dialog box appears.
5. Enter a name to identify the case.
6. Optionally, enter a namespace.
7. Click **OK**.

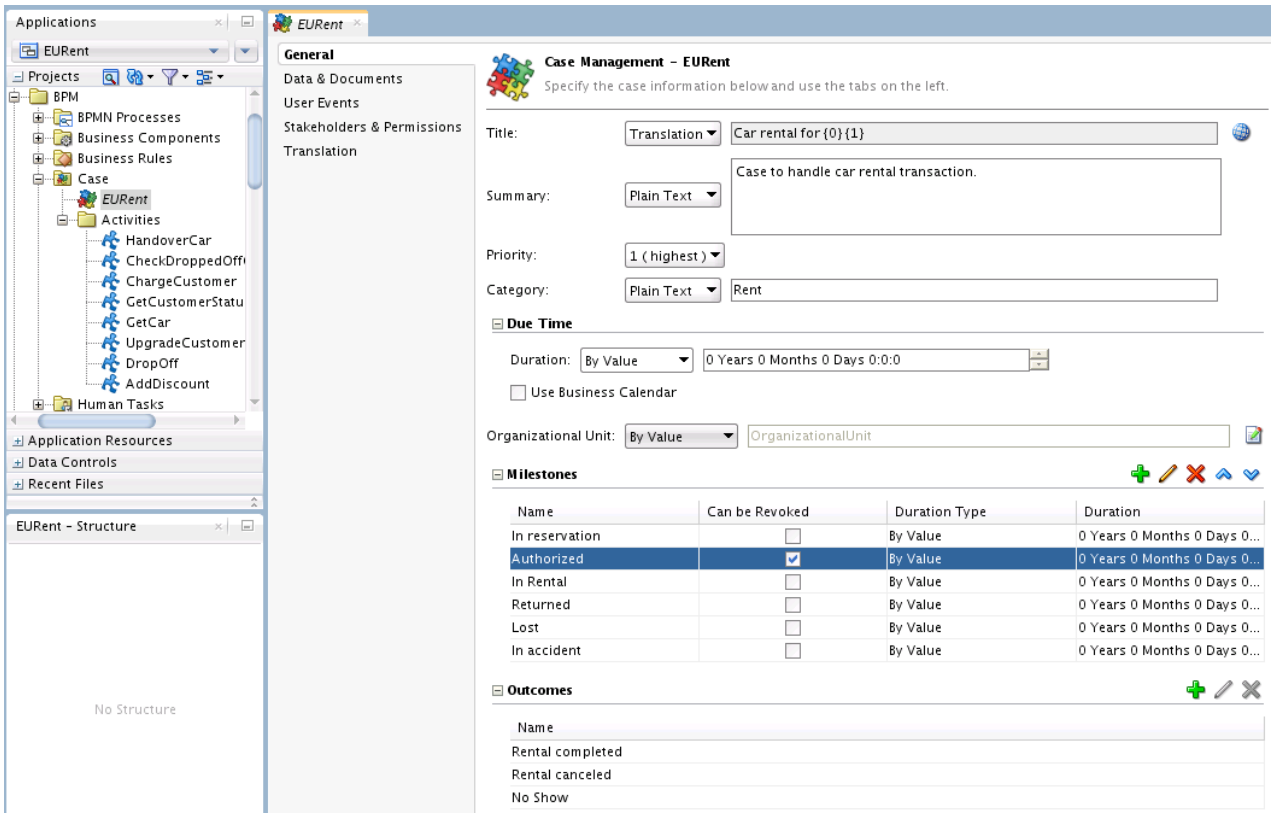
   BPM Studio creates the new case and displays it in the Case Management editor so that you can configure it.

Configuring a Case

Use the Case Management editor to configure cases.

Figure 33-2 shows the Case Management Editor.

Figure 33-2 Case Management Editor



How to Edit a Case

Edit cases using the Case Management editor. Newly created cases are displayed in the Case Management editor automatically. To edit an existing case, open the case file as described below.

To edit a case:

1. Select the **Application Navigator**.
2. Expand the project that contains the BPMN process.
3. Expand the **BPMN Content** folder.
4. Double click the *case file*.

   The case file has the .case extension. There is only one case file per BPM project.
5. Edit the case. See:
   - Configuring Case General Properties
   - How to Configure Case Data

- Configuring Case User Events
- How to Add Case Stakeholders
- Localizing a Case

Configuring Case General Properties

Use the General tab to configure the general properties of a case.

The General tab includes the following sections:

- **General properties** - use this section to configure properties for a case including Title, Summary, a text summary of what the case does, Priority, a value from 1 (high) through 5 (low), and a Category, set as Plain Text or Translation. See How to Configure the Case General Properties
- **Due Time** - specifies the case due date using the Duration value.

  The Duration can be expressed as a static value (select By Value) or as an XPath expression (select By Expression). The case due date is calculated from when a case starts.

  If Use Business Calendar is selected, and an organizational unit is specified (see below), the case due date is calculated using the business calendar associated with the organizational unit. Otherwise the normal calendar is used. See Case Deadlines.
- **Organizational Unit** - select an organizational unit to be associated with the case. It can be expressed as a static value (select By Value) or as an XPath expression (select By Expression). Only members of the organizational unit specified are able to access the case, even if they are also specified as stakeholders.
- **Milestones** - specifies milestones and their properties for the case.

  Milestones represent the completion of a deliverable or a set of related deliverables. They are checkpoints that indicate the progress of a case. Stakeholders can use them to obtain a high level idea of the status of the case.

  There is no direct activity or work associated with milestones.

  Use the Can Be Revoked checkbox to indicate that a milestone can be revoked. Only milestones that have been achieved can be revoked. This does not affect other achieved milestones for the case.

  The milestone Duration can be set using a value or an XPath expression. This is used to calculate the milestone deadline, based on when the milestone started. See Case Deadlines.

  Add, edit, delete, and re-order milestones using the controls in the panel. Use the panel to configure the Name, Can be Revoked value, Duration Type, and Duration for milestones. See How to Add Case Milestones.
- **Outcomes** - create, edit, and delete case outcomes in this section.

  Outcomes are user-defined values that are assigned to the case when it is completed. For example in a medical treatment case, possible outcomes might be: admitted, discharged, referred.

  Each outcome includes a name, and a display name. See How to Define Case Outcomes

Case Deadlines

Cases support two types of deadlines - case due dates, and milestone deadlines. Both of these are expressed as a duration, specified as either a value or an XPath expression. Durations are configured in the General tab in the Case Management editor.

Case due dates are calculated using the value specified in the Duration field in the Due Time panel, based on the starting date and time of the case. When the case due date is reached, a case deadline event is raised.

If Use Business Calendar is selected, and an organizational unit is specified, the case due date is calculated using the business calendar associated with the organizational unit. Otherwise the normal calendar is used.

Milestone deadlines are calculated using value specified in the Duration field for the milestone in the Milestones panel, based on the starting date and time of the milestone. If a milestone is still active when the milestone deadline is reached, a milestone deadline event is raised.

How to Configure the Case General Properties

Specify general case information using the General tab of the Case Management editor.

To configure the general properties:

1. Edit the case.
2. Select the **General** tab.
3. Provide a title for the case.
4. From the **Priority** list, select a priority.

   The value of the priority varies from 1 to 5, being 1 the highest and 5 the lowest.

   Priority indicates the importance of the case so that the case worker can prioritize their work.
5. Enter a category.

   Categories enable you to group similar cases together.

How to Add Case Milestones

Create milestones to track progress in cases. Do not include spaces in milestone names.

To add a milestone:

1. Edit the case.
2. Select the **General** tab.
3. Expand the **Milestones** section.
4. Click **Add**.

5. Enter the name of the milestone.
6. Click **OK**.

How to Define Case Outcomes

To define a case outcome

1. Edit the case.
2. Select the **General** tab.
3. Expand the **Outcomes** section.
4. Click **Add**.
5. Enter the name of the milestone.
6. Click **OK**.

Configuring Case Data and Documents

Data and document storage is configured in the **Data & Documents** tab in the Case Management editor.

Cases can be configured to store documents in an enterprise content manager.

Case Document Operations

A case can contain one or more related documents. Stakeholders can upload case documents that only other stakeholders with the appropriate permissions can view or delete.

To perform operations on documents, use the CaseStreamService as described in the Oracle Fusion Middleware Business Process Management Suite Java API Reference.

Specifying Permission Tags for Case Documents

You can configure who can read and update documents using permission tags. For more information about permission tags, see Defining Case Tag Permissions.

You can specify permission tags in the following situations:

- When creating a new document using the method `uploadDocument()` from the `CaseDocumentStreamService` class.
- By changing the permission on an existing document using the method `setPermissionTag()` from the `CaseService` class, passing the appropriate value in the permission tag parameter.

The support for permission tags on documents depends on the type of document store:

- Non Oracle Content Management Systems

  This feature is not supported for content management systems that are not Oracle WebCenter Content.

- BPM Database

If you use BPM DB as the document store, then you can set permission tags on case documents without having to configure anything. See Using the BPM Database for Data Storage.

- Oracle WebCenter Content

  When you set a permission tag for a document this value is stored in the metadata information field for the CaseManagementPermissionTag. You must create the CaseManagementPermissionTag information field before using permission tags on a document. To create the field, see Creating Case Fields in Oracle WebCenter Content. When you try to set a permission tag on an existing document, it fails.

Using the BPM Database for Data Storage

There are some differences in behavior when using the BPM database for data storage:

- If the parent root folder and the instance folder are not specified in the case design, the folder used to store documents is shown as a slash, similar to a root folder (for example, `/`).
- If the parent root folder and the instance folder are not specified in the case design, and while invoking the case, you override the use of the ecmFolder tag in the caseHeader payload (for example, `caseroot`), the folder used to store documents is shown as a folder under the root (for example, `/caseroot`).
- If the parent and instance folders are specified in the case design, the folder used to store documents is `/parentFolderName/InstanceFolderName`. However, if the parent and instance folders are overridden from the payload, the folder specified in the payload is shown.

Case Links in WebCenter Case Documents

Case documents stored in WebCenter can be include a web browser link to view the details of the originating case.

Before using this facility, you must create a custom attribute named **CaseManagementLink** in WebCenter. See Creating Case Fields in Oracle WebCenter Content.

When case documents are uploaded to WebCenter, the CaseManagementLink is populated.

Customizing Case Links in WebCenter Case Documents

The value of the CaseManagementLink property can be changed in Enterprise Manager in the Workflow property configuration to support different usages in custom Case Management user interfaces.

Placeholders can be used for live values to be substituted in when documents are stored and the CaseManagementLink property is populated. For example, $host could be used to represent the host name and $port could be used to represent the port number. These placeholder values must be defined in `mdm-url-resolver.xml`.

The values `$caseId` and `$caseNumber` will be replaced with their respective values without any additional configuration in `mdm-url-resolver.xml`.

Creating Case Fields in Oracle WebCenter Content

You must create the fields to support case information in Oracle WebCenter before using them. This applies to the CaseManagementPermissionTag and CaseManagement Link fields. See Specifying Permission Tags for Case Documents and Case Links in WebCenter Case Documents.

To create a field in Oracle WebCenter Content:

1. Start the **Configuration Manager** applet from the **Admin Applets** Page.
2. Click on the **Add…** button.
3. Specify the name in the **Metadata Field Name** field, for example, `CaseManagementPermissionTag` or `CaseManagementLink`.
4. Click **OK**.
5. Ensure that the **Require Value** checkbox is not selected, then click OK to keep the default values.

   The new information field appears.
6. Click **Update Database Design** to save the changes.

How to Configure Case Data

The data represents the payload of the case and defines the input parameters of the case. The data represents part of the information in the case.

Note:

Case data created based on a XSD or a business object is not initialized with the default values defined in the XSD or business object.

Note:

Basing case data on system schema types or on system types such as StartCaseInputMessage is not supported. This can cause corruption of the Adaptive Case Management project.

To configure the case data:

1. Edit the case.
2. Select the **Data & Documents** tab.
3. Expand the **Data** section.
4. Click **Add**.

   The Add Data Dialog appears.
5. Enter a name to identify the case data.

   The name is not unique. Different case data can have the same name.

6. Select an element or a type.

Note:

Case data does not support simple data types, thus they do not appear in the list.

Note:

Only the case or case activity forms data of the projects that are created in ADF appears in the list.

---

7. If you want make the case data editable, select the Editable checkbox.
8. Click **OK**.

Configuring Case Flex Fields

You can configure flex fields to map to case payload data. Use the **Flex Fields** section of the **Data & Documents** tab of the Case Management editor to create and edit flex field mappings that link flex field variables with data in the case.

Flex fields can be set to be unchangeable by clearing the Updatable checkbox to support data that does not change after creation, such as a serial number.

When case data is persisted, flex field mappings are checked and the linked data is also persisted.

You must map a flex field to a task field in the run-time task configuration as well as create the mapping to the case payload data.

How to Create a Case Flex Field

To create a case flex field mapping:

1. Edit the case.
2. Select the **Data & Documents** tab.
3. Expand the **Flex Fields** section.
4. Click **Add**.

   The Create Flex Field Dialog appears.
5. Enter appropriate information for the Name, Type, Column Name, and Source.

   The Type can be String, Number or Date.
6. Specify the Source. To use an XPath expression for the Source, click XPath to launch the Expression Builder dialog box.
7. If the flex field data can be changed after creation, select Updatable. For unchanging data, such as a Customer ID, or serial number, leave it unchecked.
8. Click **OK**.

How to Configure the Document Location

Use the **Documents** section of the **Data & Documents** tab of the Case Management editor to configure document locations. The document location is the folder in the enterprise content management system where all the documents related to the case instance are stored. This folder may contain other folders.

The case document folder name is created by concatenating the parent folder name and the case instance folder name you provide. You must provide a case instance folder name, or an exception is triggered at runtime.

To configure the documents location:

1. Edit the case.
2. Select the **Data & Documents** tab.
3. Expand the **Documents** section.
4. In the **Parent Folder** field, specify a base folder name for the case, using one of the following options:
   - Select **By Name** to provide the parent folder location using static text.
   - Select **By Expression** to provide the parent folder location using an XPATH expression.

   Note:

   The folder you provide must already exist.
5. In the **Case Instance Folder** field, specify a folder name for each case instance, using one of the following options:
   - Select **By Name** to provide the parent folder location using static text.
   - Select **By Expression** to provide the parent folder location using an XPATH expression.

   In general, this is done using an XPATH expression.

   Note:

   The folder you provide must already exist.
6. Optionally, select **Create Case Instance Folder** for Oracle BPM Suite to create a case folder in Enterprise Content Management store.

   Note: Ensure that you configure content management systems after creating and specifying the **Case Instance Folder** and **Parent Folder**. Case does not get created if CMS is not configured.
7. To add document properties, click Add in the **Document Properties** panel.

Specify the Name, Value type (By Value or By Expression), and the Value (either a value or an XPath expression) and click OK.

How to Configure Enterprise Content Management

By default Oracle BPM Suite is configured to use an Oracle Database document store. You can use the default configuration while developing your project if you do not have access to an enterprise content manager. This does not require any configuration in the Oracle SOA Server.

You can use the following enterprise content managers for storing case documents:

- Oracle WebCenter
- Alfresco CMIS

To use this content managers you must manually configure them using EM after installing BPM. For more information, see Administering Oracle SOA Suite and Oracle Business Process Management Suite.

The following list shows the configuration for the supported enterprise content managers:

- **Oracle WebCenter**

  The endpoint URL must have the following format `idc://ucm_host:4444`

  The administrator user can be `weblogic`

- **Alfresco CMIS**

  The endpoint URL must have the following format `http://alfresco_host:8080/alfresco/cmisatom`

  The administrator user can be `weblogic`

Configuring Case User Events

You can define custom user events that represent manual actions that occur while processing the case.

Case workers raise events to indicate that something occurred. The occurrence of an event may trigger the activation of a case activity or mark a milestone as completed. For example, if waiting for a fax is part of a case, when it arrives, the case worker can raise an appropriate event to indicate this has occurred.

How to Add User Events

To add an event:

1. Edit the case.
2. Select the **User Events** tab.
3. In the User Events section of the dialog, click **Add**.

   The Create User Event dialog box appears.
4. Provide a Name and a Display Name to identify the event.

5. Click **OK**.

   The new event appears in the Events section.

How to Publish Case User Events

The case management engine publishes events to Oracle EDN. These events capture system and user events in the case. System events include case lifecycle, case activity lifecycle, milestone, document, and comment related events.

You can design your process to listen to these events and react to them. For more information on how to use events in Oracle BPM, see How to Configure Your Process React to a Specific Signal.

To publish case events:

1. Open the case in the Case Management editor.
2. Select the **User Events** tab.
3. Select **Publish Case Events**.

Note:

The case event definition is available at `oramds:/soa/shared/casemgmt/CaseEvent.edl`

To view the case event schema, see CaseEvent.edl.

Defining Case Stakeholders and Permissions

Use the **Stakeholders & Permissions** tab of the Case Management editor to create, edit, and delete stakeholders and their associated permissions. The permission model enables you to define both stakeholder permissions and tag permissions.

You can define multiple stakeholders for each case you define. Only stakeholders can perform actions on case objects. Note that if an organizational unit is specified for the case, only members of the organizational unit are able to access the case, even if they are also specified as stakeholders.

Stakeholder Member Types can be configured to be Users, Groups, Application Roles, or Process Roles. The Value for these stakeholders are appropriate to the Member Type. For example, a User stakeholder might specify a particular user ID, whereas a Group stakeholder might specify a particular user group from your LDAP directory.

Stakeholder Values can be specified by providing a specific value, or by XPATH expression.

The administrator decides which actions each stakeholder can perform. By default, during deployment Oracle BPM grants stakeholders all the available permissions. After deployment the administrator can remove non-relevant permissions.

Note: **weblogic** stakeholder is a part of **BPMOrganizationAdmin** role and has all permissions. Any user who is part of this role has administrator privileges. **weblogic** stakeholder will be able to see the case and perform actions on the case even if the user is not added to the case.

Future redeployments of a case project may add new stakeholder application roles and new permission tag roles, but existing ones will not be affected if this happens. Undeploying a case does not affect any grants or application roles.

Table 33-2 shows the default permissions by case object.

Table 33-2 Permissions by Case Object

Number	Resource (Case Object)	Allowed Actions
1	CASE	READ, UPDATE
2	COMMENT	READ, UPDATE
3	DOCUMENT	READ, UPDATE
4	DATA	READ, UPDATE
5	EVENT	INVOKE
6	ACTIVITY	INVOKE
7	MILESTONE	READ, UPDATE
8	STAKEHOLDER	READ, UPDATE

Table 33-2 (Cont.) Permissions by Case Object

Number	Resource (Case Object)	Allowed Actions
9	HEADER	READ, UPDATE

Note: The READ permission for case header is not used anywhere, and changing the permission

How to Add Case Stakeholders

The stakeholders are the different persons involved in the processing of the case. They are case workers that can view the case and work on it.

Figure 33-3 shows the **Stakeholders and Permissions** tab in the Case Management Editor.

To add a stakeholder:

1. Edit the case.
2. Select the **Stakeholders and Permissions** tab.
3. Click **Add**.

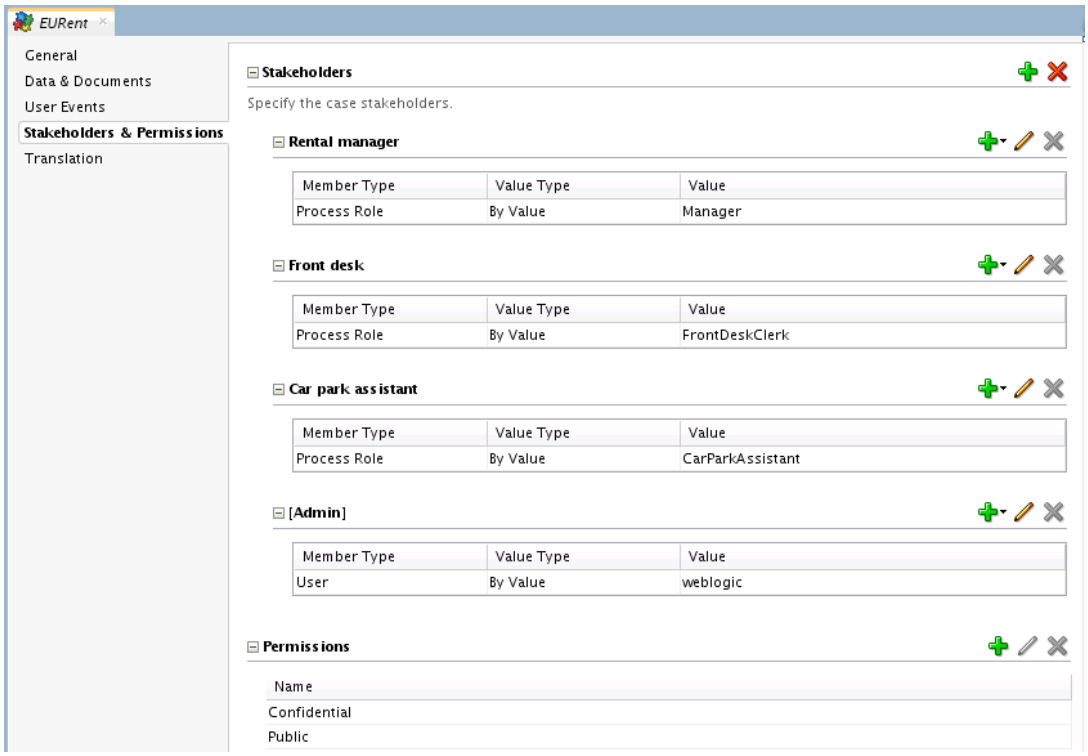
   The Create Stakeholder dialog box appears.
4. Enter a Name and a Display Name to identify the stakeholder and click **OK**.
5. Click Add for the newly created Stakeholder to add a member and select from Add User, Add Group, Add Application Role, or Add Process Role.
6. From the **Member Type** list, select a member type.
7. From the **Value Type** list, select how to define the value field - either By Value or By Expression.
8. Specify a Value, appropriate to the Value Type selection - either a static string or an XPath expression.

   Value specifies the actual user acting as a stakeholder. It specifies the actual user, group or role processing the case.

Note:

When you remove a case stakeholder definition, the underlying user, role, group or role in the organization is not removed.

Figure 33-3 Stakeholder Tab in Case Management Editor



How to Add Case Permissions

Use the Permissions section of the Stakeholders and Permissions tab of the Case Management editor to define permissions specific to a case.

You specify which users can update the case by tagging case objects with appropriate permission values. Only users with read/write OPSS permission can see or update case objects tagged with permissions.

You can attach permissions to case objects such as documents and data.

You can define your own set of permissions. The UI shows the default permissions PUBLIC and RESTRICTED. You can modify these default permissions.

Some examples of regularly used permissions are: internal, public, press release.

Note:

E-mail and simple workflow are global case activities thus their permission tag is global.

To add a permission:

1. Edit the case.
2. Select the **Stakeholders & Permissions** tab.
3. Expand the **Permissions** section.
4. Click **Add**.

5. Enter a Name and a Display Name to identify the permission.
6. Click **OK**.

How to Manage Case Permissions

You can manage the permissions assigned to each stakeholder using Oracle Enterprise Manager.

To manage permissions:

1. In Oracle Enterprise manager from **Weblogic Domain**, right click **soainfra**, then select **Security** and then select **Application Policies**.
2. In the **Application Policies** page, run a search with the following search criteria:
   a. In the **Application Stripe** field, enter *OracleBPMPRocessRolesApp*.
   b. In the **Principal Type** field, enter *Application Role*.
   c. In the **Name Starts With** field, enter the *name of the case* or leave it blank.
3. From the search result, select one of the application roles corresponding to the stakeholder whose permissions you want to edit.
4. Click the **Edit** button.

   The Edit Application Grant dialog box appears.
5. From the **Permissions** table, select a permission and click **Edit**.

Note:

Oracle Enterprise Manager does not validate action strings so you must provide the exact action string.

Note:

To assign multiple actions, separate them with commas without spaces. For example: READ,UPDATE.

Defining Case Tag Permissions

Stakeholders can assign additional permissions to case objects during runtime. For this option to be available, you must create permission tags when you design the case in Oracle BPM Studio.

For example, you can define a case with the following permission tags: PUBLIC, RESTRICTED, HIGHLY_CONFIDENTIAL.

During deployment Oracle Business Process Manager creates the application roles that correspond to the permission tags defined in the case.

For example, in a case named EURent, if you use the EURent.RESTRICTED.UPDATE action to grant a user the role EURent.RESTRICTED.UPDATE.Role, you can assign a document the RESTRICTED permission tag. Only users with the role EURent.RESTRICTED.READ.Role can access that document.

Note:

Tag permissions work together with stakeholder permissions. For example, to read or update a case object a stakeholder must have the READ/UPDATE permission and the case object must have the appropriate tags that allow reading or updating it.

For more information about stakeholder permissions, see Defining Case Stakeholders and Permissions.

How to Manage Case Tag Permissions

1. In Oracle Enterprise manager, from **Weblogic Domain**, right click **soainfra**, then select **Security** and then select **Application Policies**.
2. In the **Application Policies** page provide the following:
   a. In the **Application Stripe** field, enter *OracleBPMPRocessRolesApp*.
   b. In the **Name Starts With** field, enter the *name of the case* or leave it blank.
3. In the **Application Roles** page, select a permission tag role then click **Edit**.

   The Edit Application Role dialog box appears.
4. Click **Add** to add user, groups, or application role members to this application role.

Localizing a Case

You can configure a case to use different languages for display at run-time.

The following case artifacts can be localized:

- case title
- case category
- milestone name
- outcome
- data
- user event
- stakeholder
- permission

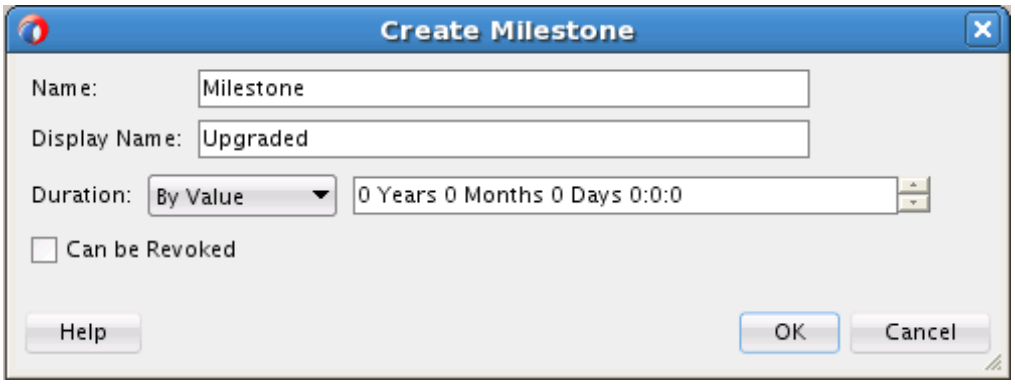
You can define display names for all the above artifacts except case title, and case categories. Display names are stored in the default locale resource bundle.

Note:

Multiple artifacts can be configured to have the same display name. However, it is a good practice to use unique display names that are descriptive and help the user quickly identify the displayed data.

Figure 33-4 shows the creation dialog box for a milestone that enables you to configure the display name.

Figure 33-4 Display Name Configuration When Creating a Milestone



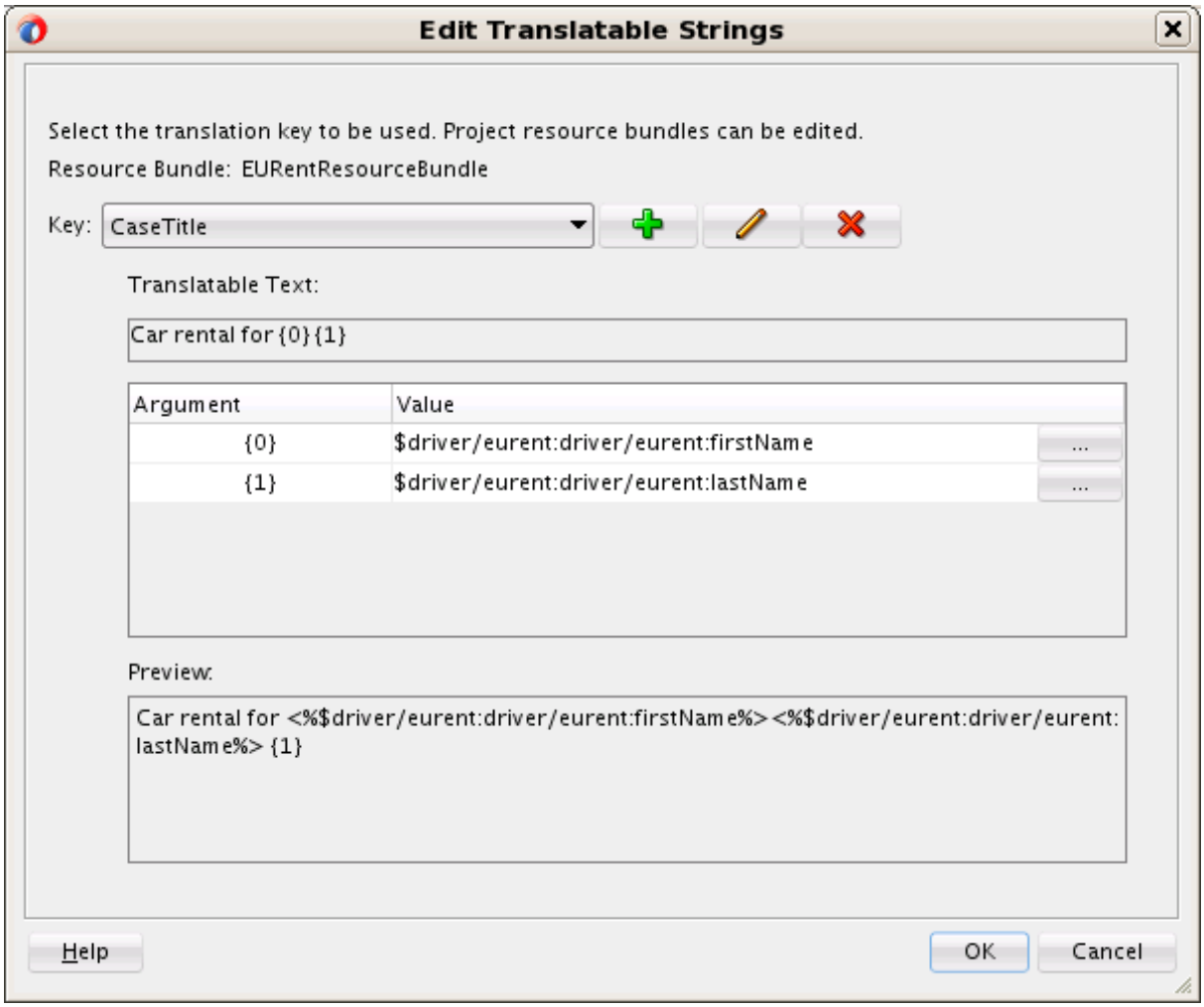
Case Title

You can specify the case title using plain text or the translation option. When you choose the translation option you must define the following:

- the key in the resource bundle
- the text to translate
- the parameters to make the title dynamic, if applicable

Figure 33-5 shows a title specified using the translation option. In this example the key in the resource bundle is CaseTitle. The text to be translated contains two parameters: Car rental for %0 %1. The parameter values are specified in the Argument table.

Figure 33-5 Localizing the Case Title



Case Category

You can specify the case category using plain text or the translation option. The translation option only supports a simple translation string.

How to Configure Case Localization

You can specify and localize each of the keys defined in a case.

To add a localization key:

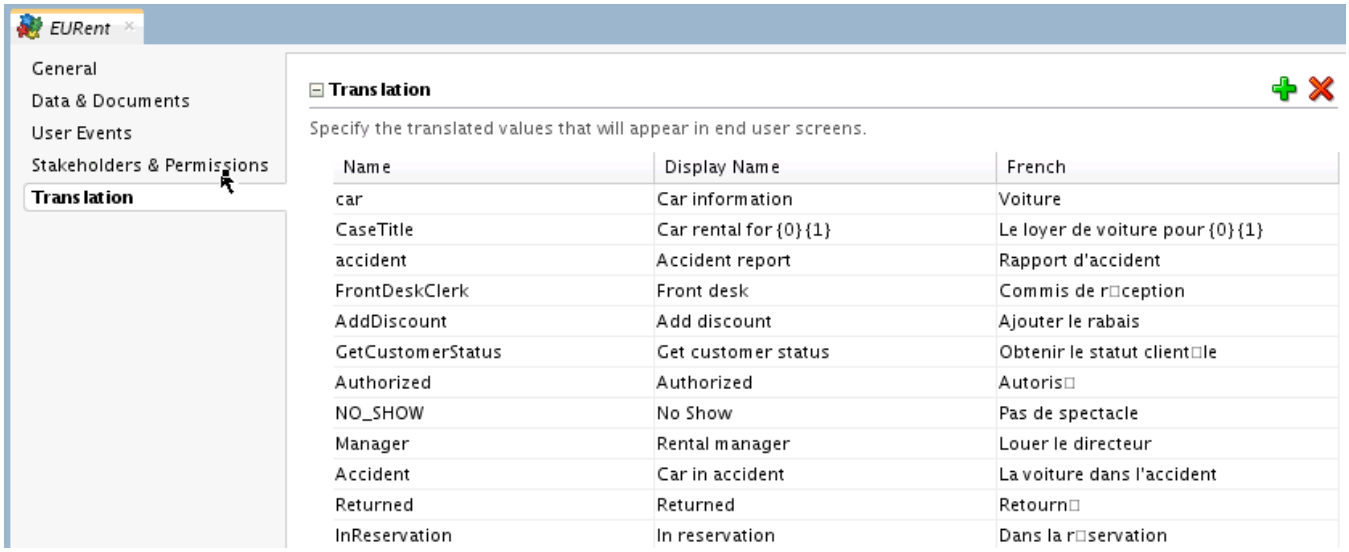
1. Edit the case.
2. Select the **Translation** tab.
3. Expand the **Translation** section.

   The Translation editor appears.

   Figure 33-6 shows the Translation editor.
4. Click Add.
5. Enter a name to identify the key.

6. Enter a value for the default language.
7. Enter a translation for the specified languages.

Figure 33-6 Translation Editor



Localizing Case Objects

The following classes are types of the class CaseObject:

- CaseData
- CaseDocumentObject
- CaseEvent
- CaseHeader
- CaseMilestone
- CaseStakeHolder
- Comment
- DatabaseDocument

The class CaseObject contains the following attributes that are shared with its types:

- Id
- CaseId
- ObjectName
- ObjectDisplayName
- ObjectType
- UpdatedBy
- UpdatedByDisplayName

- UpdatedDate
- PermissionTag

From this list, ObjectDisplayName contains the localized value of ObjectName, and UpdatedByDisplayName contains the localized value of UpdatedBy

Note:

Initially UpdateBy and UpdatedByDisplayName contain the name of the user that created the CaseObject. After the user updates the case, these fields contain the name of the user that last updated the case.

Case Activities and Sub Cases

Case activities model tasks that can be executed by the end user as part of a case. Case activities can be human tasks, BPMN processes or a custom case activities. A case activity represents a specific piece of work that case workers must perform.

Case activities can model Sub Cases. See Creating Sub Cases.

Cases are composed of various case activities. You can create these case activities based on a BPM process, or a Human Task, or you can create a custom case activity.

You can create case activities by:

- promoting a BPMN process to a case activity

  See How to Promote a BPMN Process to a Case Activity.

- promoting a Human Task to a case activity

  See How to Promote a Human Task to a Case Activity.

- creating a custom case activity

  See How to Create a Custom Case Activity.

Case Activity and Sub Case Attributes

Case activities and sub cases are defined by the following attributes:

- Manually Activated/Automatically Activated

  Manually activated case activities are available in the case and can be invoked by users. Automatically activated case activities are invoked by the system.

- Required

  Required activities must be invoked at least once before a case is closed.

- Repeatable

  Repeatable case activities can be invoked more than once in a case. Non repeatable case activities can be invoked only once. Already invoked manual case activities do not appear in the library.

- Conditionally Available

  Conditionally available manual activities are available in the library if they are activated through a business rule. Non-conditionally available manual activities are available in the library by default until you invoke them.

After invocation, repeatable activities are still shown in the library. Non conditional automatic activities are invoked after Oracle BPM starts a case.

- Permission

  The Permission value specifies access to the case activity, for example, PUBLIC or PRIVATE.

You can specify input and output parameters for case activities. For additional information, see How to Add a Case Activity Input Parameter and How to Add a Case Activity Output Parameter.

Note that sub cases do not support data mapping - they inherit data only from their parent case. See Creating Sub Cases

Predefined Case Activities

By default the cases contain the following case activities:

- Simple Workflow

  You can use this activity to create simple human tasks. You can configure the task title, priority, due date, comment, assignment type and assignees. The supported assignment types are: simple, sequential, parallel, FYI.

  You must pass the payload input to the method `initiateCaseActivity` with the key `SimpleWFActivityPayload`. You must use a payload from the schema in Simple Workflow Payload Schema.

- Email Notifications

  You can use this activity to send an email notification. You must configure the workflow notification with EMAIL for email notifications to work properly.

  You must pass the payload input to the method `initiateCaseActivity` with the key `EmailActivityPayload`. You must use a payload from the schema in Email Notification Payload Schema.

Note:

You can only send case documents as attachments in global case activity email notification activities. Documents located in your file system are not supported.

Specifying the Order of Case Activities

If you want a case activity to run after another case activity, then you must define a condition in the second activity that indicates it runs after the first activity completes.

For more information on defining conditions, see Using Business Rules with Cases.

How to Promote a BPMN Process to a Case Activity

You can create a case activity based on a BPMN process. This exposes the BPMN process as a case activity. A case activity is one single item that the case worker can perform.

When you create a case activity Oracle BPM Studio generates output arguments for all the arguments in the multiple end points of the BPMN process. However if you want

to cover the input arguments of a process that contains multiple start points, then you must create a case activity for each of the start points.

The BPMN process must already exist. You can use synchronous and asynchronous BPMN processes. The case activity only supports message start and end points.

To promote a BPMN process to case activity:

1. Select the **Application Navigator**.
2. Expand the project that contains the BPMN process.
3. Expand the **BPMN Content** folder.
4. Expand the **processes** folder.
5. Right-click the *BPMN process* you want to promote to case activity.
6. Select **Promote as Case Activity**.

   The Create Case Activity dialog box appears.
7. Enter a name to identify the case activity.
8. Enter a display name to show in the case user interface.
9. From the **Operation Name** list, select a start operation.
10. To make the case activity synchronous, select the **Synchronous** check box.
11. Click **OK**.

    BPM Studio creates the case activity and opens it in the Case Activity editor so that you can configure it.

    By default Oracle BPM Studio creates the input and output parameters based on the BPMN process input and output arguments. If the BPMN process contains multiple end points, then it creates an output parameter for each of the output arguments of the multiple end points.

Note:

The case activity can pass input parameters to the underlying BPMN process. You can also configure the case activity to read the output arguments of the BPMN process and store their value.

For more information, see Defining Input Parameters for Case Activitiesand Defining Output Parameters for Case Activities.

How to View the BPMN Process

The Advanced tab shows the service reference and the operation used to create the case activity.

To view the human task operation:

1. Open the *case*.

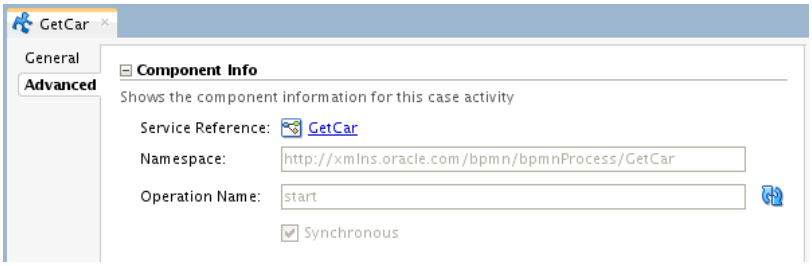
2. Select the **Advanced** tab.

   Figure 33-7 shows the Advanced tab for a BPMN process based case activity.

3. To open the BPMN process, click the *service reference*.

4. To change the operation, click the **Refresh** button.

Figure 33-7 Advanced Tab of a BPMN Process Based Case Activity



How to Promote a Human Task to a Case Activity

You can create case activities based on a Human Task.

The Human Task must already exist.

To promote a BPMN process to case activity:

1. Select the **Application Navigator**.
2. Expand the project that contains the Human Task.
3. Expand the **SOA Content** folder.
4. Right-click the *Human Task* you want to promote to case activity.
5. Select **Promote as Case Activity**.

   The Create Case Activity dialog box appears.

6. Enter a name to identify the case activity.
7. Enter a display name to show in the case user interface.
8. Click **OK**.

   BPM Studio creates the case activity and opens it in the Case Activity editor so that you can configure it.

   By default Oracle BPM Studio creates the input and output parameters based on the Human Task payload arguments.

Note:

The case activity can pass input parameters to the underlying Human Task. You can also configure the case activity to read the output parameters of the Human Task and store their value.

For more information, see Defining Input Parameters for Case Activitiesand Defining Output Parameters for Case Activities.

How to View the Human Task

The Advanced tab shows the Human Task used to create the case activity.

To view the human task operation:

1. Open the *case*.
2. Select the **Advanced** tab.
3. To open the Human Task, click the *service reference*.

How to Create a Custom Case Activity

You can create a custom case activity based on a Java class.

Custom case activities enable users to create their own case activities, for example a scheduler. To the end user there is no difference from the other types of case activities.

To create a case activity:

1. Select **File**.
2. Select **New**.

   The New Gallery dialog box appears.
3. Select **BPM Tier**.
4. Select **Custom Case Activity**.

   The **Create Custom Case Activity** dialog box appears.
5. Enter a name to identify the case activity.
6. Enter a display name to show in the case user interface.
7. In the Class Name field, enter the name of the Java callback class.

   The Java class must implement the `oracle.bpm.casemgmt.caseactivity.ICaseActivityCallback` interface.

   The callback class must be part of the composite, or must add it to the workflow customization classpath. For more information, see Developing Business Processes with Oracle Business Process Composer.
8. Click **OK**.

   BPM Studio creates the case activity and opens it in the Case Activity editor so that you can configure it.

Note:

You can add input and output parameters to a custom case activity. You can assign input parameters the value of case data or user input. You can choose to store the value of output parameters as case data.

Creating Sub Cases

Use sub cases to manage activities that contribute to the resolution of a parent case. Sub cases are similar to case activities, are instantiated at runtime within the context of a parent case.

For example, a power outage case might have a sub case for a line repair. As the parent power outrage case evolves one or a number of line repair sub cases might be kicked off to facilitate needed repairs that are discovered as the parent case is worked.

Sub cases are deployed as a separate composite from the parent case project. They are linked to the parent using the case link mechanism. The sub case composite version is always the active version of the parent composite.

Sub cases do not support data mapping - they inherit data only from their parent case.

When a sub case completes, an activity completion event is raised. The same the constraints that apply to case activities also apply to sub case activities.

How to Create a Sub Case

To create a Sub Case:

1. Select **File**.
2. Select **New from Gallery**.

   The New Gallery dialog box appears.
3. Select **BPM Tier**.
4. Select **Sub Case**.

   The **Create Case Activity** dialog box appears.
5. Specify the Name, Project Name, and Case Name values.
6. Click **OK**.

   BPM Studio creates the sub case activity and opens it in the Case Activity editor. See Case Activity and Sub Case Attributes.

Defining Input Parameters for Case Activities

Input parameters can be case data or user input.

By default, input parameters take their values from case data. You can change this so that they take their values from user input.

You must also define the case data in the .case file. If you do not define the case data, Oracle BPM Studio creates a new case data of the required type when you create the case activity.

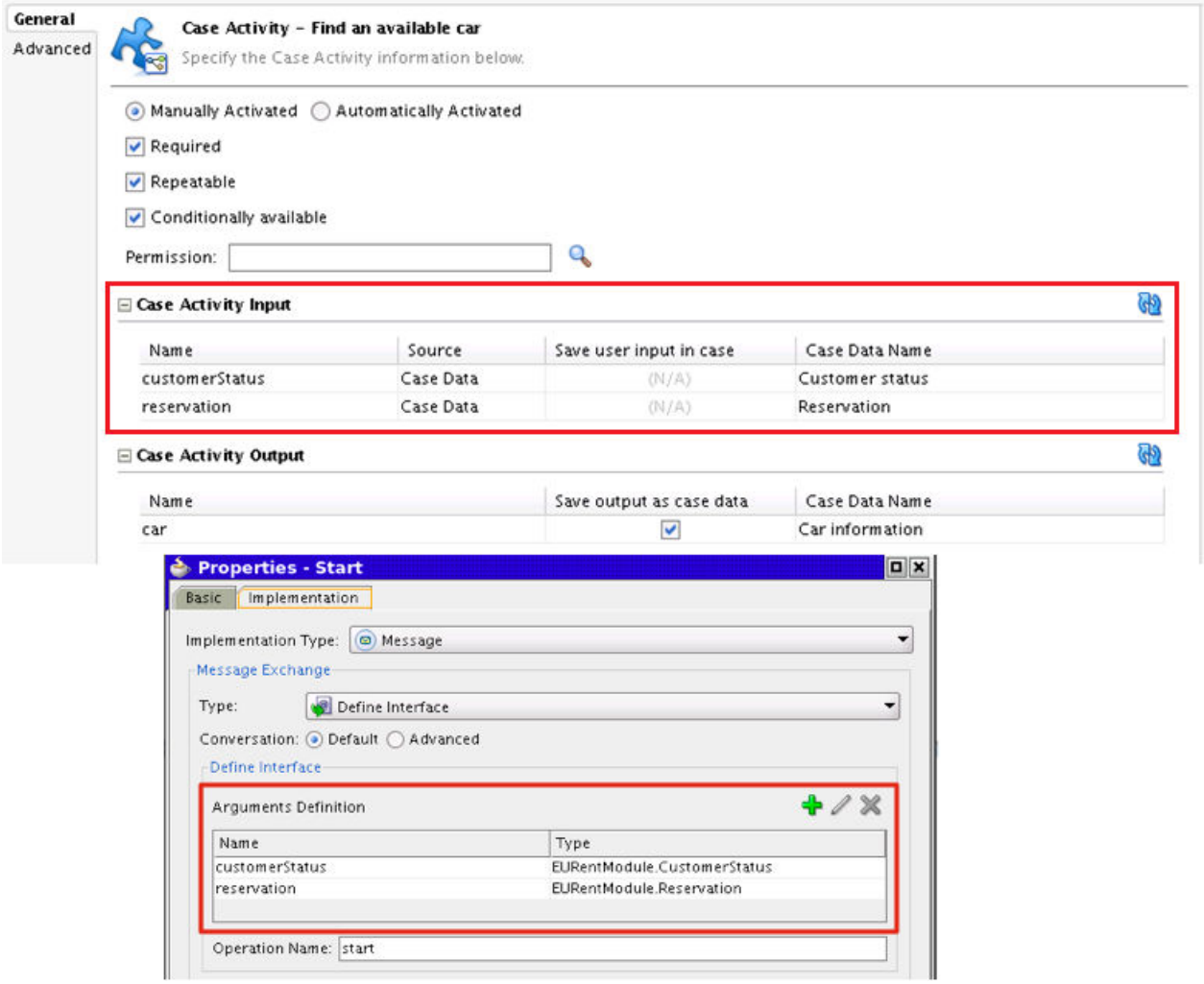
You must define case activity input parameters in the same order of the BPMN process or Human Task input arguments.

If the input parameter is of the type user input, you can save this value as case data.

Figure 33-8 shows the input arguments of a BPMN process based case activity input parameters and the arguments of the start event of a BPMN process. Note that the

name of the input parameters of the case activity matches the name of the arguments in the BPMN process.

Figure 33-8 BPMN Process Based Case Activity Input Parameters



How to Add a Case Activity Input Parameter

You can define the input parameters for a case activity.

To add a case activity input parameter:

1. Expand the **Case Activity Input** section.
2. Click **Add**.
3. Enter a name to identify the parameter.
4. Optionally select the **Store Data** option.

Defining Output Parameters for Case Activities

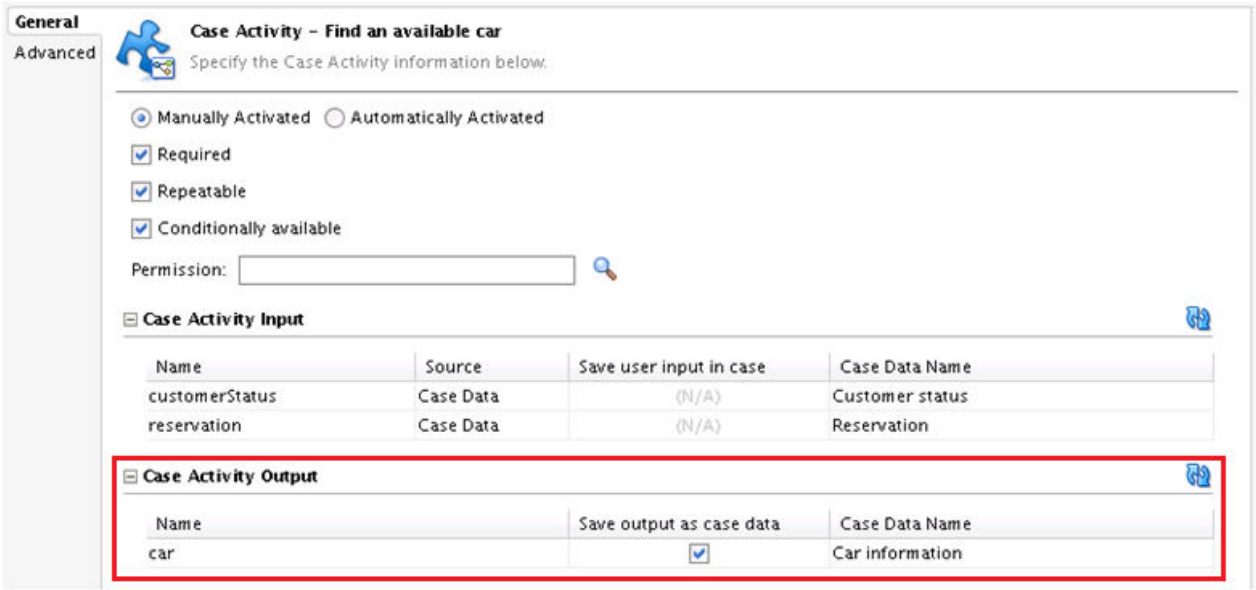
By default Oracle BPM Studio creates the output arguments based on the BPMN process or Human Task arguments. Only editable human workflow arguments appear as output arguments in a case activity.

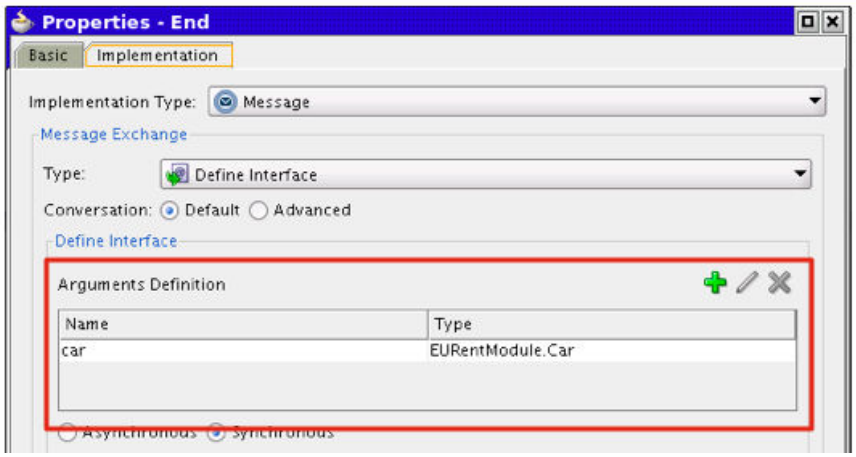
You can save the value of output parameters as case data.To do this, the name of the case activity output parameter must match the root element name of the BPMN process or Human Task argument. After you create the case, you can change the name of the output parameters.

Figure 33-9 shows the output arguments of a BPMN process based case activity output parameters and the arguments of the end event of a BPMN process. Note that the name of the output parameters of the case activity matches the name of the arguments in the BPMN process.

You can save the output as case data. By default the Case Activity editor populates the case data fields, if a case data of the same type is not available. Otherwise the Case Activity editor creates a new case data of the type in the .case file.

Figure 33-9 BPMN Process Based Case Activity Output Parameters





How to Add a Case Activity Output Parameter

You can define the output parameters for a case activity.

To add an output parameter:

1. Expand the **Output** section.
2. Click **Add**.
3. Enter a name to identify the parameter.
4. Enter a namespace.
5. Optionally select the **Store Data** option.

Configuring Case Activities

After you create a case activity BPM Studio opens the case in the Case Activity editor for you to configure it.

You can configure a case activity to behave in different ways during the case workflow by configuring the case activity properties.

How to Edit a Case Activity

To edit a case activity:

1. Select the **Application Navigator**.
2. Expand the project that contains the BPMN process.
3. Expand the **BPMN Content** folder.
4. Double click the case activity file.

   The case file has the .caseactivity extension.

   The Case Activity editor appears.
5. The Case Activity editor allows you to configure the following:
   - Configuring Case Activity Basic Properties
   - How to Add a Case Activity Input Parameter
   - How to Add a Case Activity Output Parameter

Configuring Case Activity Basic Properties

You can configure the following basic properties for the case activity you created:

- Automatic
- Required
- Repeatable

- Conditionally Available

For a detailed description of these attributes, see Case Activity and Sub Case Attributes.

You can also add input and output parameters for the case activity. See How to Add a Case Activity Input Parameter and How to Add a Case Activity Output Parameter.

Creating a Global Case Activity

Global case activities are custom case activities that are global in scope and not part of any composite. They apply to all cases regardless of their type.

They are identified by global flag true. You cannot design global activities using Oracle BPM Studio.

You must add the callback Java class for a global activity to the workflow customization classpath.

Example of Global Case Activity Metadata Schemashows an example of the metadata for a global activity metadata. Note that the value of the isGlobal attribute is set to true.

After creating the global case activity you must register it using the registerCaseActivity class from the Oracle Fusion Middleware Business Process Management Suite Java API Reference. To do this you must unmarshall the case activity document and pass the CaseActivity as a parameter.

Example 33-1 shows how to register a global case activity.

Example 33-1 Registering a Global Case Activity

```
InputStream  is = classLoader.getResourceAsStream(<file>);

public static CaseActivity unmarshall(InputStream inputStream)
throws JAXBException, IOException {
       try {
           // create a document
           DOMParser p = new DOMParser();
           p.retainCDATASection(true);
           p.parse(inputStream);
           Document    doc = p.getDocument();

           JAXBContext jaxbContext = JAXBContext.newInstance(JAXB_CONTEXT);

           //return unmarshal(doc);
           return (CaseActivity) jaxbContext.createUnmarshaller().unmarshal(doc);

       } catch (oracle.xml.parser.v2.XMLParseException e) {
           throw new JAXBException(e);
       } catch (org.xml.sax.SAXException e) {
           throw new JAXBException(e);
       }
}

private static final String JAXB_CONTEXT =
"oracle.bpm.casemgmt.metadata.activity.model";
```

Using Business Rules with Cases

You can use business rules to decide which case activities to activate for automatic or manual initiation, or to withdraw manual case activities. You can also use rules to mark a milestone as achieved or revoked.

When you create a case, Oracle BPM Studio automatically generates an associated business rule dictionary.

It is a good practice to define case management rules on events. Case management rules are fired on an event in rules. Hence it is advisable to define rules which happen on an event instead of a condition.

Oracle BPM fires business rules on every case event. Case events are logical events that occur while running the case. The following list enumerates the available case events:

- Life cycle events
- Milestone events
- Activity events
- Data events
- Document events
- Comment events
- User events

Defining the Condition of a Case Business Rule

You can define the condition of the business rule base on the following:

- The event that fired the business rule

  Table 33-3 describes the different events that can fire a business rule.

- The case instance
- Case data

  The case data configured in the case is available as facts in the business rule dictionary. You can create rules based on case data combined with case management related facts.

Table 33-3 Case Events

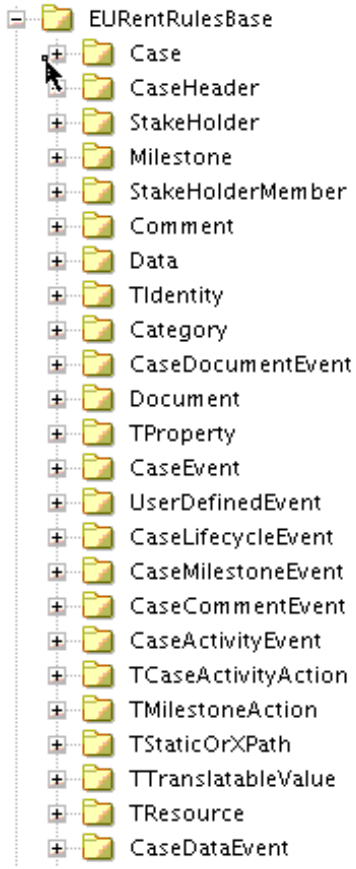
Name	Description	Attributes
CaseLifeCycleEvent	Life cycle event	state, lifecycleEvent
CaseMilestoneEvent	Milestone event	milestone, type
CaseActivityEvent	Activity event	activityName, type
CaseDataEvent	Data event	dataName
CaseDocumentEvent	Document event	document, documentName, type

Table 33-3 (Cont.) Case Events

Name	Description	Attributes
CaseCommentEvent	Comment event	comment
UserDefinedEvent	User event	eventName, event

Figure 33-10 shows the facts you can use to define the condition of a business rule based on a case Management system related data.

Figure 33-10 Business rule facts



Understanding the Case Business Rule Dictionary

The business rule dictionary created when you create the case is linked to a common base dictionary in Oracle MDS. The common base dictionary includes all the facts show in Figure 33-10. The base dictionary name is `CaseManagementBaseDictionary`.

The business rule dictionary of a case supports the following operations:

- Automatically invoke conditional automatic activities from a business rule
- Publish conditional manual activities to the case from a business rule
- Withdraw an activity from a business rule

- Achieve and revoke milestones from a business rule

For a detailed description of the functions used to perform these operations, see Table 33-4.

How to Generate a Case Business Rule Dictionary

When you create a case, Oracle BPM automatically generates an associated business rule dictionary. This case business rule dictionary enables you to define business rules with rule conditions based on the case.

To generate a case business rule dictionary:

1. Create a case.

   For information on how to create a case, see How to Create a Case.

   The case rule dictionary appears.

2. Open the case business rule dictionary.

3. Create business rules according to your business requirements.

   Table 33-4 shows the different functions you can use when creating the business rule conditions.

Table 33-4 Rule Functions

Rule Function	Description	Parameters
`activateActivity(String activityName)`	Invokes conditional automatic case activities and conditional manual case activities.	**activityName:** the name of the activity to invoke.
`withdrawActivity(String activityName)`	Withdraws a case activity.	**activityName:** the name of the activity to withdraw.
`setActivityRelevance(String activityName, String comments, tActivityRelevance relevance, tCaseActivityStatus caseAction)`	Rates a case activity, including a reason for the rating.	**activityName:** the name of the activity to withdraw. **comments:** reason for the rating. **relevance:** rating of the case activity (NONE, LOW, NORMAL or HIGH) **caseAction:** populated by the function on return

Table 33-4 (Cont.) Rule Functions

Rule Function	Description	Parameters
`setActivityRepeatability(String activityName, Bool value, tCaseActivityStatus caseAction)`	Sets the repeatability (true, false) of a case activity.	**activityName:** the name of the case activity. **value:** TRUE - case activity can be repeated, FALSE - case activity can not be repeated. **caseAction:** populated by the function on return
`setActivityRequirement(String activityName, Bool value, tCaseActivityStatus caseAction)`	Sets whether a case activity is required or not.	**activityName:** the name of the case activity. **value:** TRUE - required case activity, FALSE, not required **caseAction:** populated by the function on return
`setActivityInitiationModel(String activityName, Bool value, tCaseActivityStatus caseAction)`	Sets whether a case activity is automatically initiated or not.	**activityName:** the name of the case activity. **value:** TRUE - initiate automatically, FALSE, initiate manually **caseAction:** populated by the function on return
`reachMilestone(String milestoneName, String comments)`	Marks a milestone as achieved.	**milestoneName:** the name of the milestone to mark as achieved. **comments:** a comment stating the reason to mark this milestone as achieved.
`revokeMilestone(String milestoneName, String comments)`	Revokes a milestone.	**milestoneName:** the name of the milestone to revoke. **comments:** a comment stating the reason to revoke this milestone.

Closing Cases

Any stakeholder can close a case, if all required activities in the case are completed. Users with additional privileges can force closure of a case even if there are pending required activities.

Closing a case is a logical operation that marks its status as closed. You can close a case by invoking the `closeCase` method in the `CaseInstanceService` class. You can provide an optional outcome parameter and a comment when you close a case.

You can close a case regardless of its current state and the state of its case activities.

Closing a case sets it state to CLOSED. The list of cases for a user that you obtain using the queryCase API it includes closed cases.

Note:

You can still achieve and revoke milestones after you close or suspend a case.

Integrating with Oracle BPM

You can integrate with Oracle BPM by invoking a case from a BPMN process or by publishing case events to Oracle EDN.

You then create a process that reacts to these events.

Invoking a Case From a BPMN Process

You can invoke a case from BPMN process.

To invoke a case from a BPMN process:

1. Add a service task to the BPM process.
2. Right-click the service task.
3. Select **Properties**.
4. Click the **Implementation** tab.
5. In the **Service Call** section, click the browse button next to the **Service** field.
6. Select the case.
7. Select an operation from the **Operation** list.

   Available operations are: abortCase, closeCase, reopenCase, suspendCase, resumeCase, attainMilestone, revokeMilestone
8. Using data associations, assign a value to the attributes case ID and comment. In the case of milestone operations, also assign a value to the milestone attribute.

   For more information about data associations, see Introduction to the Data Association Editor.

Note:

The case ID is available as a predefined variable that is automatically assigned a value when you invoke a BPMN process from a case.

How to Use Correlations with Case Events

If you want to use correlations with a particular event, then you can trigger a BPMN process from a BPMN based case activity. You must pass the caseId to the message that initiates the process and use it as a correlation key.

To use correlations with case events:

1. Create a BPMN process that contains a send task that triggers another process.
2. Edit the data association of the process you created to assign the value of the predefined variable caseId to the argument of the send task.
3. Promote the BPMN process to a case activity.
4. Create a BPMN process that listens to the correlated events.

   You can start this process with a message start event and use a message catch to receive the correlated event.
5. Edit the properties of the message start event and define an argument CaseId.
6. Edit the correlation definition and configure it to use the argument CaseId. Select the initiates option.
7. Edit the properties of the message catch event and configure the correlation definition to use the CaseId variable.

Schema Reference

This section contains the simple workflow schema, the email notification schema, and an example of a global case activity metadata schema.

- Simple Workflow Payload Schema

  This schema contains the list of payloads that you can pass to the method initiateCaseActivity to create a simple human task. For more information, see Predefined Case Activities.
- Email Notification Payload Schema

  This schema contains the list of payloads that you can pass to the method initiateCaseActivity to send an email notification. For more information, see Predefined Case Activities.
- Example of Global Case Activity Metadata Schema

  This schema is an example of the metadata for a global activity. For more information on global activities, see Creating a Global Case Activity.
- CaseEvent.edl

  This schema defines the case events that you can publish to Oracle EDN. For more information, see How to Publish Case User Events.

Simple Workflow Payload Schema

Example 33-2 Simple Workflow Payload Schema

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:extension="http://xmlns.oracle.com/bpm/case/metadata/extension"
            xmlns="http://xmlns.oracle.com/bpm/case/activity/custom"
            targetNamespace="http://xmlns.oracle.com/bpm/case/activity/custom"
            xmlns:jaxb="http://java.sun.com/xml/ns/jaxb" jaxb:version="2.0"
            elementFormDefault="qualified" blockDefault="#all">
 <xsd:annotation>
  <xsd:documentation>Simple WF Activity Schema</xsd:documentation>
```

```
  <xsd:appinfo>
   <jaxb:schemaBindings>
    <jaxb:package name="oracle.bpm.casemgmt.customactivity.simplewf.model"/>
   </jaxb:schemaBindings>
   <jaxb:globalBindings generateElementClass="true" generateIsSetMethod="true">
    <jaxb:serializable uid="123456"/>
   </jaxb:globalBindings>
  </xsd:appinfo>
 </xsd:annotation>
 <xsd:import namespace="http://xmlns.oracle.com/bpm/case/metadata/extension"
             schemaLocation="ExtensibleElements.xsd"/>

<xsd:element name="SimpleWorkflowPayload" type="tSimpleWorkflowPayload"/>

<xsd:complexType name="tSimpleWorkflowPayload">
  <xsd:complexContent>
    <xsd:extension base="extension:tExtensibleElements">
       <xsd:sequence>
         <xsd:choice minOccurs="1" maxOccurs="1">
           <xsd:element name="simpleAssignmentType" type="tSimpleAssignmentType"/>
           <xsd:element name="sequentialAssignmentType" type="tSequentialAssignmentType"/>
           <xsd:element name="parallelAssignmentType" type="tParallelAssignmentType"/>
           <xsd:element name="fyiAssignmentType" type="tFyiAssignmentType"/>
         </xsd:choice>
         <xsd:element name="title" type="xsd:string" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="dueDate" type="xsd:date" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="priority" type="xsd:integer" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="comment" type="xsd:string" minOccurs="0" maxOccurs="1"/>
         <xsd:element name="payloadType" type="tPayloadType" minOccurs="0" maxOccurs="1"/>
       </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tSimpleAssignmentType">
   <xsd:complexContent>
    <xsd:extension base="extension:tExtensibleElements">
      <xsd:sequence>
         <xsd:element name="assigneeType" type="tAssigneeType" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tSequentialAssignmentType">
    <xsd:complexContent>
     <xsd:extension base="extension:tExtensibleElements">
       <xsd:sequence>
          <xsd:element name="assigneeType" type="tAssigneeType" minOccurs="1" maxOccurs="unbounded"/>
       </xsd:sequence>
     </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tFyiAssignmentType">
    <xsd:complexContent>
     <xsd:extension base="extension:tExtensibleElements">
        <xsd:sequence>
             <xsd:element name="assigneeType" type="tAssigneeType" minOccurs="1"
maxOccurs="unbounded"/>
        </xsd:sequence>
```

```
     </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tParallelAssignmentType">
    <xsd:complexContent>
     <xsd:extension base="extension:tExtensibleElements">
        <xsd:sequence>
          <xsd:element name="assigneeType" type="tAssigneeType" minOccurs="1" maxOccurs="unbounded"/>
          <xsd:element name="defaultOutcome" type="tOutcomeTypeEnum" minOccurs="1" maxOccurs="1"/>
          <xsd:element name="completionCriteriaType" type="tCompletionCriteriaType" minOccurs="1"
maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="waitForAllVotes" type="xsd:boolean"/>
      </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tCompletionCriteriaType">
 <xsd:complexContent>
  <xsd:extension base="extension:tExtensibleElements">
   <xsd:sequence>
      <xsd:element name="outcome" type="tOutcomeTypeEnum" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="outcomePercentage" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
   </xsd:sequence>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tOutcomeTypeEnum">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="APPROVE"/>
        <xsd:enumeration value="REJECT"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="tAssigneeType">
    <xsd:complexContent>
     <xsd:extension base="extension:tExtensibleElements">
       <xsd:sequence>
           <xsd:element name="identityName" type="xsd:string" minOccurs="1" maxOccurs="1"/>
           <xsd:element name="identityType" type="tIdentityTypeEnum" minOccurs="1" maxOccurs="1"/>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="tIdentityTypeEnum">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="USER"/>
      <xsd:enumeration value="GROUP"/>
      <xsd:enumeration value="APPROLE"/>
    </xsd:restriction>
</xsd:simpleType>


<xsd:complexType name="tPayloadType">
    <xsd:complexContent>
     <xsd:extension base="extension:tExtensibleElements">
       <xsd:sequence>
           <xsd:element name="parameter" type="tParameterType" minOccurs="1" maxOccurs="unbounded"/>
```

```
      </xsd:sequence>
     </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="tParameterType">
    <xsd:complexContent>
     <xsd:extension base="extension:tExtensibleElements">
       <xsd:sequence>
           <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
           <xsd:element name="value" type="xsd:string" minOccurs="1" maxOccurs="1"/>
       </xsd:sequence>
     </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

</xsd:schema>
```

Email Notification Payload Schema

Example 33-3 Email Notification Payload Schema

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:extension="http://xmlns.oracle.com/bpm/case/metadata/extension"
            xmlns="http://xmlns.oracle.com/bpm/case/activity/custom"
            targetNamespace="http://xmlns.oracle.com/bpm/case/activity/custom"
            xmlns:jaxb="http://java.sun.com/xml/ns/jaxb" jaxb:version="2.0"
            elementFormDefault="qualified" blockDefault="#all">
 <xsd:annotation>
  <xsd:documentation>Email Activity Schema</xsd:documentation>
  <xsd:appinfo>
   <jaxb:schemaBindings>
    <jaxb:package name="oracle.bpm.casemgmt.customactivity.notification.model"/>
   </jaxb:schemaBindings>
   <jaxb:globalBindings generateElementClass="true" generateIsSetMethod="true">
    <jaxb:serializable uid="123456"/>
   </jaxb:globalBindings>
  </xsd:appinfo>
 </xsd:annotation>
 <xsd:import namespace="http://xmlns.oracle.com/bpm/case/metadata/extension"
             schemaLocation="ExtensibleElements.xsd"/>

 <xsd:element name="emailPayload" type="tEmailPayload"/>
  <xsd:complexType name="tEmailPayload">
   <xsd:complexContent>
    <xsd:extension base="extension:tExtensibleElements">
     <xsd:sequence>
       <xsd:element name="from" type="xsd:string" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="to" type="xsd:string" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="replyTo" type="xsd:string" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="cc" type="xsd:string" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="bcc" type="xsd:string" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="ject" type="xsd:string" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="message" type="xsd:string" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="attachments" type="tAttachment" minOccurs="0" maxOccurs="unbounded"/>
     </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
```

```
 <xsd:complexType name="tAttachment">
  <xsd:complexContent>
   <xsd:extension base="extension:tExtensibleElements">
    <xsd:sequence>
       <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
       <xsd:element name="mimeType" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:schema>
```

Example of Global Case Activity Metadata Schema

Example 33-4 Example of Global Case Activity Metadata Schema

```
<caseActivity targetNamespace="http://xmlns.oracle.com/bpm/case/activity"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://xmlns.oracle.com/bpm/case/activity ../../../../../
interface/src/main/resources/schemas/CaseActivity.xsd"
              xmlns="http://xmlns.oracle.com/bpm/case/activity">
 <documentation xmlns="http://xmlns.oracle.com/bpm/case/metadata/extension"/>
 <name>SimpleWorkflowActivity</name>
 <activityDefinitionId>http://xmlns.oracle.com/bpm/case/activity/SimpleWFActivityDefinition</
activityDefinitionId>
 <activityType>CUSTOM</activityType>
 <repeatable>true</repeatable>
 <required>false</required>
 <manual>true</manual>
 <isGlobal>true</isGlobal>
 <isConditional>false</isConditional>
 <caseAssociations>
   <documentation xmlns="http://xmlns.oracle.com/bpm/case/metadata/extension"/>
   <allCases/>
  </caseAssociations>
 <globalActivity>
  <definition>
   <documentation xmlns="http://xmlns.oracle.com/bpm/case/metadata/extension"/>
   <className>oracle.bpm.casemgmt.customactivity.simplewf.SimpleWFActivityCallback</className>
  </definition>
 </globalActivity>
</caseActivity>
```

CaseEvent.edl

Example 33-5 CaseEvent.edl

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions xmlns="http://schemas.oracle.com/events/edl" targetNamespace="http://
xmlns.oracle.com/bpm/case/event">
    <schema-import location="oramds:/soa/shared/casemgmt/CaseEvent.xsd"
namespace="http://xmlns.oracle.com/bpm/case/event"/>
    <event-definition name="CaseEvent">
        <content xmlns:ns0="http://xmlns.oracle.com/bpm/case/event"
element="ns0:caseEvent"/>
    </event-definition>
</definitions>
```

Part VII

Appendices

This part contains appendices that describe administrative features of Oracle BPM.

This part contains the following appendices:

- Process Star Schema Views

A

Process Star Schema Views

Process Star schema data objects are created in BAM 12c. You can use Oracle SQL to access Process Star schema data objects by their database view synonyms. You can then use the views to integrate a Process Star schema with any external business intelligence tool. You can use the BAM 12c composer to browse the individual columns of the Process Star schema data objects.

This appendix contains these topics:

- Standard Data Objects
- Composite-Specific Data Objects

Standard Data Objects

The following table lists each of the standard physical data objects and its corresponding database view synonym name.

Table A-1 Standard Data Objects to DB View Synonym Name Mapping

Sno	Display Name	Internal Name	DB View Synonym Name	Star Schema Member Type
1	oracle/processanalytics/internal/physical/Composite Definition (physical)	ORACLE_PROCESSANALYTICS_COMPOSITE_DEFINITION	BPM_PV_COMPOSITE_DEFINITION_V	Dimension
2	oracle/processanalytics/internal/physical/Process Definition (physical)	ORACLE_PROCESSANALYTICS_PROCESS_DEFINITION	M_PV_PROCESS_DEFINITION_V	Dimension
3	oracle/processanalytics/internal/physical/Activity Definition (physical)	ORACLE_PROCESSANALYTICS_ACTIVITY_DEFINITIO	BPM_PV_ACTIVITY_DEFINITION_V	Dimension
4	oracle/processanalytics/internal/physical/Role Definition (physical)	ORACLE_PROCESSANALYTICS_ROLE_DEFINITION	PM_PV_ROLE_DEFINITION_V	Dimension
5	oracle/processanalytics/internal/physical/CM Case Activity Definition	ORACLE_PROCESSANALYTICS_CASE_ACTIVITY_DEF	BPM_PV_CASE_ACTIVITY_DEF_V	Dimension

Table A-1 (Cont.) Standard Data Objects to DB View Synonym Name Mapping

Sno	Display Name	Internal Name	DB View Synonym Name	Star Schema Member Type
6	oracle/processanalytics/internal/physical/CM Case Definition	ORACLE_PROCESSANALYTICS_CASE_DEFINITION	N/A	Dimension
7	oracle/processanalytics/internal/physical/HWF Task Definition	ORACLE_PROCESSANALYTICS_TASK_DEFINITION	BPM_PV_HWF_TASK_DEFINITION_V	Dimension
8	oracle/processanalytics/internal/physical/Process (physical)	ORACLE_PROCESSANALYTICS_PROCESS	BPM_PV_PROCESS_V	Fact
9	oracle/processanalytics/internal/physical/Activity (physical)	ORACLE_PROCESSANALYTICS_ACTIVITY	BPM_PV_ACTIVITY_V	Fact
10	oracle/processanalytics/internal/physical/HWF Assignment (physical)	ORACLE_PROCESSANALYTICS_ASSIGNMENT	BPM_PV_HWF_ASSIGNMENT_V	Fact
11	oracle/processanalytics/internal/physical/HWF Task (physical)	ORACLE_PROCESSANALYTICS_TASK	BPM_PV_HWF_TASK_V	Fact
12	oracle/processanalytics/internal/physical/CM Case	ORACLE_PROCESSANALYTICS_CASE	BPM_PV_CASE_V	Fact
13	Oracle/processanalytics/internal/physical/CM Case Activity	ORACLE_PROCESSANALYTICS_CASE_ACTIVITY	BPM_PV_CASE_ACTIVITY_V	Fact
14	oracle/processanalytics/internal/physical/CM Case Stake Holder	ORACLE_PROCESSANALYTICS_CASE_STAKE_HOLDER	BPM_PV_CASE_STAKE_HOLDER_V	Fact
15	oracle/processanalytics/internal/physical/CM Case Stake Holder Member	ORACLE_PROCESSANALYTICS_CASE_STAKE_HOLDER_MEMBER	BPM_PV_CASE_STAKE_HOLDER_MEM_V	Fact

Composite-Specific Data Objects

The following table lists each of the composite-specific physical data objects and its corresponding database view synonym name. If you enable the BAM 12c analytics data target, then composite-specific data objects are created when the composite is deployed.

Note:

<In the following table, <COMPOSITENAME> is the composite name truncated to 70 chars, and <IDENTIFIER> is the analytics view identifier specified for the composite.

The maximum is 10 chars. No database view synonyms are created when the analytics view identifier is not specified for a composite.

Table A-2 Composite-Specific Data Objects to DB View Synonym Mapping

Sno	Display Name	Internal Name	DB View Synonym Name	Star Schema Member Type
1	oracle/processanalytics/internal/physical/<COMPOSITE NAME> Process (physical)	ORACLE_PROCESSANALYTICS_<COMPOSITENAME>_PROCESS	BPM_PV_PRCS_<IDENTIFIER>_V	Fact
2	oracle/processanalytics/internal/physical/<COMPOSITE NAME> Activity (physical	ORACLE_PROCESSANALYTICS_<COMPOSITENAME>_ACTIVITY	BPM_PV_ACTV_<IDENTIFIER>_V	Fact