Oracle® Fusion Middleware

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

11*g* Release 1 (11.1.1.7)

E15177-09

February 2013

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

ORACLE

Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management, 11*g* Release 1 (11.1.1.7)

E15177-09

Primary Author: Oracle Corporation

Contents

Preface xix

Intended Audience xix
Documentation Accessibility xix
Related Documents xix
Conventions xx

What's New in This Guide for Release 11.1.1.7 xxi

1 Oracle Business Process Management Suite Overview

1.1 Introduction to the Oracle Business Process Management Suite 1-1
1.2 Oracle BPM User Personas 1-2
1.3 Oracle BPM Suite Components 1-3
1.3.1 Process Modeling and Implementation 1-4
1.3.1.1 Oracle BPM Studio 1-4
1.3.1.2 Oracle Business Process Composer 1-4
1.3.1.3 Oracle Metadata Service (MDS) Repository 1-5
1.3.1.4 Oracle BPM Projects 1-5
1.3.2 Oracle BPM Runtime Components 1-5
1.3.2.1 Oracle BPM Engine 1-5
1.3.2.2 Oracle Human Workflow 1-6
1.3.2.3 Oracle Business Rules 1-6
1.3.2.4 Oracle WebLogic Application Server 1-6
1.3.2.5 Oracle Enterprise Manager 1-6
1.3.3 Oracle BPM Suite Process Participant Applications 1-6
1.3.3.1 Oracle Business Process Management Workspace (Process Workspace) 1-7

1.3.3.2 Oracle Business Process Management Process Spaces (Process Spaces) 1-7
1.3.4 Other Oracle BPM Suite Components 1-7
1.3.4.1 Process Analytics 1-7
1.3.4.2 Guided Business Processes 1-7
1.4 Oracle Business Process Analysis (BPA) Suite 1-7
1.5 Introduction to the Application Development Life Cycle 1-8
1.5.1 Process Modeling 1-9
1.5.2 Implementation 1-10
1.5.3 Deployment 1-10
1.5.4 Oracle BPM Runtime 1-11
1.6 Oracle BPM Use Cases 1-11
1.6.1 Use Case: Using Business Process Composer to Create Projects 1-11
1.6.2 Use Case: Using BPM Studio to Create Project Templates 1-12
1.6.3 Use Case: Using BPM Studio to Model Processes and Deploy an Application 1-12
1.6.4 Use Case: Using The Oracle Business Process Analysis Suite to Model Your Business Processes 1-13
1.7 Accessibility Options 1-13
1.7.1 Enabling Accessibility Features in SOA Composer 1-13
1.7.2 Enabling Accessibility Features in BPM Worklist 1-14

Part I Getting Started with Oracle Business Process Composer

2 Introduction to Oracle Business Process Composer

2.1 Introduction to Oracle Business Process Composer 2-1
2.1.1 Process Design and Implementation 2-2
2.1.2 Collaboration 2-2
2.1.3 Simulation and Testing 2-3
2.1.4 Deployment 2-3
2.2 Introduction to Business Process Composer Use Cases 2-4
2.2.1 Use Case: Creating a New Oracle BPM Application 2-4
2.2.2 Use Case: Create an Oracle BPM Application Based on a Project Templates 2-7
2.3 Signing On to Oracle Business Process Composer 2-8
2.4 Introduction to the Business Process Composer User Interface 2-9
2.4.1 Introduction to the Business Process Composer Toolbar 2-9
2.4.2 Introduction to the Business Process Composer Welcome Page 2-10
2.4.2.1 Project Views 2-11
2.4.2.2 Project Browser 2-11
2.4.2.3 Control Panel 2-11
2.4.2.4 Search 2-11
2.4.3 Introduction to the Business Process Composer Main Menu 2-11
2.4.4 Introduction to the Project Welcome Page 2-13

3 Walkthrough: Creating a Basic BPM Application

3.1 Setting Up a BPM Project 3-1
3.1.1 Login to Business Process Composer 3-1
3.1.2 Create and Open a BPM Project 3-2
3.1.3 Create a BPMN Process 3-4

3.1.4 Save Your BPM Project ... 3-4
3.2 Modeling a BPMN Process ... 3-4
3.2.1 Open the Business Process ... 3-5
3.2.2 Add a User Task ... 3-5
3.2.3 Save Your Project ... 3-6
3.2.4 Create a New Role ... 3-6
3.2.5 Add a Business Rule Task ... 3-7
3.2.6 Add a New Swimlane and Create the Manager Role ... 3-8
3.2.7 Add an Exclusive Gateway ... 3-9
3.2.8 Add the Approve Request User Task ... 3-9
3.2.9 Add an Additional Exclusive Gateway ... 3-10
3.2.10 Add a Service Task to Save the Travel Request in a Database ... 3-11
3.2.11 Add a Sequence Flow to Handle Rejected Travel Requests ... 3-11
3.2.12 Edit and Add Sequence Flows for Default Approval ... 3-12
3.2.13 Create a Project Snapshot ... 3-13
3.3 Implementing a BPMN Process - Defining User Interaction ... 3-14
3.3.1 Create and Open a Human Task ... 3-14
3.3.2 Create a Web Form ... 3-15
3.3.3 Add Basic Controls to Your Web Form ... 3-15
3.3.4 Add a Tabs Control to Your Web Form ... 3-16
3.3.5 Add Additional Controls to Each Tab ... 3-17
3.3.6 Test Your Web Form ... 3-18
3.3.7 Add a Form Rule to Calculate the Total Expense ... 3-19
3.3.8 Add a Form Rule to Add Dynamic Behavior to Your Form ... 3-19
3.3.9 Add a Form Rule to Dynamically Populate a List of Options ... 3-19
3.4 Implementing a BPMN Process - Creating a Business Rule ... 3-20
3.4.1 Create Business Objects ... 3-20
3.4.2 Create a New Business Rule and Define Input and Output Data Objects ... 3-21
3.4.3 Create and Configure a Bucketset ... 3-22
3.4.4 Create and Configure a Decision Table ... 3-23
3.4.5 Assign the Business Rule to the Business Rule Task ... 3-24
3.5 Implementing a BPMN Process - Defining a Service ... 3-24
3.6 Defining the Data Used by Your Process ... 3-25
3.6.1 Create Data Objects ... 3-25
3.6.2 Define Expressions and Conditions ... 3-26
3.6.3 Define Data Associations ... 3-27

Part II Modeling and Testing Business Processes

4 Working with BPM Projects

4.1 Introduction to Oracle BPM Projects ... 4-1
4.1.1 Introduction to Project Components and Resources ... 4-1
4.1.1.1 Editable Project Resources ... 4-2
4.1.1.2 The Business Catalog ... 4-3
4.1.1.3 Business Catalog Components that Can Be Edited or Created ... 4-4
4.1.2 Introduction to the Oracle BPM Repository ... 4-4
4.2 Introduction to the Project Welcome Page ... 4-4

4.2.1 The Project Information Panel 4-5
4.2.2 The Recent Activity Panel 4-5
4.2.3 Introduction to the Project Component Pane 4-6
4.2.4 Introduction to the Quickstart Menu 4-6
4.2.5 Introduction to the Oracle Business Process Composer Editors 4-7
4.2.5.1 Process Editor 4-7
4.2.5.2 Activity Guide Editor 4-7
4.2.5.3 Human Task Editor 4-7
4.2.5.4 Business Rules Editor 4-7
4.2.5.5 Data Associations Editor 4-8
4.2.5.6 Expression Editor 4-8
4.2.6 Introduction to the Supporting Browsers and Editors 4-8
4.2.6.1 Project and Process Validation Browser 4-8
4.2.6.2 Documentation Editor 4-8
4.2.6.3 Approval Workflow Browser 4-8
4.3 Introduction to Project Sharing and Collaboration 4-8
4.3.1 Private and Public Projects 4-8
4.3.2 Edit Mode 4-9
4.3.3 Project Roles 4-9
4.4 Creating and Working with Projects 4-9
4.4.1 How to Access the Project Welcome Page 4-9
4.4.2 How to Create a New Project 4-10
4.4.3 How to Open a Project Using the Application Welcome Page 4-10
4.4.4 How to Open a Project Using the Main Menu 4-11
4.4.5 How to Share a Project with Other Users 4-11
4.4.6 How to Edit a Shared Project 4-12
4.4.7 How to Save Changes to a Project 4-12
4.4.8 How to Discard Changes to a Project 4-12
4.4.9 How to Validate a Project 4-12
4.4.10 How to Close a Project 4-13
4.4.11 How to View the History of Changes Made to a Project 4-13
4.4.12 How to View and Edit Project Properties 4-13
4.4.13 How to Mark a Project as a Favorite 4-14
4.5 Using Guided Business Processes to Create Project Milestones 4-14
4.5.1 Introduction to Guided Business Processes 4-14
4.5.1.1 Introduction to Activity Guides and Milestones 4-14
4.5.2 How to Configure the Activity Guide and Create Project Milestones 4-15
4.5.3 How to Generate a Process Report for Your Project 4-16
4.6 Defining the Roles Used in a Project 4-16
4.6.1 Introduction to Project Roles 4-16
4.6.2 How to Create Project Roles 4-16

5 Working with Processes and the Process Editor

5.1 Introduction to Business Processes 5-1
5.2 Introduction to the Process Editor 5-2
5.2.1 Introduction to the Process Editor Toolbar 5-3
5.2.2 Introduction to the Process Editor Canvas 5-4

5.2.3 Introduction to the BPMN Component Palette 5-4
5.2.4 Introduction to the Business Catalog 5-5
5.3 Working with Business Processes 5-6
5.3.1 How to Create a New Business Process 5-6
5.3.2 How to Open a Business Process 5-6
5.3.3 How to Delete a Business Process 5-6
5.3.3.1 What You Need to Know About Deleting a Business Process 5-6
5.4 Working with Flow Elements 5-7
5.4.1 How to Add a Flow Object from the Component Palette 5-7
5.4.2 How to Cut, Copy, or Delete a Flow Object 5-7
5.4.3 How to Paste a Flow Object in a Process 5-8
5.4.4 How to Add a Sequence Flow to a Process 5-8
5.4.5 How to Delete a Sequence Flow 5-8
5.4.5.1 What You Need to Know About Deleting a Sequence Flow 5-8
5.4.6 How to Edit the Properties of a Flow Object 5-8
5.4.7 How to Assign a Custom Icon to a Flow Object 5-9
5.5 Working with Business Catalog Components 5-9
5.5.1 How to Assign a Business Catalog Component to a Flow Object 5-9
5.5.2 How to Create New Human Tasks in the Business Catalog 5-9
5.6 Working with Draft Processes 5-10
5.6.1 Introduction to Draft Processes 5-10
5.6.2 How to Mark a Flow Object as Draft 5-10
5.7 Documenting Your Process 5-10
5.7.1 Introduction to the Documentation Editor 5-11
5.7.1.1 Inserting Links in Your Documentation 5-11
5.7.2 How to Add Documentation to Your Process 5-11
5.7.3 How to Add Notes to a Process 5-12
5.8 Importing and Exporting Process Models 5-12
5.8.1 Importing Process Models into Oracle BPM 5-12
5.8.2 Exporting BPMN Processes to Oracle Tutor 5-13

6 Simulating Process Behavior

6.1 Introduction to Simulations 6-1
6.1.1 Simulation Models and Simulation Definitions 6-1
6.1.2 Simulation Parameters 6-2
6.1.2.1 General Simulation Definition Parameters 6-2
6.1.2.2 Simulation Model Parameters 6-2
6.1.2.3 Resource Parameters 6-3
6.1.2.4 Start Event Parameters 6-3
6.1.2.5 Activity Parameters 6-4
6.2 Creating and Running a Simulation 6-5
6.3 Working with Simulation Definitions 6-6
6.3.1 How to Create a Simulation Definition 6-6
6.3.2 What Happens When You Create a Simulation Definition 6-8
6.3.3 How to Edit a Simulation Definition 6-9
6.3.4 How to Add a Simulation Model to a Simulation Definition 6-10
6.4 Working with Simulation Models 6-10

6.4.1 How to Create a New Simulation Model 6-11
6.4.2 How to Edit a Simulation Model 6-11
6.5 Running Simulations 6-12
6.5.1 How to Run a Simulation 6-12
6.5.2 What Happens When You Run a Simulation 6-13
6.6 Analyzing the Results of a Simulation 6-13
6.6.1 How to Analyze the Results of a Simulation Using a Chart 6-13

7 Using Process Player

7.1 Introduction to Process Player 7-1
7.1.1 How Process Player Handles the Flow Objects of Your Process 7-2
7.1.2 Enabling Process Player in Business Process Composer 7-3
7.2 Using Process Player to Test the Behavior of Business Processes 7-3
7.2.1 How to Access Process Player 7-3
7.2.2 How to Map the Roles Defined in Your Process to Users in Your Organization 7-4
7.2.3 How to Use Process Player to Run a Business Process 7-4

8 Working with the Project Life Cycle

8.1 Importing and Exporting Projects 8-1
8.1.1 How to Import a Project from Your Local File System 8-1
8.1.2 How to Export a Project to Your Local File System 8-2
8.2 Introduction to BPM Project Templates 8-2
8.2.1 Introduction to Edit Policies 8-3
8.2.1.1 Process Level Edit Policies 8-3
8.2.1.2 Component Level Edit Policies 8-4
8.2.2 About Using Data Objects and Variables in Project Templates 8-4
8.3 Working with Project Templates 8-4
8.4 Working with Project Snapshots 8-5
8.4.1 How to Create a New Project Snapshot 8-5
8.4.2 How to View the Contents of a Project Snapshot 8-5
8.4.3 How to Return to the Active Version of a Project 8-5
8.4.4 How to Delete a Project Snapshot 8-5
8.4.5 How to Export a Project Snapshot 8-6
8.4.6 How to Deploy a Project Snapshot 8-6

Part III Defining How Users Interact with Your Business Processes

9 Working with Web Forms

9.1 Introduction to Forms in Oracle BPM 9-1
9.1.1 Introduction to Web Forms 9-2
9.1.2 Form First and Data First Design 9-2
9.2 Introduction to the Web Forms Designer 9-3
9.2.1 Introduction to the Web Forms Component Palette 9-4
9.2.2 Introduction to the Web Form Editor Toolbar 9-4
9.2.3 Introduction to the Property Editor 9-4
9.2.4 Introduction to the Data Source Panel 9-5

9.2.5 Introduction to the Form Canvas 9-6
9.3 Introduction to Web Form Controls 9-6
9.3.1 Input Controls 9-6
9.3.1.1 Text 9-7
9.3.1.2 TextArea 9-7
9.3.1.3 Date, Time, Date/Time 9-7
9.3.1.4 Email 9-9
9.3.1.5 Money 9-9
9.3.1.6 Phone 9-9
9.3.1.7 Quantity 9-9
9.3.1.8 Number 9-9
9.3.2 Selection Controls 9-9
9.3.2.1 Dropdown 9-10
9.3.2.2 Radio 9-10
9.3.2.3 Checkbox 9-10
9.3.2.4 BooleanCheckbox 9-10
9.3.3 Group Controls 9-10
9.3.3.1 Sections 9-10
9.3.3.2 Tabs 9-11
9.3.3.3 Panels 9-11
9.3.3.4 Tables 9-12
9.3.3.5 Repeats 9-12
9.3.4 Other Controls 9-13
9.3.4.1 Message Control 9-13
9.3.4.2 Link Control 9-14
9.3.4.3 Button Control 9-14
9.3.4.4 Image Control 9-14
9.4 Introduction to Data Sources 9-14
9.4.1 Web Form Controls Generated by Payload Data Types 9-15
9.4.2 Modifying Web Form Controls Generated From Data Elements 9-15
9.4.2.1 What You Can Edit Using the Web Form Editor 9-15
9.4.2.2 What You Cannot Edit Using the Web Form Designer 9-16
9.4.3 Introduction to the Display As Property 9-16
9.5 Walkthrough: Creating a Web Form Using the Form First Method 9-16
9.6 Walkthrough: Creating a Web Form Using the Data First Method 9-17
9.7 Working with Web Forms 9-19
9.7.1 How to Add Controls to a Web Form 9-19
9.7.2 How to Add Controls Based on Data Sources 9-19
9.7.3 How to Show Which Web Controls Were Created from a Data Source 9-20
9.7.4 How to Edit the Properties of a Web Form and Web Form Controls 9-20
9.7.5 How to Delete a Web Form 9-21
9.7.6 How to Remove a Control from a Web Form 9-21
9.7.7 How to Test a Web Form 9-22

10 Working with Web Form Rules

10.1 Introduction to Form Rules 10-1
10.1.1 Form Rule Javascript Syntax 10-1

10.1.1.1 Control Name ... 10-2
10.1.1.2 Form Rule Identifiers ... 10-3
10.1.1.3 Strings and Numbers ... 10-4
10.1.1.4 Writing Conditions ... 10-5
10.1.1.5 Select Controls ... 10-5
10.1.1.6 Initial Control State ... 10-5
10.1.1.7 Form Rules and Repeating Controls ... 10-6
10.1.2 Using Dynamic Content in Form Rules ... 10-6
10.1.2.1 Dynamic Content ... 10-6
10.1.2.2 Reusing Dynamic Content ... 10-7
10.1.3 Using Data and Built-in Methods in a Form Rule ... 10-8
10.1.3.1 Built-in Data ... 10-8
10.1.3.2 Built-in Methods ... 10-8
10.1.4 Understanding How Form Rules Work at Runtime ... 10-9
10.1.4.1 When are Form Rules Executed? ... 10-9
10.1.4.2 Infinite Loops ... 10-9
10.1.5 Debugging Form Rules ... 10-10
10.1.5.1 **Debugging Duplicate Control Names** ... 10-10
10.1.5.2 Form Rule Profiling ... 10-10
10.2 Working with Form Rules ... 10-10
10.2.1 How to Create a Form Rule ... 10-10
10.2.2 How to Test a Form Rule ... 10-11

11 Working with Human Tasks

11.1 Introduction to Human Tasks ... 11-1
11.1.1 Introduction to Participant and Routing Types ... 11-2
11.1.1.1 Participant Types ... 11-2
11.1.1.2 Routing Types ... 11-3
11.1.1.3 Outcome ... 11-4
11.1.2 Introduction to Participant Assignment ... 11-4
11.1.3 Introduction to Duration ... 11-4
11.2 Introduction to the Human Task Editor ... 11-5
11.3 Working with Human Tasks ... 11-5
11.3.1 Walkthrough: Creating and Configuring a Human Task ... 11-5
11.3.2 How to Create New Human Task ... 11-6
11.3.3 How to Open a Human Task ... 11-6
11.3.4 How to Configure Basic Task Properties ... 11-6
11.3.5 How to Configure the Deadline (Duration) for a Human Task ... 11-7
11.3.6 How to Change the Default Participant ... 11-8
11.3.7 How to Add Participants and Routing to a Human Task ... 11-8
11.3.8 How to Assign Users, Groups, and Roles to a Participant ... 11-9
11.3.9 *How to Configure the Outcome for Parallel Routing ... 11-10
11.3.10 How to Create and Configure the Data Payload for a Human Task ... 11-11
11.3.11 How to Specify the Presentation of a Human Task ... 11-12
11.4 Assigning a Human Task to a User Task ... 11-12

Part IV Implementing and Deploying a BPM Project

12 Handling Data in Your Business Processes

12.1 About Handling Data Used by Your Business Processes 12-1
12.1.1 How to Define the Data Used by an Oracle BPM Application 12-2
12.2 Introduction to Data Objects 12-2
12.2.1 Introduction to Basic and Complex Data Objects 12-2
12.2.2 Introduction to Process and Project Data Objects 12-3
12.2.2.1 Process Data Objects 12-4
12.2.2.2 Project Data Objects 12-4
12.3 Working with Data Objects 12-4
12.3.1 How to Create a Data Object 12-4
12.3.2 How to Edit or Delete a Data Object 12-6
12.3.3 What Happens When You Delete or Edit a Data Object 12-6
12.4 Introduction to Business Objects 12-6
12.5 Working with Business Objects 12-7
12.5.1 How to Create a Business Object Manually 12-7
12.5.2 How to Create a Business Objects Based on an XML Schema Definition (XSD) 12-9
12.5.3 What Happens When You Create a Business Object 12-9
12.5.4 How to Edit and Delete a Business Object 12-9
12.6 Introduction to Data Associations 12-10
12.6.1 Introduction to the Data Associations Editor 12-12
12.6.2 How to Configure Data Associations for a Flow Object 12-12
12.7 Introduction to Expressions 12-13
12.7.1 Introduction to the Expression Editor 12-13
12.7.2 Types of Expressions 12-13
12.7.3 Simple Expressions 12-14
12.7.3.1 Operator Types 12-14
12.7.3.2 Operator Precedence 12-15
12.8 Working with Expressions 12-15
12.8.1 How to Define a Simple Expression for a Conditional Sequence Flow 12-15
12.8.2 How to Define a Simple Expression in Data Associations 12-16
12.9 Working with Business Indicators and Counter Marks 12-17
12.9.1 Introduction to Business Indicators and Counters 12-17
12.9.2 Introduction to Counter Marks 12-17
12.9.3 How to Add a New Counter Mark to a Process 12-18
12.9.4 How to Delete a Counter Mark 12-18
12.10 Measuring Process Performance Using Measurement Marks 12-18
12.10.1 How to Add a Measurement Mark to a Process 12-19

13 Using Oracle Business Rules

13.1 Introduction to Oracle Business Rules 13-1
13.1.1 Introduction to Rule Conditions 13-2
13.1.2 Introduction to Rule Actions 13-2
13.1.3 Introduction to Decision Tables 13-2
13.1.4 Introduction to Facts and Bucketsets 13-2
13.1.5 Introduction to Rulesets 13-3
13.1.6 Introduction to Decision Functions 13-3
13.1.7 Introduction to Decision Points 13-3

13.1.8 Introduction to Dictionaries 13-3
13.2 Introduction to the Business Process Composer Rules Editor 13-3
13.3 Creating and Editing Business Rules 13-4
13.3.1 How to Create a New Business Rule 13-4
13.3.2 How to Open a Business Rule 13-5
13.3.3 How to Add a Bucketset 13-5
13.3.4 How to Edit an Existing Bucketset 13-5
13.3.5 How to View Globals in the Oracle Rules Dictionary 13-6
13.3.6 How to Add a Rule to a Ruleset 13-6
13.4 Assigning a Rule to a Business Rules Task 13-6
13.5 Editing Oracle Business Rules at Run Time 13-7

14 Communicating with other Processes and Services

14.1 Defining Process Input and Output 14-1
14.1.1 How to Define the Input Arguments for a Process 14-1
14.1.2 How to Define Data Associations for a Message Start Event 14-2
14.1.3 How to Define the Output Arguments for a Process 14-2
14.1.4 How to Define Data Association for a Message End Event 14-2
14.2 Defining Conversations 14-2
14.2.1 Introduction to Conversations 14-3
14.2.2 Working with Conversations 14-3
14.2.2.1 How to define a conversation 14-3
14.2.2.2 How to set the default conversation 14-3
14.2.2.3 How to define a conversation for a BPMN flow object 14-3
14.2.2.4 How to view a collaboration diagram 14-4
14.3 Working with Services 14-4
14.3.1 How to Create New Services in the Business Catalog 14-4

15 Deploying a BPM Project

15.1 Configuring Approval Workflow for a Project 15-1
15.1.1 Introduction to Approval Workflow 15-1
15.1.2 Configuring Approval Workflow 15-1
15.1.2.1 How to Configure Approval Workflow for a Project 15-2
15.2 Deploying a Project 15-2
15.2.1 Who Can Deploy Projects? 15-2
15.2.2 How to Deploy a Project to Run Time 15-2
15.2.3 How to Deploy a Project Using an Approval Workflow 15-4
15.2.4 How to Edit a Deployed Project 15-4
15.2.5 How to Generate a Project SAR File 15-5
15.2.6 How to Generate a Deployment Plan 15-5

Part V Performing Administrative Tasks Using Business Process Composer

16 Performing Administrative Tasks

16.1 Introduction to Business Process Composer Administration 16-1
16.2 How to Assign Global Roles 16-2

16.3 Managing Projects and Project Templates 16-3
16.3.1 How to Delete a Project or Project Template 16-3
16.3.2 How to Configure Sharing for a Project 16-3
16.3.3 How to Release the Lock on a Shared Project 16-4
16.3.4 How to Import a Project Template 16-4
16.4 How to Define Administrator Credentials for Process Player 16-5
16.4.1 How to Enable Process Player 16-5
16.4.2 What Happens When You Enable Process Player 16-5

A BPMN Flow Object Reference

A.1 Using Swimlanes to Organize Your Process A-1
A.1.1 Introduction to Roles A-1
A.1.1.1 Roles in Context A-2
A.1.2 Introduction to Swimlanes A-2
A.1.2.1 Swimlanes in Context A-3
A.1.3 How to Add Roles and Swimlanes to Your Process A-3
A.1.4 How to Edit Swimlane Properties A-3
A.1.5 Sharing Roles Between Business Process Composer and BPM Studio A-4
A.2 Defining the Start and End Point of a Process A-4
A.2.1 Introduction to Start and End Events A-4
A.2.1.1 Specifying the Start Events for Different Types of Processes A-5
A.2.1.2 Using Multiple Start Events in a Process A-5
A.2.1.3 Using Multiple End Events in a Process A-5
A.2.2 Defining How a Process Instance is Triggered A-6
A.2.3 Introduction to the None Start Event A-6
A.2.3.1 The None Start Event in Context A-7
A.2.3.2 Data Associations A-7
A.2.4 Introduction to the Message Start Event A-7
A.2.4.1 The Message Start Event in Context A-8
A.2.4.2 Using Process Input and Output Arguments A-8
A.2.5 Introduction to the Signal Start Event A-8
A.2.5.1 The Signal Start Event in Context A-8
A.2.6 Introduction to the Timer Start Event A-9
A.2.7 Introduction to the Error Start Event A-9
A.2.8 Introduction to the None End Event A-9
A.2.8.1 The None End Event in Context A-10
A.2.9 Introduction to the Error End Event A-10
A.2.10 Introduction to the Message End Event A-10
A.2.11 Introduction to the Terminate End Event A-11
A.3 Adding User Interaction to Your Process A-11
A.3.1 Introduction to Human Workflow A-11
A.3.1.1 Introduction to Human Tasks A-11
A.3.2 Introduction to the User Task A-12
A.3.2.1 The User Task in Context A-12
A.3.2.2 Using Interactive Activities A-13
A.3.2.3 Using the User Task in Project Templates A-13
A.3.3 Introduction to the Manual Task A-14

A.3.3.1 The Manual Task in Context A-14
A.3.4 Introduction to the Update Task A-14
A.4 Communicating With Other Processes and Services A-15
A.4.1 Introduction to the Service Task A-15
A.4.1.1 The Service Task in Context A-16
A.4.1.2 Implementing Reusable Services in Project Templates A-16
A.4.2 Introduction to the Notification Task A-16
A.4.3 Introduction to the Call Activity A-17
A.4.4 Introduction to the Send Task A-17
A.4.4.1 The Send Task in Context A-18
A.4.5 Introduction to the Receive Task A-18
A.4.5.1 The Receive Task in Context A-18
A.4.5.2 Starting a Process with the Receive Task A-18
A.4.6 Using the Send and Receive Tasks to Communicate Between Processes A-19
A.4.7 Introduction to the Message Throw Event A-19
A.4.8 Introduction to the Message Catch Event A-20
A.4.9 Using Message Throw and Catch Events to Communicate Between Processes A-21
A.5 Adding Business Logic Using Oracle Business Rules A-22
A.5.1 Introduction to Oracle Business Rules A-22
A.5.2 Introduction to the Business Rule Task A-22
A.5.2.1 The Business Rule Task in Context A-23
A.6 Controlling Process Flow Using Sequence Flows A-23
A.6.1 Introduction to Sequence Flows A-23
A.6.2 Introduction to Unconditional Sequence Flows A-23
A.6.3 Introduction to Conditional Sequence Flows A-24
A.6.4 Introduction to Default Sequence Flows A-24
A.7 Controlling Process Flow Using Gateways A-24
A.7.1 Introduction to Gateways A-24
A.7.1.1 Split-Merge Pairs A-25
A.7.2 Introduction to the Exclusive Gateway A-25
A.7.2.1 The Exclusive Gateway in Context A-25
A.7.2.2 Splitting and Merging Exclusive Gateways A-26
A.7.3 Introduction to the Inclusive Gateway A-26
A.7.3.1 Splitting and Merging Inclusive Gateways A-27
A.7.4 Introduction to the Parallel Gateway A-27
A.7.4.1 The Parallel Gateway in Context A-27
A.7.4.2 Splitting and Merging Parallel Gateways A-28
A.7.5 Introduction to the Complex Gateway A-28
A.7.6 Introduction to the Event-based Gateway A-29
A.7.6.1 Starting a Process with an Event-Based Gateway A-30
A.8 Controlling Process Flow Using Intermediate Events A-30
A.8.1 Introduction to Intermediate Events A-30
A.8.2 Introduction to the Timer Catch Event A-30
A.8.3 Introduction to the Error Catch Event A-31
A.9 Using Subprocesses in Oracle BPM A-32
A.9.1 Introduction to Reusable Processes (Reusable Subprocesses) A-32
A.9.2 Introduction to Embedded Subprocesses (Inline Subprocesses) A-32

A.9.2.1 Embedded Subprocesses and Sequence Flows A-33
A.9.2.2 Embedded Subprocesses in Context A-33
A.9.2.3 Looping Embedded Subprocesses A-34
A.9.3 Introduction to Event Subprocesses (Event Handlers) A-34
A.10 Changing the Value of Data Objects in Your Process A-34
A.10.1 Introduction to the Script Task A-34
A.10.1.1 The Script Task in Context A-35

B BPMN Flow Object Property Reference

B.1 Common Properties B-1
B.1.1 Basic Properties B-1
B.1.2 Implementation Properties B-1
B.2 Interactive Properties B-2
B.2.1 Interactive Activities B-2
B.2.2 Manual Task B-3
B.3 Activity Properties B-3
B.3.1 Service Task B-3
B.3.1.1 Implementation Properties B-3
B.3.2 Send Task B-4
B.3.2.1 Implementation Properties B-4
B.3.3 Receive Task B-5
B.3.3.1 Implementation Properties B-6
B.3.4 Business Rule Task B-6
B.3.4.1 Implementation Properties B-6
B.3.5 Script Task B-7
B.3.6 Call Activity B-7
B.3.6.1 Implementation Properties B-7
B.3.7 Subprocesses B-7
B.3.7.1 Implementation Properties B-7
B.3.8 Inline Handlers B-7
B.4 Gateway Properties B-8
B.4.1 Exclusive Gateway B-8
B.4.2 Inclusive Gateway B-8
B.4.3 Parallel Gateway B-8
B.4.4 Complex Gateway B-8
B.4.5 Event-Based Gateway B-9
B.5 Event Properties B-9
B.5.1 The None Start Event B-9
B.5.2 The Message Start Event B-9
B.5.2.1 Implementation Properties B-9
B.5.3 The Timer Start Event B-10
B.5.3.1 Implementation Properties B-10
B.5.4 The Signal Start Event B-10
B.5.4.1 Implementation Properties B-11
B.5.5 The Error Start Event B-11
B.5.5.1 Implementation Properties B-11
B.5.6 None Catch Event B-11

B.5.7 Message Catch Event B-11
B.5.7.1 Implementation Properties B-11
B.5.8 Timer Catch Event B-12
B.5.8.1 Implementation Properties B-13
B.5.9 Error Catch Event B-13
B.5.9.1 Implementation Properties B-13
B.5.10 Message Throw Event B-13
B.5.10.1 Implementation Properties B-13
B.5.11 Signal Throw Event B-15
B.5.11.1 Implementation Properties B-15
B.5.12 None End Event B-15
B.5.13 Message End Event B-15
B.5.13.1 Implementation Properties B-15
B.5.14 Signal End Event B-16
B.5.14.1 Implementation Properties B-16
B.5.15 Error End Event B-16
B.5.15.1 Implementation Properties B-16
B.5.16 Terminate End Event B-16
B.6 Measurement Mark Properties B-16
B.7 Sequence Flow Properties B-17
B.7.1 Default Sequence Flow B-17
B.7.2 Normal Sequence Flow B-17
B.7.3 Conditional Sequence Flow B-17

C Web Form and Web Form Control Property Reference

C.1 Web Form Properties C-1
C.1.1 Settings Tab C-1
C.1.2 Style Tab C-2
C.2 Web Form Control Properties C-2
C.2.1 Web Form Control Properties - Settings Tab C-2
C.2.2 Web Form Control Properties - Style Tab C-9

D Web Form Rules Examples

D.1 Calculate a Total D-1
D.2 Show/Hide a Billing Address D-1
D.3 Show/Hide Message D-2
D.4 Enable/disable a question D-2
D.5 Compute Subtotals for Repeating Items D-2
D.6 Compute an Invoice Total D-3
D.7 Textarea Max Length D-3
D.8 Textarea Newline and Break D-4
D.9 Dropdown Options D-4
D.10 Finding a Selected Options Index D-5
D.11 Synchronized Selects D-5
D.12 Clearing Dropdown Options D-6
D.13 Default Option D-6
D.14 Checkbox Options - Assigning Color to Checkbox Choices D-6

D.15 Checkbox Options - Making a Control Visible/Invisible Based on Checkbox Choices ... D-7
D.16 Checkbox Initialization ... D-7
D.17 Displaying Selected Checkbox Labels ... D-8
D.18 Repeating Checkboxes ... D-8
D.19 Display a Message Control Inside a Repeat Control ... D-8
D.20 String Concatenation ... D-9
D.21 Dynamic Labels, Help, Hints ... D-10
D.22 Visible/Invisible ... D-10
D.23 Visible/Invisible Section ... D-10
D.24 Select Tab ... D-11
D.25 Next Tab ... D-11
D.26 Expand/Collapse Section ... D-12
D.27 Security Subject Information ... D-12
D.28 Multiple Choice ... D-12
D.29 Dynamic Options ... D-13
D.30 Triggers and Dynamic Options ... D-13
D.31 Value Change and Dynamic Options ... D-14
D.32 Dynamic Control Initialization ... D-14
D.33 Verify User ... D-15
D.34 Calculate Net Worth ... D-15
D.35 Dates and Times ... D-16
D.35.1 Duration ... D-16
D.35.2 Today's Date and Time ... D-16
D.35.3 Date/Time Stamp ... D-17
D.35.4 Invalid if Before Today ... D-17
D.35.5 Date no more then 14 days from Today ... D-17
D.35.6 Date no more then 30 days ago ... D-17
D.35.7 Central Timezone adjusted for Daylight Savings ... D-18
D.35.8 Hours >= 4 and <= 6 Apart ... D-19
D.35.9 Times ... D-19
D.36 Tenants, Roles, Users ... D-19
D.37 Repeat Item Added ... D-20
D.38 Repeat Item Added - Collapse Other Items ... D-21
D.39 Tables ... D-21
D.40 form.load ... D-22
D.41 form.unload ... D-22
D.42 Unique ID ... D-23
D.43 Repeat Item Initialization ... D-23
D.44 Repeat ItemAdded by Init Doc ... D-24
D.45 Search Popup ... D-24

E Preparing Processes for Import into BPMN

E.1 Preparing a Visio File to Import as a BPMN Process ... E-1
E.1.1 How to Update VisioUserMap.xml ... E-2
E.1.2 Valid BPMN Element Values ... E-2
E.1.3 BPMN Element Attributes ... E-3
E.2 How to Customize XPDL Import Using XSL Doc ... E-4

E.3 Preparing an XPDL File to be Imported as a BPMN Process E-6
E.3.1 Handling Namespaces E-6
E.3.2 Handling Relative Coordinates E-7
E.3.3 Handling Extended Attributes E-10
E.3.4 Handling redrawConnections E-10
E.3.5 Handling isRelativeObjectCoordinates E-11
E.3.6 Removing Invisible Elements E-12
E.3.7 Handling the Orientation Attribute E-13
E.3.8 Specifying the View Type for Subprocesses E-13
E.3.9 Handling the Object Pin E-15
E.3.10 Modifying the Height and Width of Activities E-15
E.3.11 Modifying the Height and Width of Lanes E-17
E.3.12 Modifying the Height and Width of Pools E-18
E.3.13 Location of Activities E-18
E.3.14 Including Missing Elements E-18
E.3.15 Checking the Correctness of Activities E-19

F The Sales Quote Example Process

F.1 Introduction to Business Process Management Notation (BPMN) F-1
F.1.1 What is Business Process Management Notation (BPMN) F-1
F.1.2 Business Processes F-1
F.1.2.1 Process Instances F-2
F.1.2.2 Process Tokens F-2
F.1.3 Flow Objects F-2
F.1.4 Data Objects F-2
F.2 Introduction to the Sales Quote Example F-2
F.2.1 Breakdown of the Sales Quote Example F-3
F.2.1.1 Initiate Sales Quote F-3
F.2.1.2 Determine Business Practice Review F-4
F.2.1.3 Approve Quote F-4
F.2.1.4 Approvals Outcome F-5

Preface

This guide describes the Oracle Business Process Composer application.

Intended Audience

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

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

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

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following Oracle resources:

Oracle Business Process Management

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

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

Oracle SOA and BPM Suite Installation and Administration

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

Conventions

The following conventions are also used in this manual:

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

What's New in This Guide for Release 11.1.1.7

For Release 11.1.1.7, this guide has been updated in several ways. The following table lists the content that has been added or changed.

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

Content	Changes Made
Chapter 2, "Introduction to Oracle Business Process Composer"	Chapter revised to reflect changes to the Oracle Business Process Composer use cases.
Chapter 3, "Walkthrough: Creating a Basic BPM Application"	Chapter added to provide walk-through of the use case for creating a new process-based business application using Business Process Composer.
Section 4.1.1, "Introduction to Project Components and Resources"	Section revised to reflect changes to the BPM project resources and the business catalog. For example, you can create new business rules and create and edit business objects using Business Process Composer.
Section 4.2, "Introduction to the Project Welcome Page"	Section revised to reflect redesign on the Project Welcome page.
Chapter 6, "Simulating Process Behavior"	Chapter added to describe how to use Business Process Composer to run simulations to improve process performance.
Chapter 7, "Using Process Player"	Chapter added to describe how to use process player to test the behavior of your business processes.
Chapter 9, "Working with Web Forms"	Chapter added to describe how to create and use web forms using Business Process Composer.
Chapter 10, "Working with Web Form Rules"	Chapter added to describe how to use web form rules to add dynamic behavior to a web form.
Section 11.3, "Working with Human Tasks"	Revised section to include information about creating a web form based on a human task payload.
Section 11.4, "Assigning a Human Task to a User Task"	Added section to describe how to assign a human task to a user task.
Section 12.4, "Introduction to Business Objects"	Added section to provide general information about business objects.
Section 12.5, "Working with Business Objects"	Added section to describe how to create and edit business objects using Business Process Composer.

Content	Changes Made
Section 12.9, "Working with Business Indicators and Counter Marks"	Added section to describe how to create business indicators using Business Process Composer.
Section 13.3, "Creating and Editing Business Rules"	Revised section to document how to create new business rules using Business Process Composer.
Section 16.4, "How to Define Administrator Credentials for Process Player"	Added section to describe how to enable the process player feature.
Appendix A, "BPMN Flow Object Reference"	Added an appendix describing each of the BPMN flow objects supported by Oracle BPM.
Appendix C, "Web Form and Web Form Control Property Reference"	Added a reference appendix describing each of the web form and web form control properties.
Appendix D, "Web Form Rules Examples"	Added an appendix containing various examples and use cases for web form rules.

1

Oracle Business Process Management Suite Overview

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

This chapter includes the following sections:

- Section 1.1, "Introduction to the Oracle Business Process Management Suite"
- Section 1.2, "Oracle BPM User Personas"
- Section 1.3, "Oracle BPM Suite Components"
- Section 1.4, "Oracle Business Process Analysis (BPA) Suite"
- Section 1.5, "Introduction to the Application Development Life Cycle"
- Section 1.6, "Oracle BPM Use Cases"
- Section 1.7, "Accessibility Options"

1.1 Introduction to the Oracle Business Process Management Suite

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

The Oracle BPM Suite:

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

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

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

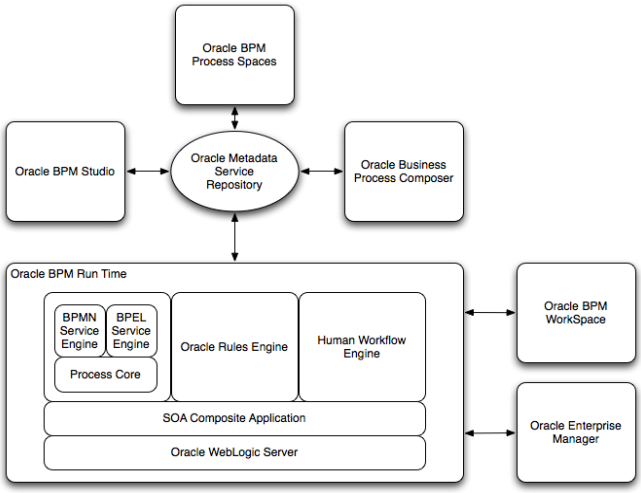
The Oracle BPM Suite provides a seamless integration of all stages of the application development life cycle from design-time and implementation to runtime and application management.

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

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

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

Figure 1–1 The Oracle BPM Suite



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

1.2 Oracle BPM User Personas

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

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

Table 1–1 Oracle BPM User Persona

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

1.3 Oracle BPM Suite Components

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

1.3.1 Process Modeling and Implementation

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

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

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

1.3.1.1 Oracle BPM Studio

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

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

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

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

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

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

1.3.1.2 Oracle Business Process Composer

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

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

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

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

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

See Chapter 2, "Introduction to Oracle Business Process Composer" for more information.

1.3.1.3 Oracle Metadata Service (MDS) Repository

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

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

1.3.1.4 Oracle BPM Projects

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

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

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

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

1.3.2 Oracle BPM Runtime Components

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

1.3.2.1 Oracle BPM Engine

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

The BPM engine is composed of three separate components:

- BPMN Service Engine

  The BPMN engine provides an environment for running BPMN processes.

- BPEL Service Engine

  The BPEL engine provides an environment for running BPEL processes.

- Process Core

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

  – Manage security

  – Generate audit trails

  – Invoke services

  – Manage persistence

1.3.2.2 Oracle Human Workflow

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

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

1.3.2.3 Oracle Business Rules

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

1.3.2.4 Oracle WebLogic Application Server

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

1.3.2.5 Oracle Enterprise Manager

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

1.3.3 Oracle BPM Suite Process Participant Applications

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

1.3.3.1 Oracle Business Process Management Workspace (Process Workspace)

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

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

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

1.3.3.2 Oracle Business Process Management Process Spaces (Process Spaces)

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

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

1.3.4 Other Oracle BPM Suite Components

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

1.3.4.1 Process Analytics

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

1.3.4.2 Guided Business Processes

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

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

1.4 Oracle Business Process Analysis (BPA) Suite

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

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

Additionally, the Oracle BPA Suite supports the following:

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

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

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

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

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

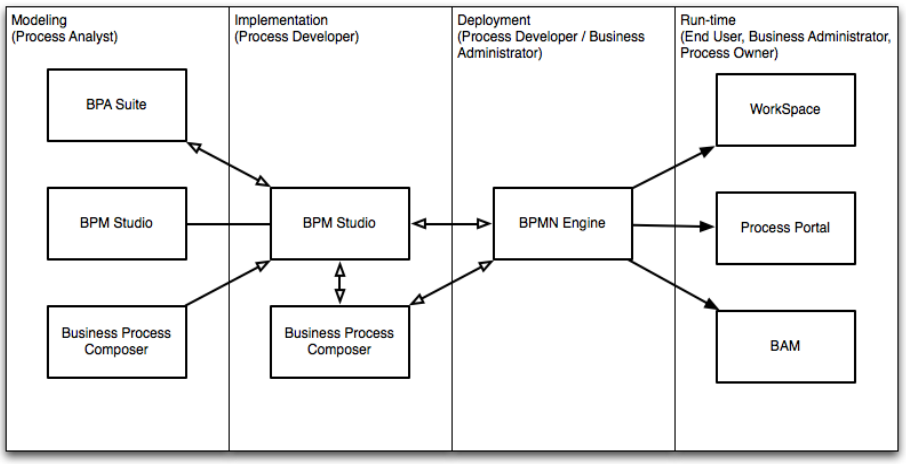
See the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for more information on using business processes created in Oracle BPA within Oracle BPM Studio.

1.5 Introduction to the Application Development Life Cycle

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

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

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



1.5.1 Process Modeling

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

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

- Oracle BPM Studio

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

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

- Oracle Business Process Composer

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

- Oracle Business Process Analysis (BPA) Suite

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

See Section 1.6, "Oracle BPM Use Cases" for more information on how each of these tools fit within the typical Oracle BPM uses cases. See Section 2.2, "Introduction to Business Process Composer Use Cases" for more information on how Oracle Business Process Composer and Oracle BPM Studio interact within the application development life cycle.

1.5.2 Implementation

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

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

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

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

1.5.3 Deployment

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

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

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

- Deployment directly from Oracle BPM Studio

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

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

- Deployment directly from Business Process Composer

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

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

- Deployment using an exported SAR file

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

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

- Deployment Using the WebLogic Scripting Tool (WLST)

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

1.5.4 Oracle BPM Runtime

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

- User interaction

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

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

- Process management and monitoring

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

- Process creation

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

- System administration

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

1.6 Oracle BPM Use Cases

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

1.6.1 Use Case: Using Business Process Composer to Create Projects

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

Typical Workflow for Using Business Process Composer to Create New Projects

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

6. Deploy the project to Oracle BPM runtime (process analyst, process administrator).

1.6.2 Use Case: Using BPM Studio to Create Project Templates

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

Typical Workflow for Using Oracle BPM Studio to Create Project Templates

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

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

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

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

Typical Workflow for Using Oracle BPM Studio to Model Processes

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

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

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

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

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

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

1.7 Accessibility Options

This section describes accessibility options available with Oracle BPM Suite.

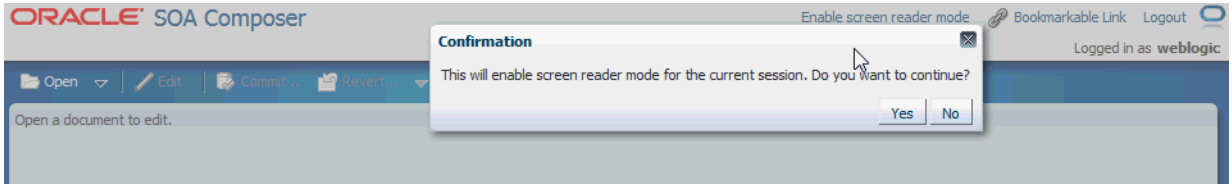
1.7.1 Enabling Accessibility Features in SOA Composer

SOA Composer provides the screen reader option, which enables your screen reader to access and read all components of the application.

To enable screen reader:

1. Click the **Enable screen reader mode** link in the top right corner.
2. A confirmation message, **This will enable screen reader mode for the current session. Do you want to continue?**, appears as shown in Figure 1–3.

Figure 1–3 Enable Screen Reader Confirmation Message



3. Click **Yes** to confirm.

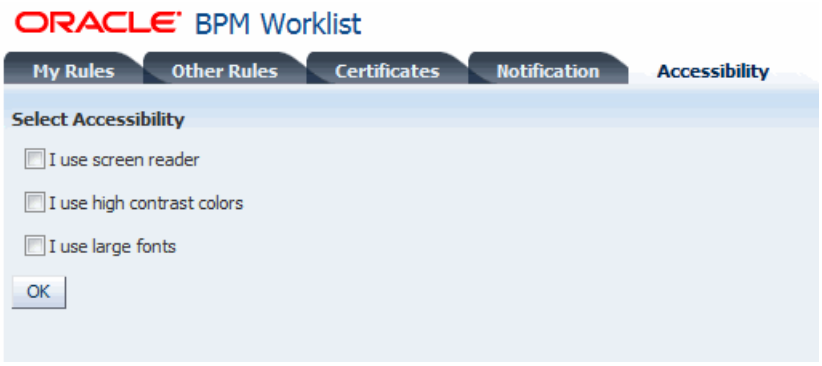
1.7.2 Enabling Accessibility Features in BPM Worklist

BPM Worklist provides accessibility options, which help you read all components of the application.

To enable accessibility options:

1. Click **Preferences** in the top right corner.
2. On the Accessibility tab, shown in Figure 1–4, select the desired option.

Figure 1–4 BPM Worklist Accessibility Options



3. Click **OK**.

Part I

Getting Started with Oracle Business Process Composer

This part provides a general introduction to the Oracle Business Process Composer application. It also provides a general description of how Business Process Composer is used within the overall process development life cycle.

This part contains the following chapters:

- Chapter 2, "Introduction to Oracle Business Process Composer"
- Chapter 3, "Walkthrough: Creating a Basic BPM Application"

2

Introduction to Oracle Business Process Composer

This chapter provides an overview of Oracle Business Process Composer. It describes the most common scenarios for using Business Process Composer to design, implement, and deploy BPM projects to create process-based business applications It also provides a general overview of the application user interface. For general information about Oracle BPM see Chapter 1, "Oracle Business Process Management Suite Overview."

This chapter includes the following sections:

- Section 2.1, "Introduction to Oracle Business Process Composer"
- Section 2.2, "Introduction to Business Process Composer Use Cases"
- Section 2.3, "Signing On to Oracle Business Process Composer"
- Section 2.4, "Introduction to the Business Process Composer User Interface"

2.1 Introduction to Oracle Business Process Composer

Oracle Business Process Composer is a web-based, role-driven, collaborative tool for the discovery, design and documentation of business processes. Business Process Composer is aimed at a business audience and enables business users to participate in the definition, feedback and design of business processes and process-based applications.

Business Process Composer:

- Provides an easy-to-use and intuitive user interface, enabling users to quickly become productive.
- Provides a standards based, comprehensive modeling environment. You can model business processes, create business rules, business data, and user interface, and validate business processes.
- Contains built-in collaboration. You can see who else is editing a project and invite others to participate.
- Provides security and permissions based on roles. Users have different privileges that can be defined at a global or project level.
- Provides model-driven implementation. The Business Process Modeling (BPMN) 2.0 process model is also the executable model. The same model is enhanced using Oracle BPM Studio. After using Oracle BPM Studio to provide implementation details, a project can be shared again with business users. There is no round-trip or transformation issues as both business and IT work on the same models.

- Enables business users to perform implementation. In addition to modeling business processes, business users can also create business rules, web forms, business data, business indicators (business metrics).

Oracle Business Process Management Studio is a separate tool within the Oracle BPM suite. Oracle BPM Studio runs in Oracle JDeveloper and is primarily used by process developers. Oracle BPM Studio provides comprehensive developer tools and enables them to develop components, including Web Services, XML, and integration development, that are required in a more complete BPM solution.

See the *Oracle BPM Modeling and Implementation Guide* for information about using Oracle BPM Studio.

2.1.1 Process Design and Implementation

The development life-cycle of an Oracle BPM application can be divided into two stages: design and implementation. These are defined as:

- Design: includes creating and testing process flow and logic. Business Process Composer supports BPMN 2.0 and provides editors that enables you to drag and drop flow objects and sequence flows into a business process.

  See Part II, "Modeling and Testing Business Processes" for information about designing business processes using Business Process Composer. See Appendix A, "BPMN Flow Object Reference" for information about Oracle's BPMN 2.0 implementation.

- Implementation: includes adding technical details to the business processes created during the design phase. Business Process Composer enables you to define process data, organizational roles, human tasks, and business rules. It provides editors for creating and editing each of these components.

  See Part IV, "Implementing and Deploying a BPM Project" for information about implementing a business process. See Part III, "Defining How Users Interact with Your Business Processes" for information about defining the user interaction of your process-based application.

Business Process Composer enables business users to perform tasks related to both design and implementation. Using Business Process Composer you can design, implement, test, and deploy your process-based application. See Section 2.2.1, "Use Case: Creating a New Oracle BPM Application" for the use case for creating a new process-based application.

In addition to creating BPM applications from the ground up, Oracle BPM enables you to create templates that contain predefined processes and implementation details. Business users can create new applications based on these templates.

See Section 2.1.2, "Collaboration" for more information. See Section 2.2.2, "Use Case: Create an Oracle BPM Application Based on a Project Templates" for information about the use case for working with project templates.

Business Process Composer also enables you to import business processes created in other applications, including Microsoft Visio. See Section 5.8, "Importing and Exporting Process Models" for more information.

2.1.2 Collaboration

Developing a business application often requires collaboration among business users and process developers. You can use Business Process Composer to collaborate with

other business users who also use Business Process Composer. You can also collaborate with process developers using Oracle BPM Studio.

To facilitate collaboration among different users and tools, Oracle BPM provides the following components:

- **BPM projects:** contains all of the processes and other components required for a fully-functioning business application. To edit a BPMN process or other application component, you must open and edit a BPM project. BPM projects can be shared among Business Process Composer and Oracle BPM Studio users.

  See Section 4.1, "Introduction to Oracle BPM Projects" for more information.

- **BPM project templates:** are templates based on a BPM project. A project template can define the basic business processes and implementation details of a business application. Business users can then create new BPM projects based on these templates.

  See Chapter 8, "Working with the Project Life Cycle" for more information working with project templates.

- **BPM repository:** provides a central, shared location for sharing BPM projects and project templates. When you save a BPM project using Business Process Composer, it is saved in the BPM repository.

  See Section 4.1.2, "Introduction to the Oracle BPM Repository" for more information on the BPM repository.

  Process developers using Oracle BPM Studio can also connect to the BPM repository.

These components of the Oracle BPM Suite enable business users and process developers to share and collaborate on a BPM project. When sharing BPM projects using the BPM repository, all users edit the same BPMN processes and application components without having to import or export a BPM project.

2.1.3 Simulation and Testing

Business Process Composer provides features for simulating and testing process models. You can perform both of these tasks during process design, without having to deploy a BPM project to runtime. This enables you to use one tool for designing and testing you application.

- **Simulations**: enable you to simulate the performance of your business processes. You can create multiple simulation models with different parameter configurations. This enable you to test and compare different scenarios to improve the performance of your processes.

  See Chapter 6, "Simulating Process Behavior" for more information.

- **Process player**: enables you to run the processes in your application to test process behavior and logic. Process player provides an environment for testing your processes within Business Process Composer, without having to deploy your project and view the processes using Process Workspace.

  See Chapter 7, "Using Process Player" for more information.

2.1.4 Deployment

Business Process Composer enables you to deploy a BPM project directly to the run time environment. This enables business users to deploy the application directly without having to rely on process developers or administrators.

See Chapter 15, "Deploying a BPM Project" for more information on deploying a BPM project using Business Process Composer.

2.2 Introduction to Business Process Composer Use Cases

There are two primary use cases for Oracle Business Process Composer:

- Create new BPM projects.

  Business Process Composer enables process analysts and developers to create deployable process-base applications from the ground up. These applications are contained in a BPM project. BPM projects contain all the business processes and other technical components required by an application.

  Business Process Composer also provides a collaborative environment where process analysts and developers can share BPM projects during all phases of the application development life cycle.

  For example, a process analyst can create the process flows required by the application. A process developer can then open the project in Business Process Composer and add the components necessary to connect the business processes with other systems and services. Or, if required, a process developer can open the project in Oracle BPM Studio to utilize all of the functionality available in the Oracle BPM and SOA suites.

  After creating the required business processes and creating and implementing the supporting components, you can use Business Process Composer to deploy a BPM project directly to runtime.

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

  Project templates are BPM projects used as templates to create other process based applications. Project templates are created in Oracle BPM Studio and are stored within the Oracle BPM Metadata Service partition. Using Business Process Composer, you can use these templates to create new or modify existing BPM projects.

  After creating and modifying a BPM project based on a project template, you can then save them to the BPM repository deploy them to runtime.

  See Section 8.2, "Introduction to BPM Project Templates" for information on using project templates in Business Process Composer. For information on creating project templates see the *Oracle BPM Modeling and Implementation Guide*.

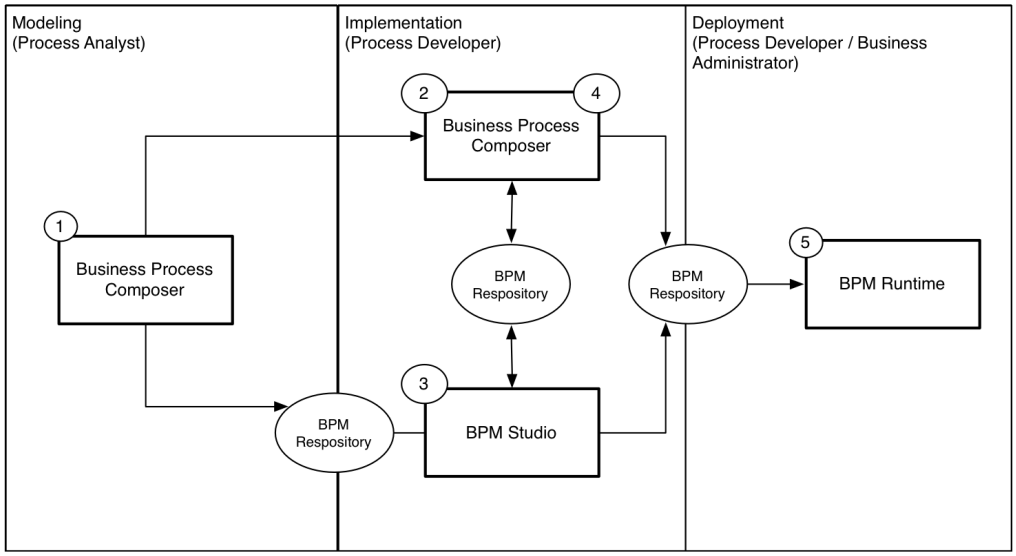
  The following sections describe typical scenarios for using Business Process Composer by showing the interactions between different Oracle BPM components and the persona who would be responsible for each stage of the development cycle. However, the exact workflow you use depends on your business needs. There may be multiple iterations where process analysts and developers collaborate to create and refine a business process

2.2.1 Use Case: Creating a New Oracle BPM Application

Figure 2–1 describes the workflow for creating a new Oracle BPM application using Business Process Composer. This workflow outlines the most general path for creating a deployable application. A real-world scenario may involve several iterations of the design, implementation, and testing stages. It would also include collaboration between multiple analysts and developers.

See Chapter 3, "Walkthrough: Creating a Basic BPM Application" for a detailed walkthrough of this use case.

Figure 2–1 Use Case: Creating a New BPM Applications



The following describes each stage of the workflow. See Section 1.2, "Oracle BPM User Personas" for information about the user personas used in this workflow.

1. Model your business process. (process analyst / process developer)

   The first step in creating a process-based application is to model the required business processes. This is often performed by a process analyst who has in-depth knowledge of the business needs of the application, but may also involve collaboration with process developers who provide input on the technical requirements of the process model.

   The general workflow for designing a business process is:

   a. Create a BPM project.

      A BPM project contains all of the processes and resources of a process-based application. See Section 4.4, "Creating and Working with Projects" for more information.

   b. Create a BPMN process.

      After creating a project, the next step in designing an Oracle BPM application is to create the required business processes. Business processes contain the models that define the behavior of your application.

      See Section 5.3.1, "How to Create a New Business Process" for information on creating a business process. See Section 5.1, "Introduction to Business Processes" for general information on business processes.

   c. Add the required BPMN flow objects and sequence flows to your process.

   d. Design the user interaction of your business process.

   See Part II, "Modeling and Testing Business Processes" for information about using Business Process Composer to design business processes.

2. Implement your business process (process analyst / process developer).

After the process analyst has modeled and tested the processes required by an Oracle BPM application, the process developer can then integrate those processes with other components of the application as well as external systems and services.

a. Define the data used in your application (process developer).

Oracle BPM enables you to define the data structures required by your application.

See Section 12.1.1, "How to Define the Data Used by an Oracle BPM Application" for general procedures for creating the data structures in your application. For information on data objects see Section 12.2, "Introduction to Data Objects". For information on business objects see Section 12.4, "Introduction to Business Objects".

b. Define the human tasks required by your application (process developer).

Human tasks are used to define the workflow your application users follow to accomplish their tasks. Human tasks employ forms to define the user interface for your application.

c. Connect your processes to other processes, systems, and services (process developer).

Process developers can configure BPMN processes to access other processes, systems and services. See Section 14, "Communicating with other Processes and Services" for more information.

3. Create advanced implementation details (process developer).

Use Oracle Business Process Management Studio to create and advanced implementation details required by your application.

a. Define how your process handles errors.

b. Configure advanced human task parameters.

4. Test the performance and behavior of your process (process analyst).

You can test the performance of your process by running simulations using Business Process Composer. See Chapter 6, "Simulating Process Behavior" for more information.

You can also use Business Process Composer to test the behavior of your process using process player. See Chapter 7, "Using Process Player" for more information.

5. Deploy your BPM project (process analyst).

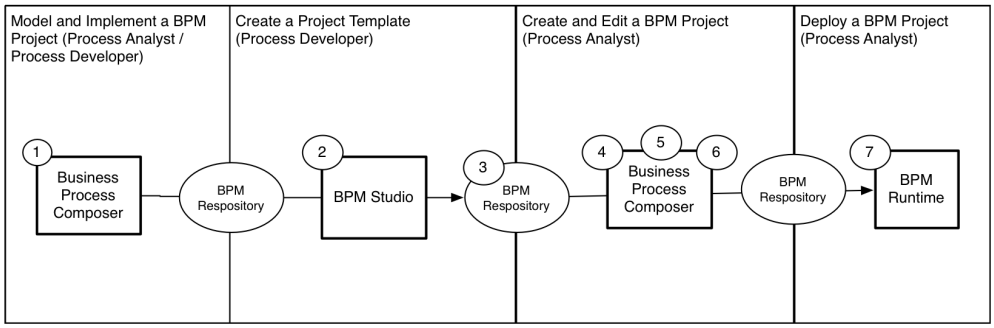
Deploying a BPM project is the process of installing the project in the Oracle BPM runtime environment. There are several methods of deploying the BPM project to runtime. Which method you use depends on your role and the tool used to deploy the application.

- Deploy the process directly to runtime using Business Process Composer (process analyst, process developer). See Section 15.2, "Deploying a Project" for more information.
- Save the BPM project to the BPM repository, then open and deploy it from Oracle BPM Studio (process developer). See the *Oracle BPM Modeling and Implementation Guide* for more information.
- Export the project as a SAR file (process analyst, process developer), which is then deployed to Oracle BPM runtime (business administrator).

2.2.2 Use Case: Create an Oracle BPM Application Based on a Project Templates

Figure 2–2 describes the typical workflow for using BPC to create an Oracle BPM application based on a project template. Project templates can be used to provide a library of ready-made BPM projects that can be used to quickly create new deployable applications. Project templates contain all of the required implementation pre-configured as part of the business catalog. This enables process analysts to modify the process as necessary, connect the necessary implementation, and deploy the project to runtime.

Figure 2–2 Creating an Oracle BPM Application Based on a Project Template



The following describes each stage of this workflow. See Section 1.2, "Oracle BPM User Personas" for information about the user personas used in this workflow.

1. Create a working BPM application as described in Section 2.2.1, "Use Case: Creating a New Oracle BPM Application"
2. Create a project template (process developer).

   Process developers create project templates using Oracle BPM Studio. Project templates are based on a normal BPM project, but generally contain all of the required elements of the business catalog pre-configured.
3. Publish the project template to the Oracle BPM repository (process developer)

   After creating a project template, you can publish it to the Oracle BPM repository Oracle BPM Studio. Project templates published in the Oracle BPM are available to process analysts using Business Process Composer.
4. Create a project based on a project template using Business Process Composer. (process analyst)

   You can use Business Process Composer to create new BPM projects based on project templates.

   See Section 8.3, "Working with Project Templates" for information on how to create a new project based on a project template. For general information about project templates, see Section 8.2, "Introduction to BPM Project Templates".
5. Edit the BPM project and processes based on the edit policies defined in the template (process analyst).

   Edit policies determine what changes you can and cannot make to a project created from a project template. These are determined by the process developer who creates the project template.

   If no edit policies are defined, then you can edit the project and processes like any other BPM project. See Appendix A, "BPMN Flow Object Reference" for reference information on each of the BPMN flow objects supported by Oracle BPM.

6. Validate the project (process analyst).

   Although you should validate your project throughout the modeling and implementation cycle, you must resolve any validation errors before the project can be deployed to runtime.

7. Deploy the project to runtime.

   Deploying a project is the process of installing the project in the Oracle BPM runtime environment. There are several methods of deploying the BPM project to runtime. Which method you use depends on your role and the tool used to deploy the application.

   - Deploy the process directly to runtime using Business Process Composer (process analyst, process developer). See Section 15.2, "Deploying a Project" for more information.
   - Save the BPM project to the BPM repository, then open and deploy it from Oracle BPM Studio (process developer)
   - Export the project as a SAR file (process analyst, process developer), which is then deployed to Oracle BPM runtime (business administrator).

2.3 Signing On to Oracle Business Process Composer

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

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

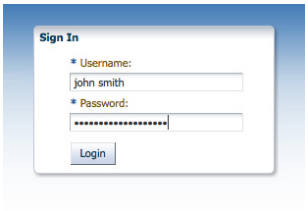
> **Note:** Oracle Application Server Single Sign-On is enabled by default in Oracle BPM Suite. OracleAS Single Sign-On enables you to use one sign on session to access multiple web-based applications. If OracleAS Single Sign-On is enabled and you have previously signed on to another application, the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

To sign on to Oracle Business Process Composer

1. Go to the Business Process Composer URL.

   Figure 2–3 shows the sign-on screen that appears after the Oracle BPM application loads.

Figure 2–3 Oracle Business Process Composer Sign-on Screen



Business Process Composer

****

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

After signing on to Business Process Composer the Application Home page displays as shown in Figure 2–4.

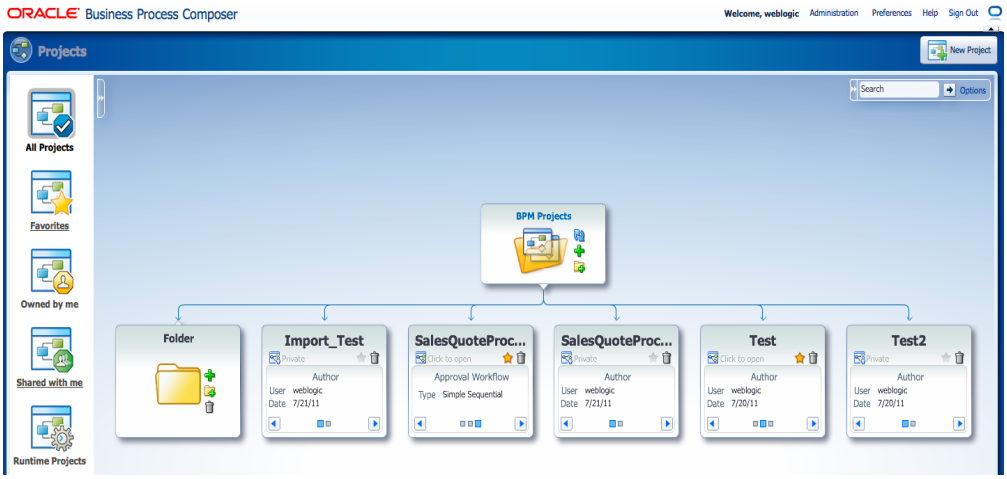
> **Note:** You can only sign on to Business Process Composer from one browser session. Concurrent sessions for the same user are not supported.

2.4 Introduction to the Business Process Composer User Interface

The Business Process Composer application is designed to enable you to easily create, edit, and manage BPM projects. A BPM project is the core component of an Oracle BPM application which contains all the require resources of the application, including business processes. See Chapter 4, "Working with BPM Projects" for more information.

The Business Process Composer user interface is shown in Figure 2–4.

Figure 2–4 The Oracle Business Process Composer Application User Interface



There are four main components of the Business Process Composer application:

- Business Process Composer toolbar
- Business Process Composer main menu
- Business Process Composer welcome page
- Project editor

Each of these are described in the following sections.

2.4.1 Introduction to the Business Process Composer Toolbar

The Business Process Composer toolbar provides access to general application functionality. Figure 2–5 shows the application toolbar.

Figure 2–5 The Business Process Composer Toolbar

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

This toolbar is located in the upper right-hand corner of the application. It is available from each page of the application. The application toolbar provides access to the following:

Table 2–1 The Business Process Composer Application Toolbar

Toolbar element	Description
Username	Displays the name of the current user. This text field is read-only.
Administration	Provides access to the Business Process Composer administration page. This item is only visible to users who have been granted the Administrator security role. See Chapter 16, "Performing Administrative Tasks" for more information.
Preferences	Enables you to configure general application preferences.
Help	Displays an HTML version of this guide.
Sign Out	Sign out the current user.
Network connectivity	Displays the current network status. ■ A stationary blue icon indicates that there is no network activity. ■ A spinning blue icon indicates that Business Process Composer is connecting to the application server. ■ A spinning red icon indicates that Business Process Composer is attempting to connect to the application server, but is unable to connect. ■ A stationary red icon means that the connection with the application server has been lost.

2.4.2 Introduction to the Business Process Composer Welcome Page

Use the Business Process Composer Application Welcome Page to view and work with the projects stored in the BPM repository. Figure 2–6 shows the project Welcome page. The different areas and functions of the welcome page are described in the following sections.

Figure 2–6 The Project Browser

Chapter 4, "Working with BPM Projects" for information on.

2.4.2.1 Project Views

Project views enable you to view the project browser based on certain criteria. You can select a project view by selecting its icon from the left-hand side of the Project Welcome Page as shown in Figure 2–6. The different types of project views are described in Table 2–2.

Table 2–2 Project Views

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

2.4.2.2 Project Browser

The project browser provides a hierarchical view of the BPM repository, including projects and project folders as shown in the center of figure Figure 2–6.

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

2.4.2.3 Control Panel

The control panel enables you to control how projects and project folders are displayed in the project browser. The project browser control panel is shown in Figure 2–7.

Figure 2–7 The Project Browser Control Panel

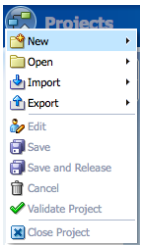
2.4.2.4 Search

The search field enables you to search for projects within the repository based on name, author, or description. The search field is available in the upper-right hand corner of the Project Welcome Page as shown in Figure 2–6.

2.4.3 Introduction to the Business Process Composer Main Menu

The application main menu provides access to frequently used commands and functionality. This menu is accessible from the Application Welcome page and the Project Welcome pages. It is accessible by clicking on the icon shown at the top of Figure 2–8.

Figure 2–8 The Application Main Menu as Viewed from the Project Welcome Page



The application main menu provides access to the menu items describe in Table 2–3.

Table 2–3 Main Application Menu Items

Menu item	Description
New	Enables you to create the following: ▪ BPM project. See Section 4.4.2, "How to Create a New Project" for more information. ▪ Simulation ▪ Service
Open	Provides the following options: ▪ Open a Project Enables you to open a project stored in the BPM repository. See Section 4.4.4, "How to Open a Project Using the Main Menu" for more information. ▪ Open a Deployed Project Enables you to open a project deployed to the BPM runtime environment. You can edit the business rules at runtime. See Chapter 13, "Using Oracle Business Rules" for more information.
Import	Provides functionality for importing projects and process models into Business Process Composer. Imported projects are stored in the BPM repository. See Section 8.1, "Importing and Exporting Projects" for more information.
Export	Enables you to export projects to the local file system. See Section 8.1, "Importing and Exporting Projects" for more information.
Deployment	Provides the following options: ▪ Deploy project ▪ Generate project SAR file ▪ Generate deployment plan See Chapter 15, "Deploying a BPM Project" for more information.
Process Report	Enables you to generate a report which displays details about the processes within your project.
Process Player	Launches the process player window. See Chapter 7, "Using Process Player" for more information.
Edit	Switches the current project to edit mode.
Save	Saves changes made to the current project.

Table 2–3 (Cont.) Main Application Menu Items

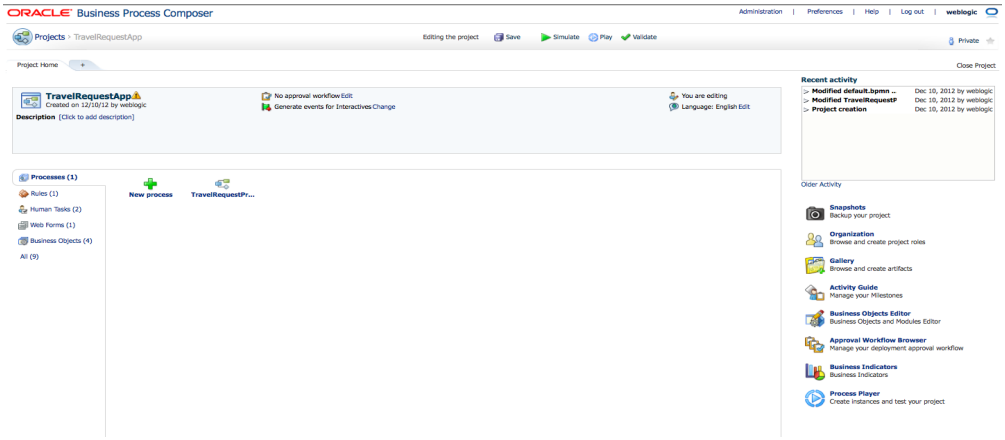
Menu item	Description
Share	Enables you to configure the project to be shared with other users.
Cancel	Releases the lock held on the project without saving changes.
Validate project	Validates the project.
Close project	Closes the current project.

2.4.4 Introduction to the Project Welcome Page

When you open a BPM project, the Project Welcome Page displays by default. The Project Welcome Page provides access to all the resources within a BPM project.You can view the properties and status of a project. You can also create and work with business processes and other project components.

Figure 2–9 shows the Project Welcome Page for the Sales Quote example project.

Figure 2–9 The Project Welcome Page



See Section 4.2, "Introduction to the Project Welcome Page" for information on using the Project Welcome Page.

3

Walkthrough: Creating a Basic BPM Application

This chapter provides a general outline of the steps required to create a BPM project, create, design, and test BPMN processes, and deploy a BPM project to run time. It provides links to other chapters containing additional information and procedures for performing the required tasks.

This chapter contains the following sections:

- Section 3.1, "Setting Up a BPM Project"
- Section 3.2, "Modeling a BPMN Process"
- Section 3.3, "Implementing a BPMN Process - Defining User Interaction"
- Section 3.4, "Implementing a BPMN Process - Creating a Business Rule"
- Section 3.5, "Implementing a BPMN Process - Defining a Service"
- Section 3.6, "Defining the Data Used by Your Process"

3.1 Setting Up a BPM Project

Before you can begin modeling a business process, you must first set up a BPM project. The following sections describe how to login to Oracle Business Process Composer to create and save a BPM project.

BPM projects contain all of necessary components required by a business application. This includes the BPMN processes that form the core of an Oracle BPM application and the technical components used to connect the application to other processes, databases, etc. See Section 4.1, "Introduction to Oracle BPM Projects" for more information.

The following sections describe how to login to Business Process Composer, create a new BPM project and create the BPMN process that will define the behavior of the travel request application.

3.1.1 Login to Business Process Composer

Before creating a BPM project, you must login to Business Process Composer using the following information provided by your system administrator.

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

To Login in to Oracle Business Process Composer

1. Go to the Business Process Composer URL.

   Figure 3–1 shows the sign-on screen that appears after the Oracle BPM application loads.

Figure 3–1 Oracle Business Process Composer Sign-on Screen

Business Process Composer

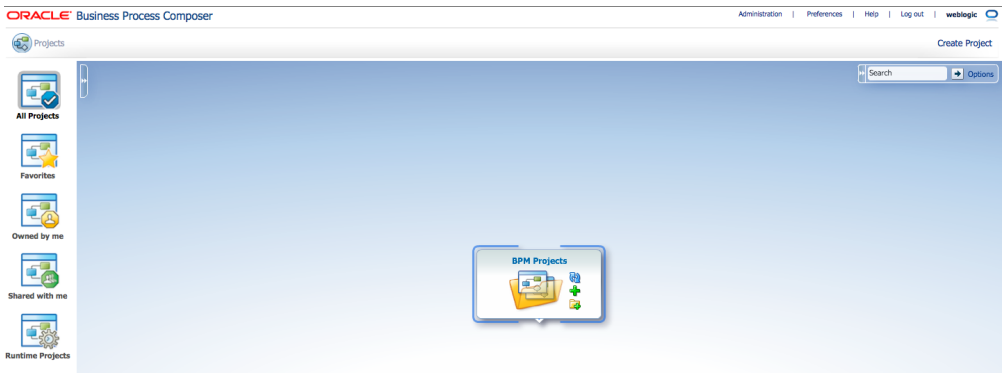
****

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

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

After logging in, Business Process Composer displays the Application Welcome page. as shown in Figure 3–2.

Figure 3–2 The Application Welcome Page



See Section 2.4, "Introduction to the Business Process Composer User Interface" for a description of the Application Welcome page and the Business Process Composer user interface.

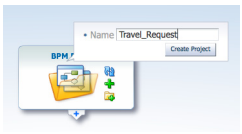
3.1.2 Create and Open a BPM Project

After logging into Business Process Composer, the first step in creating a BPM application is to create a new BPM project. You can create a new project directly from the Application Welcome page.

To create and open a BPM project:

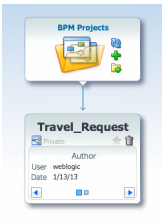
1. Click the **New Project** (+) icon.
2. Enter "Travel_Request" then click **Create Project** as shown in Figure 3–3.

Figure 3–3 Creating a New Project from the Application Welcome Page



The new project appears in the hierarchical project tree as shown in Figure 3–4.

Figure 3–4 The Travel Request Project in the Application Welcome Page



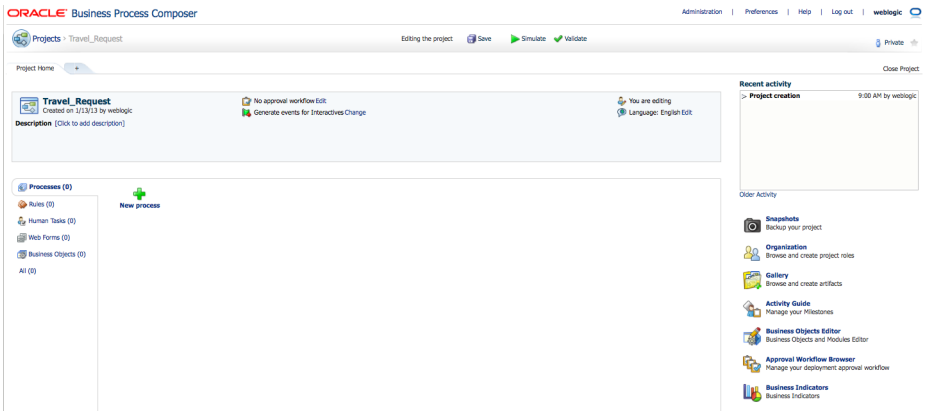
The project tree lists all the project you created or have access to view. See Chapter 4, "Working with BPM Projects" for more information about working with BPM projects.

3. To open the Travel Request project, click its name in the project tree.

When you open a BPM project the Project Welcome page appears. The Project Welcome page enables you to create, view, and edit the processes and components required for a fully-functioning BPM application.

Figure 3–5 shows how the Project Welcome page appears for a new BPM project.

Figure 3–5 The Project Welcome Page after Creating a New BPM Project



See Chapter 4, "Working with BPM Projects" for general information about BPM projects.

3.1.3 Create a BPMN Process

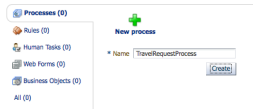
After opening a BPM project, you can create the resources, including BPMN processes, required by the travel request application. In the Travel Request application you must create a BPMN process called TravelRequestProcess.

One definition of a business process is a sequence of tasks that result in a well-defined outcome. A BPMN process defines the tasks performed by your application using flow objects. See Section 5.1, "Introduction to Business Processes" for general information about business processes in Oracle BPM.

To create a new BPMN process:

1. In the Project Welcome page, select the **Processes** tab.
2. Click **New Process**, then enter TravelRequestProcess as shown in Figure 3–6.

Figure 3–6 Creating the Travel Request Process



3. Click **Create**.

After creating the process, it appears the **Processes** tab. Additional BPMN processes that you create also appear under this tab.

3.1.4 Save Your BPM Project

When editing a BPM project, you can save your changes at any time. Business Process Composer saves BPM projects in the BPM repository. The BPM repository is a shared repository that enables you to collaborate with other Business Process Composer and Oracle BPM Studio users.

See Section 4.1.2, "Introduction to the Oracle BPM Repository" for more information.

To save a BPM project to the BPM repository:

1. In the Project toolbar, click **Save** as shown in Figure 3–7.

Figure 3–7 The Project Toolbar

Saving a project is the final step in setting up a basic BPM project. After saving your project you can begin modeling your business process by adding flow objects and sequence flows. Section 3.2, "Modeling a BPMN Process" describes how to use these elements to define the specific behavior of the travel request application.

3.2 Modeling a BPMN Process

BPMN is a standard format for defining process flows.

Sequence flows define the order in which these tasks are performed. See Chapter 5, "Working with Processes and the Process Editor" for general information about creating business processes using Business Process Composer. See Appendix A,

"BPMN Flow Object Reference" for information about the BPMN flow objects supported by Oracle Business Process Management.

In the travel request approval application, these tasks include each of the steps involved in creating and approving the travel request. This includes both the end user's interaction with the application and any required automated tasks like updating a database.

3.2.1 Open the Business Process

You can open a business process directly from the Project Welcome page.

To open a business process:

1. From the Project Welcome page select the **Processes** tab.
2. Click TravelRequestProcess.

The process opens in the process editor as shown in Figure 3–8.

Figure 3–8 The Process Editor after Creating a New Process

The process editor enables you to add flow objects and sequence flows to a process. See Section 5.2.2, "Introduction to the Process Editor Canvas" for information about the process editor. See Appendix A, "BPMN Flow Object Reference" for information about the BPMN flow objects supported by Oracle BPM.

When you create a new BPMN process, by default it is created with a none start and end event. See Appendix A.2.3, "Introduction to the None Start Event" for more information.

3.2.2 Add a User Task

After creating and opening a business process, you can begin adding the BPMN flow objects that define the process flow. The first activity required by the Travel Request application is to enable users to submit a travel request.

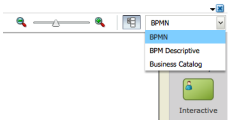
The user task is the BPMN flow object used to define user interaction. You can add a user task to your process by dragging and dropping it from the BPMN component palette.

See Appendix A.3.2, "Introduction to the User Task" for more information about the user task. See Section 5.2.3, "Introduction to the BPMN Component Palette" for information about the component palette.

To add a user task to your process:

1. In the BPMN component palette, select **BPMN** from the drop down list as shown in Figure 3–9.

Figure 3–9 Selecting the BPMN Component Palette



2. Double-click **Interactive** in the component palette.

   For the initial task of the travel request process, you must add an initiator task. Initiator user tasks are used to begin a business process. Initiator tasks create a process instance and are the first user task within a process flow

3. Click the **Initiator** icon, then drag it onto the process editor canvas to a point on the sequence flow between the start and end events of the process.
4. When the sequence flow begins flashing, click again to add the task to your process. After adding the user task you process appears as shown in Figure 3–10.

Figure 3–10 The Travel Request Process After Adding a User Task

   When adding a flow object to a sequence flow in this manner, Business Process Composer automatically connects the sequence flows as incoming and outgoing sequence flows.

5. Move the mouse over the newly added task, then click the **Edit** icon.
6. In the **Name** field, change the name to Submit Request.
7. Click outside of the properties popup to record your changes.
8. Save your project.

Adding the Submit Request user task defines the first stage of the travel request process. Section 3.3, "Implementing a BPMN Process - Defining User Interaction" describes how to define the user interface implemented by this user task.

3.2.3 Save Your Project

As you continue adding BPMN flow objects to your process, you should continually save your changes. You save changes to your process by saving the BPM project. See Section 3.1.4, "Save Your BPM Project" for more information.

3.2.4 Create a New Role

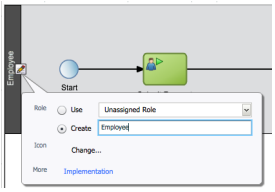
Roles define the users or groups of users that are responsible for performing the tasks defined by your process. They also define who has access to these tasks. Within a BPMN process the roles correspond to the swimlanes that extend horizontally across the process editor canvas.

The travel request process requires a role for an employee who is responsible, and also has permission, to create a travel request. To add this role, you must create a role called Employee that contains the Submit Request user task.

To create the Employee role in your process:

1. Move the cursor over the Unassigned Role text at the top left of the process editor canvas, then click the **Edit** icon.
2. Select **Create**, then enter "Employee" as shown in Figure 3–11.

Figure 3–11 Creating a New Role in the Travel Request Process



3. Click outside the properties popup to record your changes.
4. Save your project.

The Employee role is created and appears as the title of the swimlane. After creating a role, you can use it again in other swimlanes within the same process. Roles can also be shared by other processes within the same BPM project.

When deploying a project, a process administrator maps the roles defined in your BPM project to the real-world users within your organization using Process Workspace.

3.2.5 Add a Business Rule Task

In the travel request application, if an employee submits a travel request that is under a certain amount, it does not require manual approval. To model this within a BPMN process, you can use a business rules task to calculate if the given request is above or below a certain amount.

A business rule task is a BPMN flow object that enables you to implement a business rule within a process. In the travel request application, a business rule performs a check on the value of a data object which stores information about the travel request.

In this context, creating the business rules and data objects is performed as part of the implementation of a business process.Section 3.4.2, "Create a New Business Rule and Define Input and Output Data Objects" describes how define a business rule for the travel request application.

To add a business rule task to your process:

1. In the BPMN component palette, select BPMN from the drop down list.
2. Double-click **Activity**.
3. Click the **Business Rule** icon and drag it to a point on the sequence flow between the Submit Request user task and the end event.
4. When the sequence flow begins flashing, click again to add the task to your process.
5. Move the mouse over the newly added task, then click the **Edit** icon.

6. In the **Name** field, change the name to Approval Rules.
7. Click outside the properties popup to record your changes.
8. Save your project.

After adding and editing the business rule task, you process should appear as shown in Figure 3–12.

Figure 3–12 The Travel Request Process after Adding a Business Rule Task

3.2.6 Add a New Swimlane and Create the Manager Role

In addition to the Employee role, the travel request process requires a role for managers who are responsible for approving the travel request. After creating a new role, you must also create a new swimlane within the BPMN process. Business Process Composer enables you to create a new swimlane quickly by dragging an activity to a blank area of the process editor canvas.

To add a new swimlane and create the manager role:

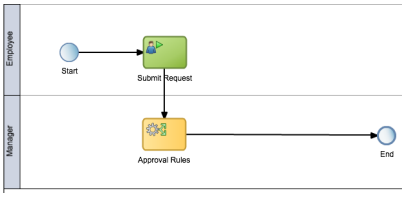
1. Double-click the business rule task, then drag it to an area below the Employee swimlane.
2. Release the mouse.

   Business Process Composer creates a new swimlane. By default, new swimlanes are not assigned a role.
3. Move the cursor over the Unassigned Role text at the left of the process editor canvas, then click the **Edit** icon.
4. Select **Create**, then enter "Manager."
5. Click outside the properties popup to record your changes.

   The Employee role is created and appears as the title of the swimlane. After creating a role, you can use it again in other swimlanes within the same process. Roles can also be shared by other processes within the same BPM project.
6. Double-click the end event and drag it to the Manager swimlane.
7. Save your project.

After creating a adding a new swimlane and creating the Manager role, your process should appear as shown in Figure 3–13.

Figure 3–13 The Travel Request Process after Adding the Manager Role



3.2.7 Add an Exclusive Gateway

In the travel request process, a business rules task performs a check on the amount of a travel request. However, the actual path a token takes through a business process is determined by a gateway. Gateways are BPMN flow objects that determine the path through a business process based on certain conditions, usually the value of data objects. See Section A.7, "Controlling Process Flow Using Gateways" for more information about the gateways supported by BPMN.

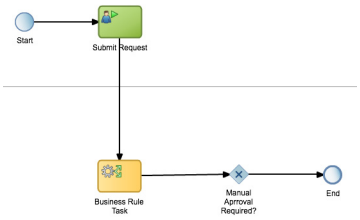
Within the travel request process there are two possible paths depending on the amount of the travel request. Choosing between two possible paths, where only one option is possible, is performed by an exclusive gateway.

To add a exclusive gateway to your process:

1. In the BPMN component palette, select BPMN from the drop down list.
2. In the component palette, double-click **Gateway**.
3. Double-click the **Exclusive Gateway** icon, then move the mouse to a point on the sequence flow between the business rules task and the end event.
4. When the sequence flow begins flashing, click again to add the exclusive gateway to your process.
5. Double-click the text of the exclusive gateway, then enter "Manual Approval Required?"
6. Hit Enter to record your changes.
7. Save your project.

After adding an exclusive gateway, your process should appear as shown in Figure 3–14.

Figure 3–14 The Travel Request Process After Adding and Exclusive Gateway



Later in the design phase of the travel request application you add an additional sequence flow. Section 3.2.11, "Add a Sequence Flow to Handle Rejected Travel Requests" describes how to add the conditional sequence flow that is followed if the travel request does not require manual approval.

When defining the data objects used by the BPMN process must also define the expressions that are used to evaluate which outgoing sequence flow the process will follow. See Section 3.6, "Defining the Data Used by Your Process" for more information.

3.2.8 Add the Approve Request User Task

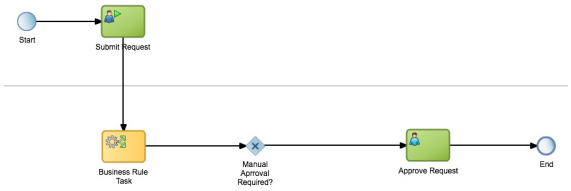
In the travel request process, if manual approval is required, you must add an additional user task that enables a user to approve or reject the travel request.

To add the approval user task:

1. In the BPMN component palette, select **BPMN** from the drop down list.
2. Double-click **Interactive** in the component palette.
3. Double-click the **User Task** icon, then drag it onto the process editor canvas to a point on the sequence flow between the exclusive gateway and the end event.
4. When the sequence flow begins flashing, click again to add the task to your process.
5. Move the mouse over the newly added task, then click the **Edit** icon.
6. In the **Name** field, change the name to Approve Request.
7. Click outside of the properties popup to record your changes.
8. Save your project.

After adding an addition user task to handle approval of the travel request, you process you appear as shown in Figure 3–15.

Figure 3–15 The Travel Request Process after Adding the Approval User Task



3.2.9 Add an Additional Exclusive Gateway

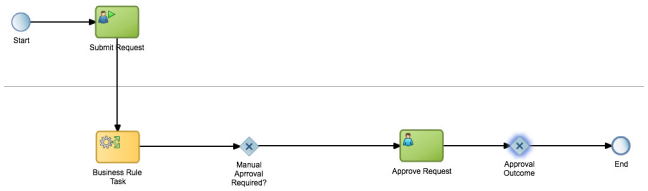
Within the travel request application, if a travel request is rejected, process flow should return back to the initial step to enable users to enter another travel request. To model this in BPMN, you can add an additional exclusive gateway to handle the approval outcome.

To add an additional exclusive gateway:

1. In the BPMN component palette, select BPMN from the drop down list.
2. In the component palette, double-click **Gateway**.
3. Double-click the **Exclusive Gateway** icon, then move the mouse to a point on the sequence flow between the Approve Request user task and the end event.
4. When the sequence flow begins flashing, click again to add the exclusive gateway to your process.
5. Move the mouse over the newly added task, then click the **Edit** icon.
6. In the **Name** field, change the name to "Approval Outcome."
7. Click outside of the properties popup to record your changes.
8. Save your project.

After adding the exclusive gateway to handle the approval outcome, your process should appear as shown in Figure 3–16.

Figure 3–16 The Travel Request Process after Adding an Additional Exclusive Gateway



3.2.10 Add a Service Task to Save the Travel Request in a Database

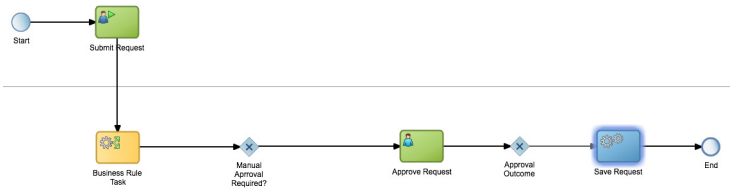
The final requirement for the travel request application is to save the travel request in a database. To model connectivity with other systems like databases, BPMN uses the service task. During the design phase you can add a service task to your process model and later define the technical implementation details of the service.

To add a service task:

1. In the BPMN component palette, select BPMN from the drop down list.
2. Double-click **Activity**.
3. Double-click the **Service Task** icon and drag it to a point on the sequence flow between the Approval Outcome exclusive gateway and the end event.
4. When the sequence flow begins flashing, click again to add the task to your process.
5. Move the mouse over the newly added service task, then click the **Edit** icon.
6. In the **Name** field, change the name to Save Request.
7. Click outside the properties popup to record your changes.
8. Save your project.

After adding the service task, your process should appear as shown in Figure 3–17.

Figure 3–17 The Travel Request Process after Adding a Service Task



The service task is the final BPMN flow object required by travel request application. The process flow shown Figure 3–17 defines the basic flow of the travel request application. The following sections describe how to add and configure sequence flows to handle alternate paths through your process.

3.2.11 Add a Sequence Flow to Handle Rejected Travel Requests

The first alternative path through your process to consider is the case where a travel request is rejected. In Section 3.2.9, "Add an Additional Exclusive Gateway" you added the exclusive gateway to determine if the travel request is rejected or approved. Now,

you must add a sequence flow to handle rejected travel requests. This sequence flow returns the process instance to the Submit Request user task.

To add a new sequence flow:

1. Move the cursor over the Approval Outcome exclusive gateway.

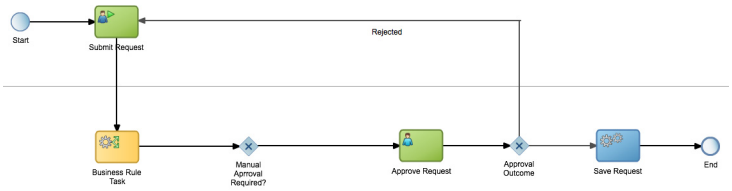
   The **Edit** and **Add Sequence Flow** icons appear as shown in Figure 3–18.

Figure 3–18 Adding a New Sequence Flow

2. Click the **Add Sequence Flow** icon and drag the mouse to the Submit Request user task.
3. When the Submit Request user task begins flashing, click the mouse to add the sequence flow.
4. Click and drag the sequence flow to reposition it within the process editor canvas.
5. Move the mouse over the sequence flow and click the **Edit** icon.
6. Enter Rejected in the **Name** field and **Conditional** from the drop down list.

After adding the sequence flow to handle rejected travel requests, your process should appear as shown in Figure 3–19.

Figure 3–19 The Travel Request Process after Adding the Rejected Sequence Flow



3.2.12 Edit and Add Sequence Flows for Default Approval

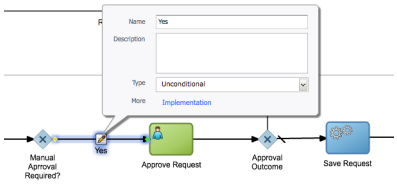
The second alternative path required by the travel request application handles cases where manual approvals are not required. In this case, the request is saved in the database. To handle this cased, you must perform the following:

- Edit the default outgoing sequence flow of the Manual Approval Required? exclusive gateway.
- Add an additional sequence flow from the exclusive gateway to the Save Request service task.

To handle default approval:

1. Move the mouse over the sequence flow between the Manual Approval Required gateway and the Approve Request user task, then click the edit icon.
2. Change the name to "Yes" and select **Unconditional** from the drop down list as shown in Figure 3–20.

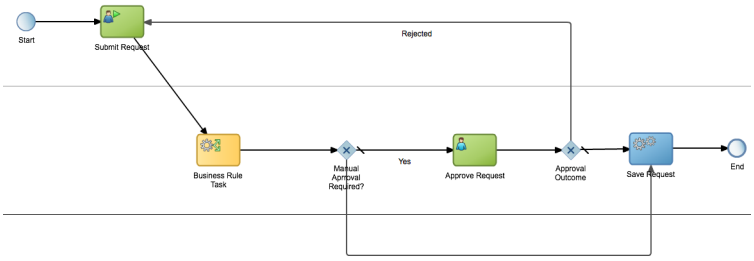
Figure 3–20 Editing the Sequence Flow of the Approval Outcome Gateway



3. Click outside of the properties popup to record your changes.
4. Move the mouse over the **Manual Approval Required?** exclusive gateway and click the Add Sequence Flow icon.
5. Move the mouse over the Save Request service task.
6. When the service task begins flashing, click to add the sequence flow.
7. Click and drag the sequence flow to change its position.

After editing and adding the sequence flows for handling approval of the travel request, you process should appear as shown in Figure 3–21.

Figure 3–21 The Finished Travel Request Process



3.2.13 Create a Project Snapshot

You can create a back of your project by creating a project snapshot. Project snapshots are stored in the BPM repository and enable you to revert back to older versions and revisions.

To create a project snapshot:

1. Go to the Project Welcome page, then click **Snapshots**.

   The Snapshots browser appears.
2. Click the Add (+) icon, then enter "TravelRequestVersion1" in the **Name** field.
3. Click **Create**.

   The snapshot appears in the list, which also displays the date and user who created the snapshot.
4. Click **OK** to close the snapshot browser.

From the Snapshot browser you can open previous versions of your project.

3.3 Implementing a BPMN Process - Defining User Interaction

After designing the general flow of your business process, you can begin adding specific implementation details for the flow objects. This section walks you through the process for defining how users interact with your process. It describes how to create a human task and how to create and configure web forms.

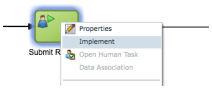
3.3.1 Create and Open a Human Task

A human task is a type of service in Oracle BPM that enables you to define how users interact with your BPM applications. In additional to the technical and runtime functionality, human tasks also enable you to define the user interface components of your application. See Chapter 11, "Working with Human Tasks" for more information about using human tasks in Oracle BPM.

To create and open human task:

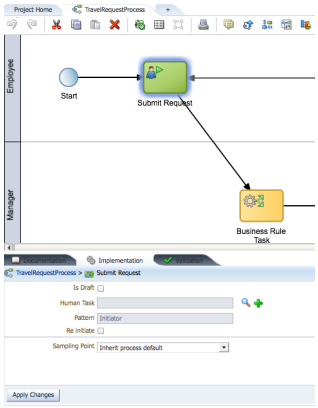
1. Open the TravelRequest process created in Section 3.2, "Modeling a BPMN Process"
2. Right-click the Submit Request user task, then select **Implement** as shown in Figure 3–22.

Figure 3–22 Selecting the Implement OPtion from a User Task



The Implementation editor appears as shown in Figure 3–23.

Figure 3–23 The Implementation Editor



3. Click the **Add** (+) icon next to the **Human Task** text box.
4. Enter "SubmitRequest" in the **Name** field, then click **Create**.
5. Click **Apply Changes**.
6. Close the Implementation editor by clicking the **Collapse Pane** arrow on the right-hand side of the window.
7. Right click the Submit Request user task, then select **Open Human Task**.

The human task opens in the human task editor. See Section 11.2, "Introduction to the Human Task Editor" for more information. After creating a human task it also appears under the **Human Task** tab in the Project Welcome page. You can also create and open human tasks directly from the Project Welcome page.

3.3.2 Create a Web Form

Web forms define the user interface components of your process-based application. Web forms are associated to a human task. See Section 9.1, "Introduction to Forms in Oracle BPM" for more information about working with web forms in Oracle BPM.

Using the web form editor, you can create the layout of a form by dragging and dropping web form components onto the form canvas.

To create a web form:

1. From the human task editor, select the **Basic** tab.
2. Under **Presentation**, select **Web Form**, then click the **Add** (+) icon.
3. Enter "TravelApprovalForm" in the **Name** field and click **Create**.
4. Click the **Edit** (pencil) icon.

The web form opens in the web form editor.

3.3.3 Add Basic Controls to Your Web Form

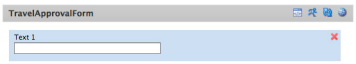
The first step in creating a web form is to begin adding web form controls to the web form canvas. You can add form controls by clicking and dragging them from the form controls palette to the canvas.

To add a control to a web form:

1. From the web form controls palette, click and drag a Text web form control onto the rectangular area of form canvas. When the green position icon displays, click again to add the web form control.

   See Section 9.7.1, "How to Add Controls to a Web Form" for more information about adding web form controls. After adding the control, your web form should appear as shown in Figure 3–24.

Figure 3–24 The Travel Approval Form after Adding a Text Web Form Control

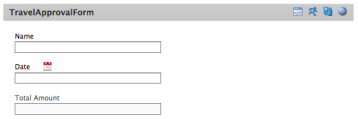


2. In the web form canvas, select the text control you just added.
3. In the **Properties** editor, enter "Name" in the **Label** property.
4. From the web form controls palette, click and drag a Date web form control onto the form canvas to a position towards the bottom of the "Name" text control. When a green down arrow appears, click again to add the web form control.
5. In the web form canvas, select the date control you just added. When a control is selected, it is highlighted in blue.
6. In the **Properties** editor, enter "Date" in the **Label** property.

7. From the web form controls palette, click and drag a Money web form control onto the form canvas to a position toward the bottom of the "Date" date control. When a green down arrow appears, click again to add the web form control.
8. In the web form canvas, select the money control you just added.
9. In the **Properties** editor, enter "Total Amount" in the **Label** property.
10. Deselect the Enabled property.
11. Save your project.

After adding the Name, Date, and Total Amount controls, your web form should appear as shown in Figure 3–25.

Figure 3–25 The Travel Approval Web Form after Adding Basic Web Form Controls



3.3.4 Add a Tabs Control to Your Web Form

Some web form controls are designed to organize or group other web form controls. To make the travel approval web form easier to use, you can add a tabs control which creates a tabbed pane within a web form.

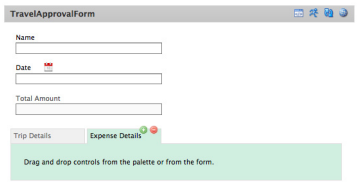
To add a tab control to a web form:

1. From the web form controls palette, click and drag a Tabs web form control onto the form canvas to a point below the "Total Amount" web form control. When the cursor displays a green down arrow, click again to add the tab control.

   When you add a tab control to a form, it automatically creates three tabs. However, for this web form, you only need two tabs.
2. Select **Tab 2**, then click the **Delete** (-) icon.
3. Select **Tab 0**.
4. In the Properties editor, enter "Trip Details" in the **Name** field
5. Select **Tab 1**.
6. In the Properties editor, enter Expense Details in the Name field.

After adding a tabs web form control, your web form should appear as shown in Figure 3–26.

Figure 3–26 The Travel Approval Web Form after Adding a Tabs Web Form Control



3.3.5 Add Additional Controls to Each Tab

To complete the layout of the travel approval web form, you can add additional controls to each tab.

To add controls to a tab control:

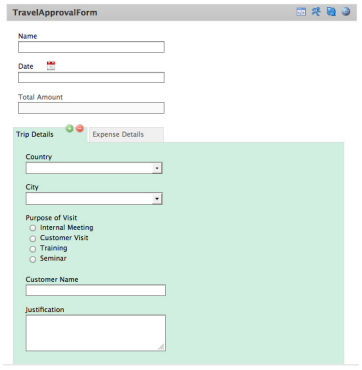
1. Select the "Trip Details" tab.
2. From the web form controls palette, click and drag each of the controls shown in Table 3–1onto the "Trip Details" tab in the web form canvas. In addition to adding each control, you should also customize the control by editing its properties using the **Settings** and **Styles** tabs of the Properties editor.

Table 3–1 Web Form Control Settings for the Trip Details Tab

Control Name	Control Type	Settings	Styles
Country	Dropdown	Label = "Country"	none
City	Dropdown	Label = "City"	none
Purpose of Visit	Radio	Option 1=Internal Meeting Option 2=Customer Visit Option 3=Training Option 4=Seminar	Item width=25%
Customer Name	Text	Label= "Customer Name" Deselect Visible	None
Justification	Text Area	Label = "Justification"	Width = 80%

After adding each of the web form controls to the Trip Details tab, your form should appear as shown in [Fig x.x].

Figure 3–27 The Travel Approval Web Form Showing the Trip Details Tab



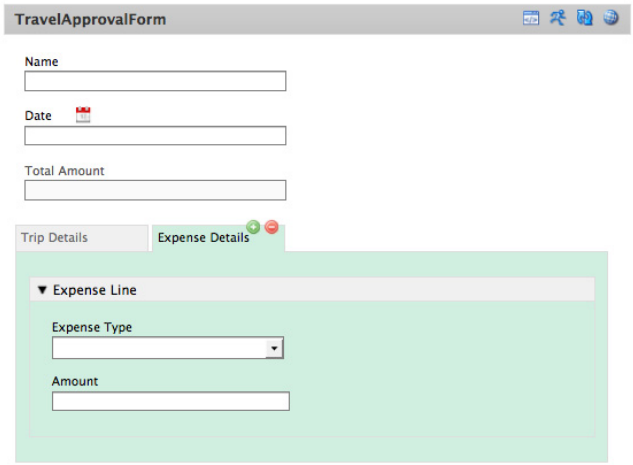
3. Select the Expense Details tab.
4. From the web form controls palette, click and drag each of the controls shown in Table 3–2 onto the "Expense Details" tab in the web form canvas. In addition to adding each control, you should also customize the control by editing its properties using the **Settings** and **Styles** tabs of the Properties editor.

Table 3–2 Web Form Control Settings for the Expense Details Tab

Control Name	Control Type	Settings	Styles
Expense Line	Section	Label = "Expense Line"	none
Expense Type	Dropdown	Label = "Expense Type" Option 1=Air Option 2=Cab Option 3=Train Option 4=Hotel Option 5=Other	none
Amount	Money	Label = "Money"	

After adding each of the web form controls to the Expense Details tab, your finished form should appear as shown in Figure 3–28.

Figure 3–28 The Travel Approval Web Form Showing the Expense Details Tab



3.3.6 Test Your Web Form

After creating the layout of your web form and adding the required web form controls, you can test its behavior directly from the web forms editor.

To test a web form:

1. In the web form editor toolbar, click the **Test Web Form** button as shown in Figure 3–29.

Figure 3–29 The Web Form Toolbar

   The web form is displayed in a popup window. You can the data input controls by enter values in each field. Fields that have invalid data display errors.

2. Click the **Close** button when you are done testing the form.

3.3.7 Add a Form Rule to Calculate the Total Expense

Form rules enable you to add dynamic behavior to a web form. The TravelRequest web form uses a form rule to calculate the total expense of the travel request.

To add a form rule for calculating total expense:

1. In the web forms editor toolbar, click the **Form Rules** button.
2. Click the **Add** (+) button.
3. Click the **Edit Rule** button.
4. In the **Name** field, enter "Total Expense."
5. Enter the following text in the **Rule** text area:

```
var total = 0;
for (var i = 0; i < Amount.value.length; i++) {
    total = total + Amount[i].value;
}
TotalAmount.value = total;
```

6. Click the **Test Web Form** button as shown in Figure 3–29.

3.3.8 Add a Form Rule to Add Dynamic Behavior to Your Form

In the TravelRequest application, you can create form rules to dynamically enable the customer field if the purpose the visit is set to "Customer Visit.

To add a form rule to add dynamic behavior to a form:

1. In the web form rules editor, click the **Create New Form Rule** (+) button.
2. Click the **Edit** button for the form rule.
3. In the **Name** field, enter "Enable Customer."
4. Enter the following text in the **Rule** text area:

```
if (PurposeOfVisit.value == 'Customer Visit') {
   CustomerName.visible = true;
}
else {
   CustomerName.visible = false;
}
```

5. Click the **Test Web Form** button as shown in Figure 3–29.

3.3.9 Add a Form Rule to Dynamically Populate a List of Options

The second form rule required by the Travel Request application is a form rule that dynamically populates a list of options.

To add a form rule to dynamically populate a list of options:

1. In the web form rules editor, click the **Create New Form Rule** (+) button.
2. Click the **Edit** button for the form rule.
3. In the **Name** field, enter "Country Dropdown."
4. Enter the following text in the **Rule** text area:

```
if (form.load) {
```

```
    url = 'http://localhost:7001/LOVHelper/jersey/listvalues/countries';
    eval ('x=' + http.get(url));
    Country.options = x;
}1
```

5. Click the **Test Web Form** button as shown in Figure 3–29.

3.4 Implementing a BPMN Process - Creating a Business Rule

In Section 3.3, "Implementing a BPMN Process - Defining User Interaction" you defined the user interface of the travel request application by implementing the human task and web form associated with the users tasks. This section walks you through the process of creating the business rules required by the travel request application and how to assign a business rule to a business rule task flow object.

3.4.1 Create Business Objects

Before creating a business rule, you must create the data objects that define the input and output arguments of the business rule. Data objects enable you to define the data structures used by your application. See Section 12.2, "Introduction to Data Objects" for more information.

To create a business object:

1. From the Project Welcome page, select **Business Object** tab, then click **New Business Object**.
2. Enter RuleInput in the **Name** field, then click the **Add** (+) icon next to **Parent Module**.
3. Enter BusinessData, then click **OK**.
4. Select BusinessData from the drop down list, then click **Create**.
5. Click **New Business Object**.
6. Enter RuleOutput in the **Name** field, then select BusinessData from the **Parent Module** drop down list.
7. Click **Create**.

After creating the data objects, you must configure the data objects by adding attributes which define the data types used within the data objects.

To configure a business objects:

1. From the Project Welcome page, select the **Business Object** tab, then click RuleInput.
2. In the data objects tree, select RuleInput.
3. Click the **Add** menu, then select **Add New Attribute**.
4. Enter "amount" in the **Name** field, then select **Real** from the drop down list.
5. Click **OK**.

   The amount attribute appears in the data objects tree under RuleInput.
6. In the data objects tree, select RuleOutput.
7. Click the **Add** menu, then select **Add New Attribute**.

8. Enter manualApproval in the **Name** field and select **Bool** from the drop down list.
9. Click **OK**.

   The manualApproval attribute appears in the data objects tree under RuleOutput.
10. Save your project.

The data objects defined in this section are relatively simple. In a real-world application, a data object may contain multiple modules each with multiple data types.

3.4.2 Create a New Business Rule and Define Input and Output Data Objects

After defining the business objects for the input and output data, you can create a new business rule.

To create a business rule and define input and output data objects:

1. From the Project Welcome page, select the Rule tab, then click **New Business Rule**.
2. Enter ApprovalRule in the **Name** field.
3. Add an input data object.
   a. Click the **Add** (+) icon.
   b. Select **Input** from the drop down list to the right of the **Name** field.
   c. Enter "ruleInput" in the **Name** field.
   d. From the **Type** drop down list, select **<component>**.

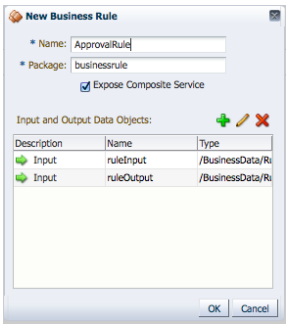
      The **Business Object** drop down list appears.
   e. From the **Business Object** drop down list, select /BusinessData/RuleInput.
   f. Click **Add**.

   The input data object appears in the table.
4. Add an output data object.
   a. Click the **Add** (+) icon.
   b. Select **Output** from the drop down list to the right of the **Name** field.
   c. Enter "ruleOutput" in the **Name** field.
   d. From the Type drop down list, select **<component>**.

      The **Business Object** drop down list appears.
   e. From the Business Object drop down list, select /BusinessData/RuleOutput.
   f. Click **Add**.

   The output data object appears in the table as shown in Figure 3–30.

Figure 3–30 The Create Business Rule Dialog Showing Input and Output Data Objects



5. Click **OK** to create the business rule.

   The business rule appears in the **Business Rules** tab of the Project Welcome page.

6. Save your project.

3.4.3 Create and Configure a Bucketset

After creating a new business rule, the first step in configuring it is to create and configure a bucketset. A bucketset defines a list of values or value ranges for different properties of the business rule. In the TravelRequest application, bucket sets define the range of values that determine whether a travel request must be manually approved or not.

To create and configure a bucketset:

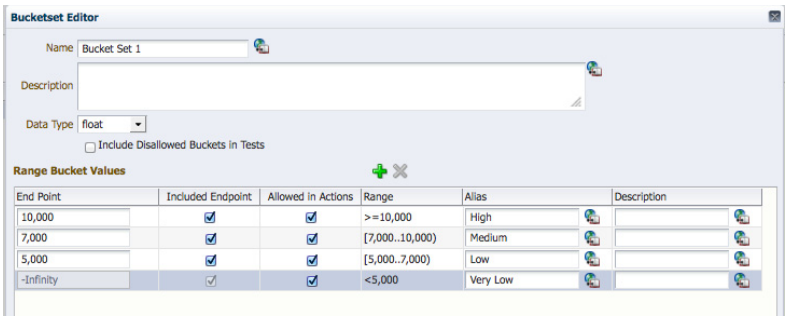
1. From the Project Welcome page, select the Rules tab, then click ApprovalRule.

   The rules editor appears. See Chapter 13, "Using Oracle Business Rules" for general information about working with business rules and the rules editor.

2. Select the **Bucketsets** tab.
3. Click the Add menu, then select **List of Ranges**. A new bucketset is created and appears in the table.
4. Select the bucketset from the table and click the **Edit Bucketset** icon.
5. Enter the following values for the bucketset:
   - **Name**: AmountRange
   - **Data Type**: float
6. Create buckets:
   a. Click the **Add Bucket** (+) icon.
   b. Enter 5000 in the **Endpoint** column.
   c. Enter "Low" in the **Alias** column.

   Add additional buckets with Endpoint values of 7000 and 10000. You can also update the alias column to reflect what the values represent. The bucketset table should appear as shown in Figure 3–31.

Figure 3–31 Buckets Defined for the Approval Rule



7. Click **OK** to finish configuring the bucketset.

See "Working with Facts and Bucketsets" in *Oracle Fusion Middleware User's Guide for Oracle Business Rules* for more information.

3.4.4 Create and Configure a Decision Table

After defining a bucketset, the next step in configuring a business rule is defining a decision table. A decision table provides a series of IF-THEN statements that determine the performance of a business rule. These statements are arranged in a table format similar to a spreadsheet.

See "Working with Decision Tables" in *Oracle Fusion Middleware User's Guide for Oracle Business Rules* for more information.

To create and configure a decision table:

1. From the Project Welcome page, select the Rules tab, then click ApprovalRule.
2. Click **Ruleset1**.
3. From the **Add** menu, select **Add Decision Table**.
4. Enter "Approval Decision Table" in the **Name** field.
5. Create and configure conditions:
   a. Select **Conditions** in the decision table.
   b. From the Add menu, select **Add Condition**.
   c. In the Condition browser, expand **RuleInputType**, then select **Amount**.
   d. Click **OK**.

      RuleInputType.amount appears in the table.
   e. Select RuleInputType.amount in the table, then select Amount_Ranges from the list of bucketsets.
   f. Click in the cell in column R1 that corresponds to RuleInputType.amount, then select the check boxes next to Low and Very Low.
6. Create and configure actions:
   a. Select **Actions** in the decision table.
   b. From the **Add** menu, select **Add Action**, then **assert new**.
   c. Under Facts, select **RuleOutputType**.
   d. Under **Properties**, check **Parameterized** and **Constant**.

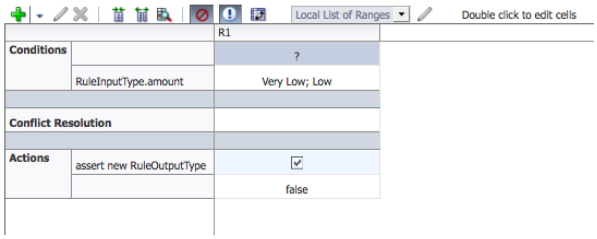
e. Click **OK**.

f. In the cell in column **R1** (which stands for Rule 1) that corresponds to assert new **RuleOutputType**, select the checkbox.

g. In the cell below, enter "false" in the text field.

After creating and configuring actions, the decision table should appear as shown in Figure 3–32.

Figure 3–32 The Decision Table After Creating and Configuring Actions



3.4.5 Assign the Business Rule to the Business Rule Task

After creating and configuring your business rule, you must assign it to a business rule task.

To assign a business rule to the business rule task:

1. From the Project Welcome page, select the **Processes** tab, then click the TravelRequest process.
2. Right-click on the Approval Rules business task, the select **Implementation**.
3. Next to the **Rule** field, click the **Browse** icon, then click ApprovalRule.
4. Click **Apply Changes**.

3.5 Implementing a BPMN Process - Defining a Service

During the design phase, you added a service task to model the step required to save the travel request to a database. The next step in implementing a BPMN process is to define the service used to connect to the database. Using Business Process Composer you can create new services.

To define a service and assign it to a service task:

1. From the Project Welcome Page open the TravelRequest process.
2. From the drop down list above the component palette, select **Business Catalog**.
3. Click the **Add Service** (+) icon.
4. Enter "SaveRequest" in the **Name** field.
5. From the **Type** drop down list select **Reference**.
6. Under **WSDL Service**, select **URL**, then enter a value in the text field. A typical value is:

```
http://localhost:7001/soa-infra/services/default/SaveRequest/saverequestsync_
client_ep?WSDL
```

The specific value you should use depends on your environment and should be provided by your system administrator. After you enter a URL, Business Process Composer connects to the URL to determine information about the WSDL. If this connection is successful, Business Process Composer populates the **Port Type** and **Callback Type** drop down lists.

7. From the **Port Type** and **Callback Type** drop down lists, select a value.
8. Click **OK**.
9. Click **Save**.
10. Open the TravelRequest process.
11. Right-click on the SaveRequest service task, then select **Implement**.
12. From the **Type** drop down list, select **Service Call**.
13. Click the SaveRequest service.
14. Select **Process** from the **Operation** drop down list.
15. Click **Apply Changes**.
16. Save your project.

3.6 Defining the Data Used by Your Process

When modeling and implementing a business process, you must define that data that is used by your application. To define the data required by the Travel Request application, you must define the data types required to store information about the travel request. These data types are defined by data objects. When a users enters information in the user interface defined by the web forms created during the implementation phase, this information is stored in the data objects.

You can also define the expressions used in your BPMN process. Expressions are used to evaluate and change the value of data stored in a data object.

3.6.1 Create Data Objects

Use the data object editor to create and edit data objects in Business Process Composer.

To create the data objects required by the travel request process:

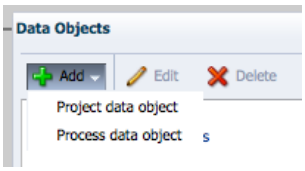
1. Open the TravelRequest process.
2. From the process editor toolbar, click the **Data Objects** button as highlighted in Figure 3–33.

Figure 3–33 The Process Editor Toolbar Highlighting the Data Object Button

The data objects editor appears.

3. From the **Add** menu, select **Process Data Object** as shown in Figure 3–34.

Figure 3–34 Creating a New Data Object in the Data Object Editor



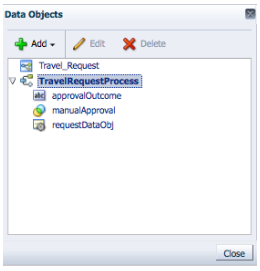
4. In the **Name** field, enter "requestDataObj."
5. From the Type drop down list, select **<component>**. After selecting this option, the **Business Object** drop down list appears.
6. Select Types.TravelApprovalForm from the drop down list, then click **OK**.

   The new data object is created and appears under then TravelRequest node. If you had created a project data object, it would have appeared under Travel_Request. See Section 12.2.2, "Introduction to Process and Project Data Objects" for information about project and process data objects.
7. Using the procedures above, create two additional process data objects using the parameters shown in the following table.

Name	Type
approvalOutcome	String
manualApproval	Boolean

   When you have finished creating these data objects, the data object editor should appear as shown in Figure 3–35.

Figure 3–35 The Data Objects Editor Showing Process Data Objects



8. Click **Close**.

3.6.2 Define Expressions and Conditions

Expressions enable you to evaluate the data stored in a data objects and change its value if required. The Travel Request process uses an expression to

To define an expression for the manual approval sequence flow

1. Move the mouse over the **No** sequence flow, then click the **Edit** button.
2. Click **Implementation**.
3. Click **Edit** next to the Condition text field.

The expression editor displays.

4. In the expression editor enter manualApproval==false.
5. Click **Validate** to ensure the expression is valid, then click **OK**.
6. Click **Apply Changes** in the Implementation editor.

To define an expression for the rejected sequence flow:

1. Move the mouse over the **Rejected** sequence flow, then click the **Edit** button.
2. Click **Implementation**.
3. Click Edit next to the Condition text field.
4. In the expression editor, enter approvalOutcome=="REJECT".
5. Click **Validate** to ensure the expression is valid, then click **OK**.
6. Click **Apply Changes** in the Implementation editor.

3.6.3 Define Data Associations

The final step in defining how data is used in your business process is the define the data associations. Data associations define how information is passed between one BPMN flow object to another.

To define the data associations for a human task:

1. Right-click on the **Submit Request** human task, then select **Data Associations**

   The Data Associations edit displays.
2. In the Data Objects pane on the right-hand side, expand Data Objects.
3. Click and drag requestDO to the Outputs text field for the Submit Request human task.
4. Also, click and drag requestDO to the Input text field for the Submit Request human task.
5. Click **Apply**.

Part II

Modeling and Testing Business Processes

This part describes how to use Business Process Composer to model your business processes. It includes a general overview of the application.

This part contains the following chapters:

- Chapter 4, "Working with BPM Projects"
- Chapter 5, "Working with Processes and the Process Editor"
- Chapter 6, "Simulating Process Behavior"
- Chapter 7, "Using Process Player"
- Chapter 8, "Working with the Project Life Cycle"

4

Working with BPM Projects

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

This chapter includes the following sections:

- Section 4.1, "Introduction to Oracle BPM Projects"
- Section 4.2, "Introduction to the Project Welcome Page"
- Section 4.3, "Introduction to Project Sharing and Collaboration"
- Section 4.4, "Creating and Working with Projects"
- Section 4.5, "Using Guided Business Processes to Create Project Milestones"
- Section 4.6, "Defining the Roles Used in a Project"

4.1 Introduction to Oracle BPM Projects

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

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

BPM projects promote collaboration between process analysts and process developers. Business Process Composer and Oracle BPM Studio users can share projects using the BPM repository. See Section 4.1.2, "Introduction to the Oracle BPM Repository" for more information.

You can validate BPM projects and deploy them to runtime using Oracle Business Process Composer. See Chapter 15, "Deploying a BPM Project" for more information.

4.1.1 Introduction to Project Components and Resources

Each project contains one or more business processes and may include other resources used by the business processes or application. These include reusable resources that enable you to connect your application to other applications and systems or define the user interface of your application.

Business Process Composer enables you to create, test, and deploy a fully-functioning BPM application. However, there are some advanced features that can only be performed using Oracle BPM Studio. These include:

- Create project templates

- Configure advance human task parameters
- Create and view ADF forms
- Create external references
- Create exceptions

4.1.1.1 Editable Project Resources

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

- Processes

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

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

- Human Tasks

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

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

- Activity Guides

  An activity guide is a part of Guided Business Processes that enables you to define milestones for a project. Each project contains one activity guide where you can define multiple project milestones. Business Process Composer enables you to create and configure milestones.

  See Section 4.5, "Using Guided Business Processes to Create Project Milestones" for more information on creating and using project milestones.

- Oracle Business Rules

  Oracle Business Rules are statements that describe business policies or describe key business decisions. Using Business Process Composer you can create and edit business rules. Business rules are editable components of a project, but they also appear as part of the business catalog.

  See Chapter 13, "Using Oracle Business Rules" for information.

- Web forms

  Web forms define the interface that your application users see in Process Workspace. Business Process Composer enables you to create this interface from the ground up, or you can create a web form based on the existing data structure of a human task.

  See Chapter 9, "Working with Web Forms" for more information.

Note: Although web forms are a resource of a BPM project, they can only be created and edited in Business Process Composer. They cannot be viewed or edited in Oracle BPM Studio.

- Simulation definitions and models

  Use simulations to test the performance of your business processes. Simulations definitions and models define parameters used in a simulation and are stored as part of a BPM project.

  See Chapter 6, "Simulating Process Behavior" for more information.

4.1.1.2 The Business Catalog

You can use Business Process Composer to create and edit business catalog components. The business catalog is a container of reusable, shared implementation assets available to all the processes within a BPM Project.

After creating these elements, you can assign them to specific BPMN artifacts.

Business catalog components are accessible from the component palette. The reusable resources in the business catalog are:

- Human Tasks

  Human tasks define how end users interact with your BPMN processes. You can add human tasks to your business process by using the user task. You can assign a human task from the business catalog to each user task in your business process. See Section A.3, "Adding User Interaction to Your Process" for information on using human tasks within a BPMN process.

- Oracle Business Rules

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

  See Chapter 13, "Using Oracle Business Rules" for information on working with Oracle Business Rules using Business Process Composer. See Section A.5.2, "Introduction to the Business Rule Task" for more information on using the business rules task within a BPMN process.

- Business Objects

  Business objects define the data structures used within your application. You can use simple data objects to define complex data objects using Business Process Composer. See Section 12.4, "Introduction to Business Objects" for more information.

- Services

  Services define how a BPMN process connects other processes, systems, and services, including BPEL processes and databases.

  Using Business Process Composer, you can create new services based on web services. See Section 14.3, "Working with Services" for more information. You can also use services that were created as part of the project template.

  Within a BPMN process, services are implemented by assigning the service to a service task. See Section A.4.1, "Introduction to the Service Task" for more information.

- External references

  References define the interface of your BPMN processes. References implement message events, send tasks and receive tasks. See Section A.4, "Communicating With Other Processes and Services" for more information.

4.1.1.3 Business Catalog Components that Can Be Edited or Created

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

Table 4–1 Business Catalog Components in Business Process Composer

Business Catalog Component	Can be created?	Can be edited?
Business rules	Yes	Yes
Human tasks	Yes	Yes
Business Objects	Yes	Yes
Services	Yes	Yes
External References	No	No
Errors	No	No

4.1.2 Introduction to the Oracle BPM Repository

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

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

- Public: contains Oracle BPM projects.
- Templates: contains Oracle BPM project templates.

Your system administrator installs and configures the Oracle BPM repository when installing Business Process Composer.

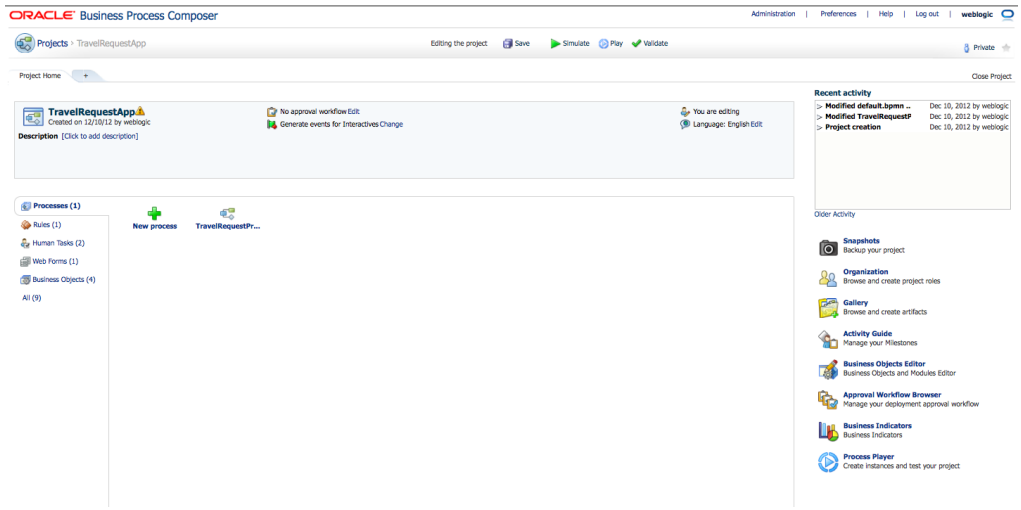
See Section 2.2, "Introduction to Business Process Composer Use Cases" for more information about sharing projects and project templates between Oracle BPM Studio and Business Process Composer.

See Section 4.4, "Creating and Working with Projects" for more information about opening projects in the Oracle BPM repository. See Section 8.2, "Introduction to BPM Project Templates" for information about creating and working with projects based on project templates.

4.2 Introduction to the Project Welcome Page

The Project Welcome Page provides access to information about a BPM project and access to common project-related features. Figure 4–1 shows the Project Welcome Page for the Sales Quote example project.

Figure 4–1 The Project Welcome Page



4.2.1 The Project Information Panel

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

Figure 4–2 The Project Information Panel

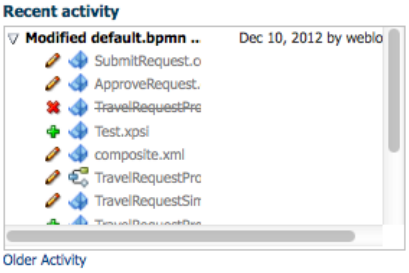
This pane displays the following information:

- **Title**: displays the title, creation date, and name of the user who created the project.
- **Description**: provides an optional description of the project.
- **Approval workflow:** specifies if an approval workflow is defined for the project.
- **Generate events**: defines how events are generated for this project.
- **Edit mode:** specifies whether the project is being edited or viewed.
- **Project language:** specifies the current language of the project.

4.2.2 The Recent Activity Panel

The recent activity browser provides a history of the major changes made to the current BPM project. Figure 4–3 shows an example of the recent activity browser.

Figure 4–3 The Recent Activity Panel



See Section 4.4.11, "How to View the History of Changes Made to a Project" for information about viewing recent project activity.

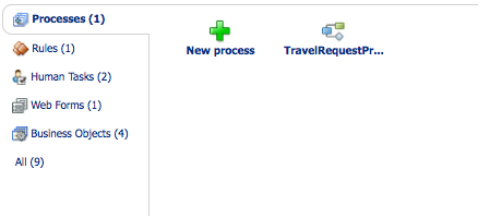
4.2.3 Introduction to the Project Component Pane

Use the project component pane to view and create the following project components:

- Processes
- Rules
- Human tasks
- Web forms
- Business Objects

Figure 4–4 shows how the project component pane appears in a project.

Figure 4–4 The Project Component Pane



4.2.4 Introduction to the Quickstart Menu

Use the Quickstart menu to quickly access the following common functionality within Business Process Composer:

- **Snapshots**: opens the snapshot editor. Snapshots provide a record of the changes made to a project during the development life-cycle. See Section 8.4, "Working with Project Snapshots" for more information.
- **Organization**: opens the organization editor which enables you to create and browse the roles defined for a project. See Section 4.6, "Defining the Roles Used in a Project" for more information.
- **Gallery**: opens the gallery view which enables you to create and browse processes and human tasks. You can also browse business rules.
- **Activity guide**: launches the activity guide editor that enables you to manage milestones within your project.

- **Business Objects Editor**: launches the business objects editor. See Section 12.4, "Introduction to Business Objects" for more information.
- **Approval workflow browser:** enables you to manage the deployment of BPM projects.
- **Business Indicators**: launches the business indicators editor.
- **Process Player**: launches the process player. This option only appears if process player has been enabled. See Section 7.1.2, "Enabling Process Player in Business Process Composer" for more information.

4.2.5 Introduction to the Oracle Business Process Composer Editors

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

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

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

4.2.5.1 Process Editor

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

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

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

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

4.2.5.2 Activity Guide Editor

Use the Activity Guide editor view, create, and edit milestones within an activity guide. You can access the Activity Guide editor by clicking the **Activity Guide** link in the Quickstart menu. See Section 4.5, "Using Guided Business Processes to Create Project Milestones".

4.2.5.3 Human Task Editor

Use the Human Task editor to create and edit human tasks included as part of the business catalog. See Chapter 11, "Working with Human Tasks" for more information.

4.2.5.4 Business Rules Editor

Use the business rules editor to create, view and edit Oracle Business Rules. To access the business rules editor, open a business rule from the project navigator. See Section 13, "Using Oracle Business Rules" for more information.

4.2.5.5 Data Associations Editor

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

See Section 12.3, "Working with Data Objects" for information about using data associations.

4.2.5.6 Expression Editor

Use the expression editor enables to define the expressions used within data associations and conditional sequence flows. See Section 12.7, "Introduction to Expressions" for information on using expressions and accessing the expression editor.

4.2.6 Introduction to the Supporting Browsers and Editors

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

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

4.2.6.1 Project and Process Validation Browser

The project and process browsers display any validation errors for each individual process or the whole project. See Section 4.4.9, "How to Validate a Project" for more information about validating projects.

4.2.6.2 Documentation Editor

Use the documentation editor to create and edit documentation for your processes. See Section 5.7, "Documenting Your Process" for more information.

4.2.6.3 Approval Workflow Browser

The approval workflow browser displays the status of a project within the approval workflow. Once all users defined as approvers have responded, users with the correct permissions can to Oracle BPM runtime. See Chapter 15, "Deploying a BPM Project" for more information.

4.3 Introduction to Project Sharing and Collaboration

Oracle BPM provides features for sharing BPM projects among Business Process Composer users. All BPM projects are stored in the BPM repository. You can control who has access to view or edit projects.

4.3.1 Private and Public Projects

BPM projects are defined as either private or public. Only the project owner can view or edit private projects. The project owner and other users who have the correct permissions can edit and view public projects.

See Section 4.4.5, "How to Share a Project with Other Users" for information about how to share projects.

4.3.2 Edit Mode

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

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

  See Section 4.4.6, "How to Edit a Shared Project" for information about how to set the edit mode.

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

4.3.3 Project Roles

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

- Owner: When a user creates a project, they are defined as the owner of the project. You can also define another user as the owner of a project. The owner of a project can perform the following:
  - Deploy a project
  - Create a project snapshot
  - Share a project with other users
  - Delete a project
- Editor: An editor can make changes to a project. When a user with the editor role opens a project, by default it is in read mode. If no other users are editing the project, the user can begin editing it. See Section 4.4.6, "How to Edit a Shared Project" for more information.
- Viewer: A viewer can view a project, but cannot make any changes to it.

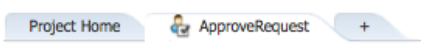
4.4 Creating and Working with Projects

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

4.4.1 How to Access the Project Welcome Page

The Project Welcome Page is displayed by default when you open a project from the Application Welcome page. When you are editing a component within a project, you can return to the Project Welcome Page by clicking the **Project Home** tab as shown in Figure 4–5.

Figure 4–5 The Project Home Tab



4.4.2 How to Create a New Project

From the Application Welcome page, you can create a new Oracle BPM project. Before creating a new project, you must decide whether to create it based on an existing project template or to create a new project.

New projects are not based on project templates. These projects contain only basic business processes created by process analysts.

To create a new project using the project navigator:

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

1. Start Oracle Business Process Composer.
2. From the project navigator, click New Project as shown in Figure 4–6.

Figure 4–6 The New Project Dialog

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

To create a new project using the main menu:

When you create a project from the main menu, you can configure project properties and select a project template.

1. Start Oracle Business Process Composer.
2. From the main menu, select **New** then **Project**.
3. Enter a name for the project.
4. Enter the following optional information:
   - **Description**: Provides a description of the project.
   - **Folder**: Specifies a folder in the BPM repository where the project is stored.
   - **Use template**: Creates the new project based on a project template. Click Choose to select the project template.
5. Select an optional deployment option from the drop-down list.
6. Click **Finish** to create the new project.

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

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

4.4.3 How to Open a Project Using the Application Welcome Page

To open a project directly from the application welcome page, click on the name of the project.

4.4.4 How to Open a Project Using the Main Menu

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

To open a project:

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

4.4.5 How to Share a Project with Other Users

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

To share a project publicly with all users:

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

To share a project with specific users or groups:

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

      All:

      Users:

      Groups:
   3. Click **Search**.
   4. Select an item from the **Available** column, then click **Move**.
   5. Click **OK**.
4. Select a role from the drop-down list.
   - Owner
   - Editor
   - Viewer
5. Click **Share**.
6. Click **Close**.

4.4.6 How to Edit a Shared Project

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

To begin editing a project:

To begin editing, click **Edit** at the top of the Project Welcome Page.

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

4.4.7 How to Save Changes to a Project

You can save changes to your project as you are editing processes and other project components. Changes are saved directly to the BPM repository and the changes are recorded in the **Recent Activity** pane in the **Project Welcome** page. See Section 4.4.11, "How to View the History of Changes Made to a Project" for more information.

To save changes to a private project:

To save changes to a private project, click the **Save** button in the project toolbar. The project is saved in the BPM repository. You can continue make changes to the project as necessary.

To save changes to a shared project:

- If you want to save your changes and continue editing the project, click **Save** in the process editor toolbar.

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

- To release the lock on the project, click **Save and Release** in the process editor toolbar.

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

4.4.8 How to Discard Changes to a Project

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

To discard changes made to the current project:

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

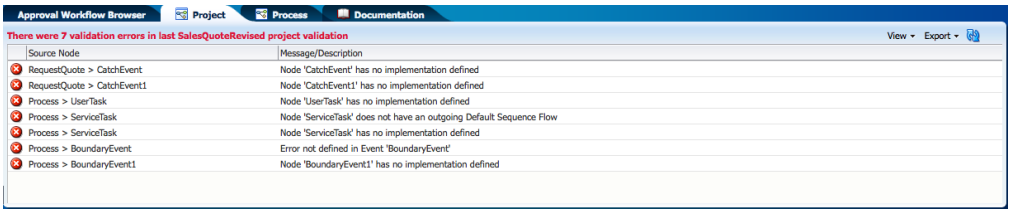
> **Note:** After discarding your changes, they cannot be recovered.

4.4.9 How to Validate a Project

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

Figure 4–7 shows an example of the types of errors displayed in the project error browser.

Figure 4–7 The Project Error Browser



To validate a project:

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

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

> **Note:** You cannot deploy projects that contain errors.

4.4.10 How to Close a Project

To close a project, click **Close Project** in the upper right hand corner of the project home page. Alternately, you can select

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

To view the history of major changes made to a project, use the recent activity browser. This browser displays these changes, including the following:

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

To view the history of a project:

1. Navigate to the **Project Welcome** Page.

   Project changes are displayed in the **Recent Activity** pane.
2. To view the details of a specific change, click the expand icon next to it.

4.4.12 How to View and Edit Project Properties

To view and edit project properties, use the project information pane of the Project Welcome Page. The project information pane is shown in Figure 4–2.

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

- **Description**: Adds an optional description of your project. This is useful when sharing your project with other users.
- **Approval workflow**: Defines the approval workflow for the project. See Section 15.1, "Configuring Approval Workflow for a Project" for more information.

- **Event generation:** Configures how sampling points are generated for the project as a whole. Use sampling points to generate information about the performance of an flow object within in a running process. The data generated according to this configuration is stored in the Process Analytics Database.

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

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

- Edit mode: Displays the edit mode for the project. See Section 4.4.6, "How to Edit a Shared Project" for information on changing the edit mode of a project.
- Sharing: Displays the sharing configuration of the project.
- Project Language: Displays the project language.

4.4.13 How to Mark a Project as a Favorite

You can mark important or frequently use project as favorites. In the project browser, you can choose to view only projects flagged as favorites.

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

4.5 Using Guided Business Processes to Create Project Milestones

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

4.5.1 Introduction to Guided Business Processes

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

Process designers can use Guided Business Processes to direct process participants to complete a business process through a set of guided steps associated with the process. By following the steps outlined in a Guided Business Process, process participants require less training to complete a business process, and the results of the process are more predictable.

4.5.1.1 Introduction to Activity Guides and Milestones

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

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

4.5.2 How to Configure the Activity Guide and Create Project Milestones

Use Business Process Composer to configure Guided Business Processes and add milestones to them.

To configure the activity guide for a project:

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

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

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

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

To create a new milestone:

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

To add a user task to a milestone:

1. Open the process where you want to add a milestone.
2. Right-click the user task you want to add to a milestone.

3. Select a milestone from the list, then click **OK**.

4.5.3 How to Generate a Process Report for Your Project

You can generate an HTML report that lists each of the processes in your project and shows detailed information about each process. This information includes:

- Process Properties
- Process Image
- Lanes
- Process Steps

To generate a process report:

1. Open your project.
2. From the Application Main menu, select **Process Report**.
3. Select the output, then click **OK**.

4.6 Defining the Roles Used in a Project

This section describes how to create project roles.

4.6.1 Introduction to Project Roles

Use project roles to model the users or groups that perform the work your business process represents. Roles define functional categories that correspond to job functions or responsibilities within your organization. Use Business Process Composer to create and edit the required roles within your process and assign them to swimlanes.

Use Business Process Composer to define the roles used by your BPM project. To perform more advanced mapping and configuration of project roles, including assigning users to the roles, use BPM Studio or Oracle Business Process Management Workspace. When a project is deployed to runtime, project roles can be mapped to the real-world users and groups of your organization.

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

4.6.2 How to Create Project Roles

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

To create a project role:

1. Access the Project Welcome Page.
2. Expand **Organization**, then click the **Add** icon.
3. Provide a name for the new role, then click **Add Role**.

To delete a project role:

1. Access the Project Welcome Page.
2. Expand **Organization**.
3. Select the role you want to delete, then click the **Delete** icon.

5

Working with Processes and the Process Editor

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

See Appendix A, "BPMN Flow Object Reference" for information on using Business Process Management Notation (BPMN) to design a business process.

This chapter includes the following sections:

- Section 5.1, "Introduction to Business Processes"
- Section 5.2, "Introduction to the Process Editor"
- Section 5.3, "Working with Business Processes"
- Section 5.4, "Working with Flow Elements"
- Section 5.5, "Working with Business Catalog Components"
- Section 5.6, "Working with Draft Processes"
- Section 5.7, "Documenting Your Process"
- Section 5.8, "Importing and Exporting Process Models"

5.1 Introduction to Business Processes

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

Although projects are higher level wrappers that contain all the resources of a business application, the processes within the project determine how the application works. Within a business process, BPMN flow objects define the flow and behavior.

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

Table 5–1 describes how the type of a process determines how it behaves in relation to other business processes.

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

Table 5–1 Types of Business Processes

Process Type	Description
Synchronous Service	Synchronous services are a type of business process that is invoked from another process or service synchronously. In a synchronous service, the calling process waits until the process completes before continuing.
Asynchronous Service	Asynchronous services are a type of business process that is invoked from another process or service asynchronously. In an asynchronous service, the calling process does not wait until the process completes before continuing.
Manual Process	Manual processes are those require user interaction. Manual processes must begin with a none start event. They must end with a none end event.
Reusable Process (Reusable Subprocess)	Reusable processes are processes that can be called by a BPMN process. In BPMN terminology, reusable processes are often called *reusable subprocesses*. See Section A.9, "Using Subprocesses in Oracle BPM" for more information about the different types of subprocesses supported by Oracle BPM. Use the call activity to call reusable subprocesses within your business process. See Section A.4.3, "Introduction to the Call Activity" for information about calling reusable processes.

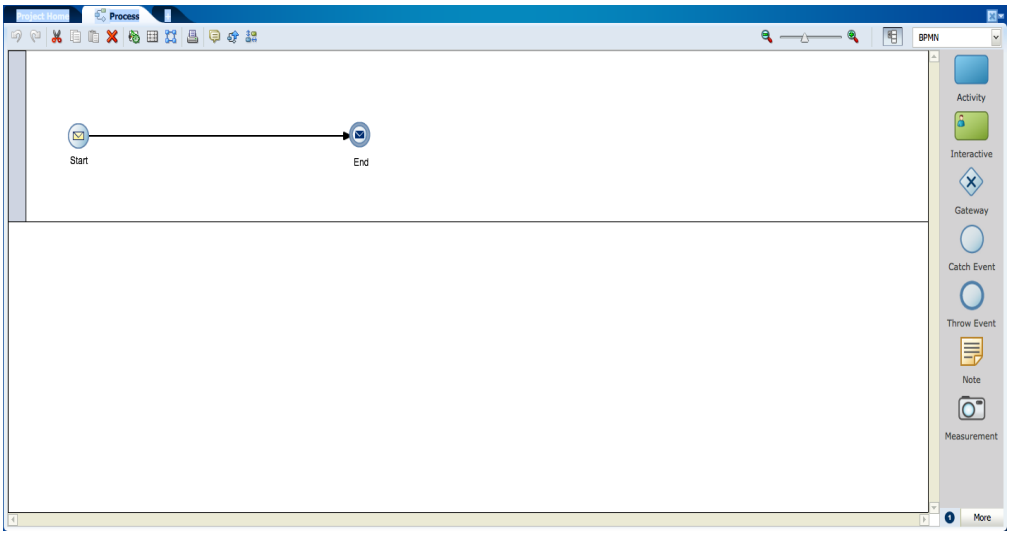
5.2 Introduction to the Process Editor

The process editor enables you to quickly create and edit BPMN processes. The process editor appears as a tabbed pane in the Business Process Composer application and is divided into three areas:

- Process Editor Toolbar: provides quick access to menu items related to process design. See Section 5.2.1, "Introduction to the Process Editor Toolbar" for more information.
- Process Editor Canvas: provides a work area for creating your process flow. Section 5.2.2, "Introduction to the Process Editor Canvas" for more information.
- BPMN Component Palette: enables you to quickly access the flow objects, sequence flows, and other elements that comprise a BPMN process. You can drag and drop these elements from the BPMN component palette onto the process editor canvas. See Section 5.2.3, "Introduction to the BPMN Component Palette" for more information.
- Business Catalog Palette: provides a list of the business catalog elements that you can use within your BPM project. See Section 5.2.4, "Introduction to the Business Catalog" for more information.

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

Figure 5–1 The Process Editor Window



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

5.2.1 Introduction to the Process Editor Toolbar

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

Table 5–2 Process Editor Menu

Menu Item	Description
Undo	Reverts the last change made to your process.
Redo	Reverses the last undo action you performed.
Cut	Cuts the selected items and copies them the clipboard.
Copy	Copies the selected items to the clipboard.
Paste	Pastes the items currently in the clipboard.
Delete	Deletes the selected elements from the process.
Autolayout	Automatically adjusts the layout of your process.
Toggle grid visibility	Shows or hides a grid in the process editor window.
Snap to grid	Centers the flow objects in your process on the nearest grid axis. Existing flow objects are automatically centered. New flow objects are automatically be centered when added. This menu item is active only when toggle grid visibility is enabled.
Print	Prints the process using your browser's printer setup.
Edit conversations	Opens the conversations editor. Use this editor to define the interface that determines the input and output data objects.
Find process usage	Determines which other processes within the current project call the current process.
View collaboration	Switches the process editor to collaboration view.
Data objects	Opens the data objects editor. See Section 12.3, "Working with Data Objects" for more information.

Table 5–2 (Cont.) Process Editor Menu

Menu Item	Description
Business indicators	Opens the business indicator editor.
Player	Launches the process player. See Chapter 7, "Using Process Player" for more information.
Zoom slider	Zooms in and out of your process.

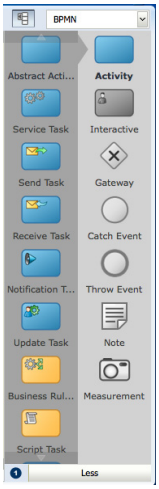
5.2.2 Introduction to the Process Editor Canvas

The process editor canvas is the central area of the process editor window. Use the process editor canvas to create the graphical representation of your process using the elements available in the BPMN component palette. In addition to a process flow, the process editor canvas also displays swimlanes.

5.2.3 Introduction to the BPMN Component Palette

Use the BPMN component palette to add flow objects, sequence flows and business catalog elements to your process. Figure 5–2 shows the component palette.

Figure 5–2 The Component Palette



Use the component palette to drag and drop artifacts to the process editor window.

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

The component palette separates BPMN elements into the following groups:

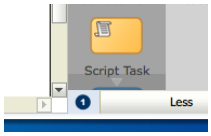
- Activity
- Interactive
- Gateway
- Catch Event
- Throw Event
- Note

- Measurement

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

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

Figure 5–3 Single Object Entry Mode



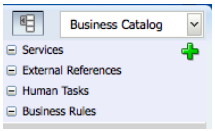
- Multiple object mode: enables you to quickly add multiple flow objects of the same type. This mode is indicated by an "N" within a blue circle.

Click the blue circle, shown in Figure 5–3, to toggle between the two entry modes.

5.2.4 Introduction to the Business Catalog

Use this panel to select reusable services from the business catalog. Figure 5–4 shows the business catalog.

Figure 5–4 The Business Catalog



These services are grouped as follows:

- Services
- External references
- Human tasks
- Business rules

> **Note:** You cannot create external references using Business Process Composer. Create external references using Oracle BPM Studio.
>
> Share projects containing references using the BPM repository or include them as part of a project template.
>
> You can use Business Process Composer to create services based on Web Services. However, you must create other services based on adapters and other SOA components in Oracle BPM Studio. You can include them within a project template or shared project.

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

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

5.3 Working with Business Processes

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

5.3.1 How to Create a New Business Process

You create business processes within an Oracle BPM project. You can add one or more processes to your project.

To create a new business process

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

The new process appears in the list of processes.

When you create a new business process it contains a start and end event connected by a default sequence flow. By default, both the start and end events are none events. You can change the type as required by your business process.

See Appendix A.2, "Defining the Start and End Point of a Process" for more information.

5.3.2 How to Open a Business Process

After opening an Oracle BPM project, you can open any of the processes it contains.

To open a business process

1. Go to the Project Welcome Page.
2. Click **Processes**.
3. Click the name of the process you want to open.

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

5.3.3 How to Delete a Business Process

You can delete processes from your project.

To delete a business process for a project:

1. Open your project.
2. Go to the Project Welcome Page.
3. Click **Processes**.
4. Move the cursor over the name of the process you want to delete.
5. Click the delete icon, the click **OK**.

5.3.3.1 What You Need to Know About Deleting a Business Process

When deleting a process, ensure that there are no remaining references to the deleted process elsewhere in your project. For example, if the deleted process was invoked from another process through a message throw event, you must ensure that you have

reconfigured the invoking process so it is no longer referring to the deleted process. An error is displayed during validation if any remaining references to the deleted process still exist.

5.4 Working with Flow Elements

This section describes the basic mechanics of using the process editor to add flow elements to a process. See Appendix A, "BPMN Flow Object Reference" for information about designing your business process using BPMN 2.0.

5.4.1 How to Add a Flow Object from the Component Palette

To add flow objects to your process drag them from the component palette onto the process editor canvas.

To add a flow element from the component palette:

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

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

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

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

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

5.4.2 How to Cut, Copy, or Delete a Flow Object

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

To cut, copy, or delete a flow object:

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

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

5.4.3 How to Paste a Flow Object in a Process

You can paste the flow object that you previously cut or copied.

To paste a flow object in a process:

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

5.4.4 How to Add a Sequence Flow to a Process

Sequence flows define the order or sequence which the work is performed within a process. For more information, see Section A.6, "Controlling Process Flow Using Sequence Flows"

To add sequence flows to your process:

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

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

5.4.5 How to Delete a Sequence Flow

You can delete a sequence flow from a BPMN process.

To delete a sequence flow from a process:

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

5.4.5.1 What You Need to Know About Deleting a Sequence Flow

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

5.4.6 How to Edit the Properties of a Flow Object

You can edit the basic properties of a sequence flow.

To edit the properties of a flow object:

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

   Your changes are automatically saved and the dialog window closes.

5.4.7 How to Assign a Custom Icon to a Flow Object

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

To assign a custom icon to a flow object:

1. Right-click the flow object, the select **Properties**.
2. Click **Change**, then select the icon you want to use.
3. Click outside the properties window to apply your changes.

5.5 Working with Business Catalog Components

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

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

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

You can assign business catalog components to the these flow objects:

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

To assign a business catalog component to a flow object:

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

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

5.5.2 How to Create New Human Tasks in the Business Catalog

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

After creating human tasks, you can then assign them to the user tasks within your process. See Appendix A.3, "Adding User Interaction to Your Process" for information about using human tasks within a BPMN process.

5.6 Working with Draft Processes

Oracle BPM enables you to create and deploy draft processes.

5.6.1 Introduction to Draft Processes

A draft process has one or more flow objects which do not have their implementation defined. By deploying a draft process, you can test the parts of your process that have been completed without having to wait until all flow objects have been implemented. To create a draft process, mark one or more flow objects within the process as draft.

When a flow object is marked as draft, you cannot configure data associations for it. If you mark a flow object as draft that has previously had data associations configured, these are 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 does not generate errors during project validation.

When a project containing a draft flow object is deployed, implementation details for those flow objects are ignored. For example, if your process contains a user task marked as draft, the runtime engine does not create instances of the associated human task.

5.6.2 How to Mark a Flow Object as Draft

To mark a flow objects as draft, edit the implementation properties.

To mark a flow object as draft:

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

5.7 Documenting Your Process

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

Using Oracle BPM you can create two different types of documentation:

- **End User**: Documentation visible to end users of your process-based application using Process Workspace.
- **Internal (Use Case)**: Documentation for process analysts and developers who are collaborating on a business process or who may revise it later.

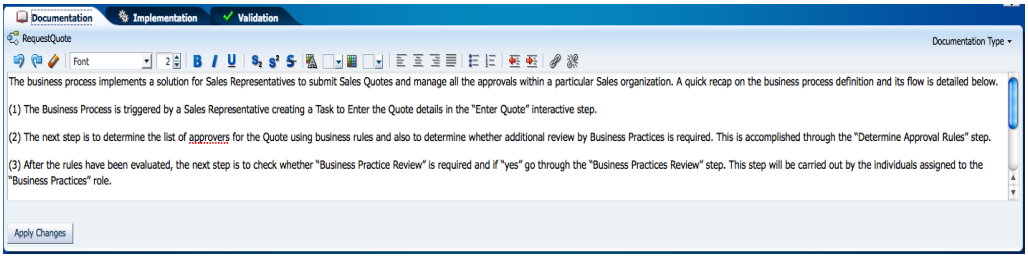
  You can define use-case documentation for each of the activities, events, and gateways within your process.

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

5.7.1 Introduction to the Documentation Editor

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

Figure 5–5 The Documentation Editor



5.7.1.1 Inserting Links in Your Documentation

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

5.7.2 How to Add Documentation to Your Process

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

To add documentation to a process:

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

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

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

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

6. Click **Apply** to save your changes.

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

5.7.3 How to Add Notes to a Process

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

To add a note to a process:

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

5.8 Importing and Exporting Process Models

Use Business Process Composer to import and export process models created in other programs.

5.8.1 Importing Process Models into Oracle BPM

Use Business Process Composer to import process models and convert them to BPMN notation. Oracle BPM supports importing and converting process models in the following formats:

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

You may need to modify Visio and XPDL processes before conversion to ensure that they are converted accurately. See Appendix E, "Preparing Processes for Import into BPMN" for more information.

> **Note:** If the original file contains properties and artifacts that are not supported by BPMN, the unsupported elements are not converted and are omitted from the final BPMN process.
>
> For example, if the original file contains loop characteristics on a regular activity, which is not supported in BPMN, the BPMN process does not contain the loop characteristics after conversion.

To import a process model:

1. From the main menu, select **Import**, then **Import Model**.
2. On your local file system, browse to the file you want to import, then click **OK**.

3. If you are importing a Visio or XPDL file, select one of the following:
   - Create a separate model from each pool
   - Merge pools into a single model

   This dialog appears even if the original file does not contain multiple pools.
4. Click **OK**.

You can view the newly created BPMN processes from the Project Welcome Page.

5.8.2 Exporting BPMN Processes to Oracle Tutor

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

To export a BPMN process to Oracle Tutor:

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

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

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

6 Simulating Process Behavior

This provide information on using Oracle Business Process Composer to run simulations to improve the performance of your business processes. It describes how to create simulation models and simulation definitions. It also describes how to run a simulation and analyze the results.

This chapter contains the following sections:

- Section 6.1, "Introduction to Simulations"
- Section 6.2, "Creating and Running a Simulation"
- Section 6.3, "Working with Simulation Definitions"
- Section 6.4, "Working with Simulation Models"
- Section 6.5, "Running Simulations"
- Section 6.6, "Analyzing the Results of a Simulation"

6.1 Introduction to Simulations

Oracle BPM provides functionality for simulating the behavior and performance of BPMN processes. Using Business Process Composer, process designers can run simulations during the design phase.

After creating and configuring a simulation, you run it in Business Process Composer to determine the efficiency of process using the resources allocations you have defined. Using simulations, you can:

- Define multiple simulation 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

> **Note:** In Business Process Composer you can only run simulations based on test data you define using Business Process Composer. You can run simulations on real-world data using Oracle BPM Studio.

6.1.1 Simulation Models and Simulation Definitions

Before running a simulation, you must define the simulated behavior of the project as and the processes you want to include in the simulation. To define a simulation you must create and configure the following in your BPM Project:

- **Simulation definition**

Simulation definitions define the processes and resources for a specific scenario. In a simulation definition you specify the processes included in the simulation by selecting the simulation models associated to those processes. A process can 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.

Within a BPM project you can define multiple simulation definitions, each with its own parameter definitions and simulation models. This enables you to compare multiple scenarios.

- **Simulation model**

  Simulation models define the simulated behavior of an individual process model. You can have multiple simulation models for each business process, enabling you to simulate different scenarios.

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.

6.1.2 Simulation Parameters

In addition to the general parameters defined for a simulation definition and a simulation model, you can also define simulation parameters for the start events and activities within a BPMN process.

6.1.2.1 General Simulation Definition Parameters

The following parameters define the general behavior of a simulation definition.

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

You must define these parameters when creating a simulation definition. However, you can redefine them later if necessary.

6.1.2.2 Simulation Model Parameters

The following parameters define the general behavior of a simulation model:

- **Model name:** defines the name of the simulation model.
- **Specify number of process instances to be created:** specifies the number of simulated instances are created during simulation.
- **Interactive tasks:** defines the distribution type used for interactive tasks. The available distribution types are:
  - Constant
  - Uniform
  - Exponential

  - Normal

  These are identical to the distribution methods defined for specific activities within a process. When you change distribution type or other parameters for a specific activity within a process, these values override the general values defined in the simulation model. See Section 6.1.2.5, "Activity Parameters" for more information.

- **Automatic tasks**: defines the number of simulated threads available when performing an automatic task. This parameter is identical to the Threads parameter you can define for individual activities within your process. See Section 6.1.2.5, "Activity Parameters" for more information.

6.1.2.3 Resource Parameters

Resources define the simulated resources within your organization. Resources are defined in a simulation definition and can be shared between each of the process models included. These resources can be associated with a specific role within your project. You can use these parameters together to create a resource profile that determines the expense, time, and efficiency of a person or group.

The following resource parameters are supported:

- **Name**: defines the name of the resource.
- **Cost per hour**: specifies the cost of the resource per hour when performing an activity.
- **Efficiency**: specifies how efficient the resource is when performing an activity. This parameter is used when selecting the Maximum efficiency policy when defining how organizational resources are allocated. See Section 6.1.2.5, "Activity Parameters" for more information.
- **Capacity**: specifies how many activities can be performed at one time.
- **Availability**: specifies the percentage of time this resource is available.
- **Roles**: specifies the roles associated to this resource.

6.1.2.4 Start Event Parameters

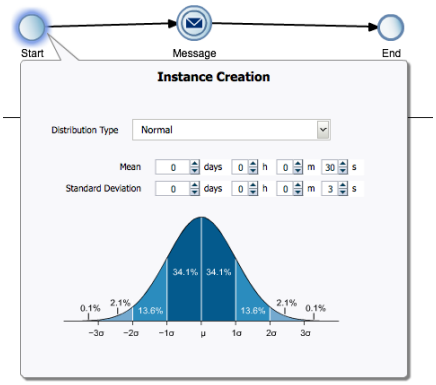
To simulate the behavior and performance of start events, Oracle BPM uses a statistical model that simulates the probability of a certain behavior. You must select the statistical distribution that Oracle BPM uses.

The following distribution types are available:

- **Constant**: triggers the start event at a specific interval. For example, if you specify a period of 1 minute and 15 seconds, the event is triggered every 1 minute and 15 seconds. Each time the event is triggered, a new instance of the process is created.
- **Uniform**: repeats the start event at intervals within a specific margin specified by the mean and delta parameters. For example, if you define the mean as 30 seconds and the delta as 3 seconds, the start even creates a new instance in intervals between 27 and 33 seconds, with an equal probability.
- **Exponential**: triggers the start event a specified number of times within a certain interval. For example, if you define the frequency as 10 and the interval as 1 hour, the event is triggered 10 times per hour. The distribution of events is based on exponential distribution.
- **Normal**: triggers the start event base on normal, or Gaussian, distribution. This is a continuous probability distribution that repeats based on a bell curve. For

example, if you define a mean of 10 minutes and a standard deviation of 2 minutes, the event will repeat according to the probabilities shown in Figure 6–1.

Figure 6–1 Simulation - Normal Distribution in a Start Event



In this example, the event has a 62.8 percent probability of repeating in an interval between 8 and 12 minutes.

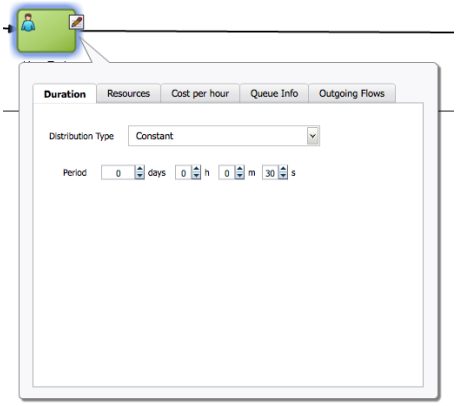
- **Real**: triggers the start event based on user-specified intervals: hourly, daily, weekly, or monthly. You can define the mean and standard deviation for each interval. Using these parameters, the start event creates new process instances using normal distribution.

  For example, you can specify a daily distribution with a mean of 5 minutes and a standard deviation of 1 minute for weekdays and a mean of 1 hour and a standard deviation of 10 minutes for weekends.

6.1.2.5 Activity Parameters

In addition to start events, you can define parameters that determine the simulated behavior or interactive and automatic activities within a process. You define these parameters the tabbed panes of the simulation editor. Figure 6–2 shows an example of the tabbed panes of an interactive activity.

Figure 6–2 Simulation Parameters for an Interactive Activity



The simulation parameters for activities are:

- **Duration**: defines the statistical model used to determine the amount of time required to perform an activity. The statistical models used are similar to those

defined for start events except, instead of defining how often an event is triggered, they define how long it takes to perform the work of a specific activity.

See Section 6.1.2.4, "Start Event Parameters" for a description of each distribution type.

- **Resources** (interactive activities only): defines how many interactive activities can be performed simultaneously. You can define how simulated resources are allocated to this activity.
  - **Organizational resources**: are resources shared between all the processes within a simulation definition. You must specify the policy that determines how the simulated participants are selected to perform the activity.
    * **Minimum cost**: selects less costly resources first.
    * **Maximum efficiency**: selects the most efficient resources first.
    * **Random**: randomly selects between Minimum cost and Maximum efficiency.

    See Section 6.1.2.3, "Resource Parameters" for information on defining organization resources within a simulation definition.
  - **Fixed resources**: explicitly defines the number of resources available to perform the interactive activity.
- **Threads** (available only for automatic activities): defines the number of simulated threads that are used to perform an automatic activity.
- **Cost per hour**: defines the cost required to perform the activity. Use this parameter to create cost-base reports.
  - **Activity Cost Type**: can be defined as a base cost or as a base cost plus the cost of resources assigned to perform the activity.
  - **Activity Fixed Base Cost**: defines the value (specified as a decimal number) of the fixed base cost.

  See Section 6.1.2.3, "Resource Parameters" for information on defining organization resources within a simulation definition.
- **Queue info**: specifies the maximum size of the queue for this activity. This is the number of instances that are currently waiting at this activity. When this number reaches the maximum size, the simulation issues a warning.
- **Outgoing flows**: specifies the probability (defined as a decimal number) of a process instance continuing along each of the outgoing sequence flow. If only one outgoing sequence flow is defined, the probability is specified as 1 and cannot be changed.

6.2 Creating and Running a Simulation

To run a simulation, you must first define a simulation model for your project and at least one simulation definition for each of the processes you want to include in your simulation. You can create multiple simulation definitions to test and compare the performance of your processes.

To create and run a simulation:

1. Create the BPMN processes you want to include in your simulation.

   See Section 5.3, "Working with Business Processes" for information about creating and working with business processes.

2. Create a simulation definition and simulation models.

   The initial simulation model for a process can only be created in the simulation definition editor. After you have created at least one simulation model for a process, you can create additional simulation models for a process.

   See Section 6.3.1, "How to Create a Simulation Definition" for information about running the simulation wizard to create simulation definitions and simulation models.

3. Configure parameters to define the simulated behavior of your processes.

   See Section 6.1.2, "Simulation Parameters" for information about the different parameters you can define for a simulation definition and simulation model. See Section 6.4.2, "How to Edit a Simulation Model"for information about how to configure parameters for the flow objects within your process.

4. Run the simulation.

   After creating and configuring a simulation definition and simulation model, you can run your simulation.

   See Section 6.5, "Running Simulations" for more information.

5. Analyze the results of the simulation.

   After running a simulation you can analyze the results of the simulation and make adjustments to the parameters to determine how to improve the performance of your process. See Section 6.6, "Analyzing the Results of a Simulation" for more information.

6.3 Working with Simulation Definitions

Simulation definitions define the simulated behavior of your BPM project as a whole. Within a simulation definition, you can define general parameters, including the organizational resources, and select which simulation models to include in a simulation. You can define multiple simulation definitions to test different combinations of parameters and processes.

After creating a simulation definition, you can edit its resources, add simulation models, etc. as described in the following sections.

6.3.1 How to Create a Simulation Definition

In a simulation definition, you can changes the values of different parameters to see how they influence the performance a project. The parameters you can define include:

- The start time and duration of the simulation
- Which process simulation models you want to include in the project simulation
- Participant resources you want to include in the simulation

Business Process Composer provides a wizard that walks you through the process of creating and configuring a new simulation definition, creating and configuring new simulation models, and configuring organizational resources.

To create and configure a simulation definition:

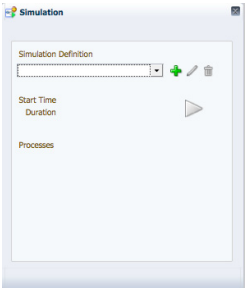
1. In the **Project Welcome** page toolbar, click **Simulate** as shown in Figure 6–3.

Figure 6–3 The Simulate Button in the Project Welcome Page

The Simulation editor appears.

2. Click the **Add (+)** icon as shown in Figure 6–4.

Figure 6–4 The Simulation Editor window



3. Enter a name for the simulations and values for the duration and start time of the simulation definition.

   You must define these parameters when creating a simulation definition. However, you can redefine them later if necessary.

   See Section 6.1.2.1, "General Simulation Definition Parameters" for more information about each of these parameters.

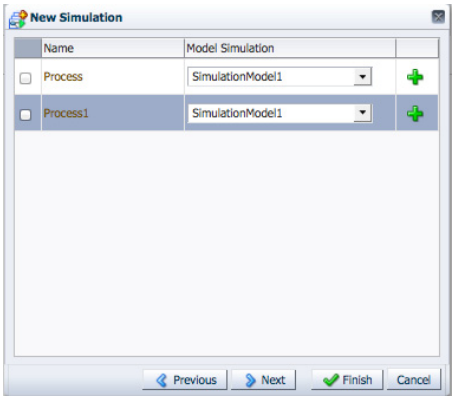
4. Click **Next**.

5. Create a new simulation model for each process you want to include in the simulation definition.

   Simulation models can be shared across simulation definitions. However, the first time you create a simulation definition, you must create at least one simulation model for each process if you have not created one previously.

   a. Click the **Add** (+) icon to create a new simulation model as shown in Figure 6–5.

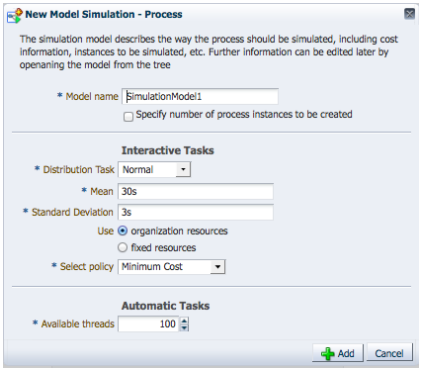
Figure 6–5 The Simulation Model Panel



A simulation model defines the simulated behavior of a process. You can define multiple simulation models for a process, however only one simulation model is used for each process within a simulation definition.

b. Provide information for each of the fields as shown in Figure 6–6.

Figure 6–6 The New Simulation Model Editor



See Section 6.1.2.2, "Simulation Model Parameters" for more information about these parameters.

c. Click **Add**.

Create additional simulation models as necessary. You should create at least one simulation model for each process in your project even if you do not include it in the simulation definition. This enables you to create additional simulation models later and add them to other simulation definition.

6. Select the check box next to each process whose simulation model you want to include in the simulation definition as shown in Figure 6–5.

   If you have not created a simulation model for a process, you cannot select the process.

7. Click **Next**.

8. Optionally, click the **Add resource** icon to add resources to the simulation definition.

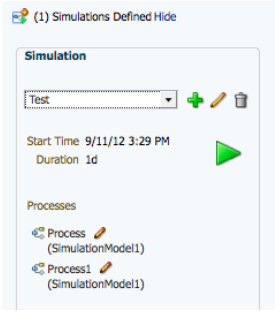
   See Section 6.1.2.3, "Resource Parameters" for information on the resource parameters you can define.

9. Click **Finish**.

6.3.2 What Happens When You Create a Simulation Definition

After creating a new simulation definition and simulation models the simulation panel appears as shown in Figure 6–7.

Figure 6–7 The Simulation Panel After Creating a Simulation Definition



This panel displays a drop down list containing all the simulation definitions defined in your project. It also displays all the simulation models included in the currently selected simulation definition. If you select a different simulation definition from the drop down list, the simulation panel displays the simulation models defined for that simulation definition.

From this panel you can perform the following tasks:

- Edit a simulation definition. See Section 6.3.3, "How to Edit a Simulation Definition" for more information.
- Run a simulation. See Section 6.5.1, "How to Run a Simulation" for more information.
- Edit a simulation model. See Section 6.4.2, "How to Edit a Simulation Model" for more information.

6.3.3 How to Edit a Simulation Definition

After creating a simulation definition, you can edit it from the simulation panel.

To edit a simulation definition:

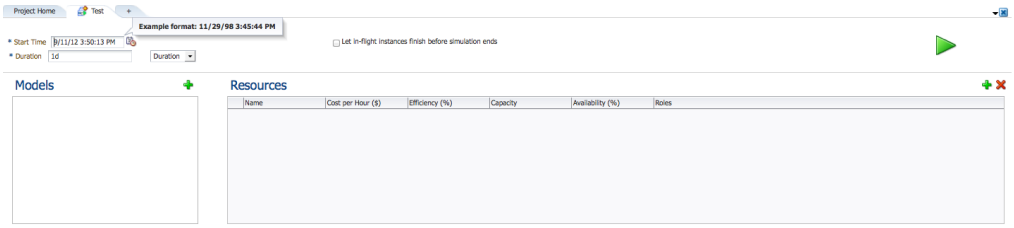
1. From the Project Welcome page toolbar, click **Simulate** as shown in Figure 6–3.

   After clicking **Simulate**, the simulation panel appear as shown in Figure 6–7.

2. From the drop down list, select the simulation definition you want to edit, then click the **Edit** icon.

   The simulation definition editor appear in a tabbed pane as shown in Figure 6–8.

Figure 6–8 The Simulation Definition Editor



From this editor, you can add or remove simulation models from a simulation definition, add resources, and run simulations.

3. Edit the Start Time, Duration, and End Time as necessary. See Section 6.1.2.1, "General Simulation Definition Parameters" for more information about these parameters.
4. To add resources to a simulation definition, click the **Add Resource** icon, then edit the fields of the **Resources** table as necessary. See Section 6.1.2.3, "Resource Parameters" for more information about the parameters you can define for a resource.

6.3.4 How to Add a Simulation Model to a Simulation Definition

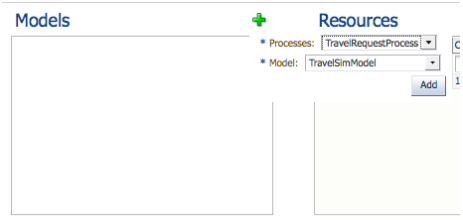
Before creating a new simulation model, you need to create at least one simulation definition. See Section 6.3.1, "How to Create a Simulation Definition" for more information.

To add a simulation model to a simulation definition:

1. From the **Project Welcome** page toolbar, click **Simulate** as shown in Figure 6–3.
2. From the drop down list, select the simulation definition where you want to add a simulation model.

   New simulation models are created from the simulation definition editor. However after creating a simulation model for a process, it can be shared with multiple simulation definitions.
3. Click the **Edit** icon.
4. Click the **Add Model Simulation** icon in the **Model** pane.
5. Select a process from the drop down list, then select a simulation model as shown in Figure 6–9.

Figure 6–9 Adding a simulation model to a simulation definition



6. Click **Add**.
7. The new simulation model appears in the list.

6.4 Working with 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.

6.4.1 How to Create a New Simulation Model

You can create multiple simulation models for a process. Different models can be included in different simulation definitions to determine performance based on different parameter settings.

Note: You cannot directly create a the first simulation model for a process. The first simulation model for a process must be created using the simulation definition wizard. After creating the initial simulation model, you can create additional models using the following procedures.

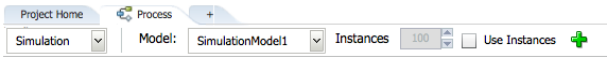
See Section 6.3.1, "How to Create a Simulation Definition" for information about creating a simulation model when creating a simulation definition.

To create a new simulation model for a process

1. In the **Project Welcome** page toolbar, click **Simulate** as shown in Figure 6–3.
2. Select a simulation definition that contains the process where you want to create a new simulation definition.
3. Click the edit icon next to the simulation model.

   The process opens in the simulation canvas. This canvas is similar to the process editor canvas. You can toggle back and forth between the process designer and simulation canvases using the drop down list in the process editor toolbar
4. In the process editor toolbar, click the **Add** icon as shown in Figure 6–10.

Figure 6–10 The simulation editor toolbar



5. Enter a name, then click **Create**.

The new simulation model is created. You can use this model when creating a new situation definition or editing an existing one. See Section 6.3.4, "How to Add a Simulation Model to a Simulation Definition" for more information.

6.4.2 How to Edit a Simulation Model

To edit a simulation model

1. From the **Project Welcome** page toolbar click **Simulate**.
2. From the drop down list, select the name of the simulation definition containing the simulation model you want to edit.
3. Click the **Edit** icon next to the simulation model you want to edit.

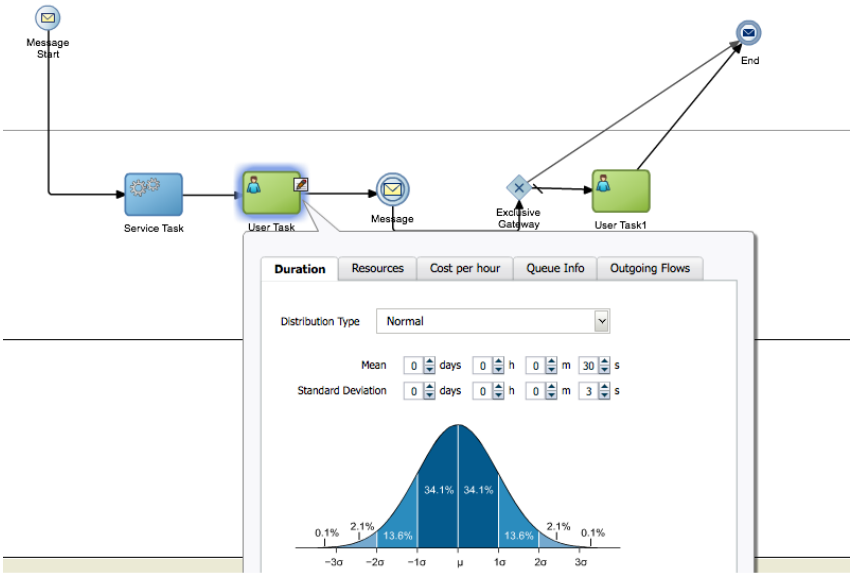
   The process displays in the simulation canvas. This canvas is similar to the process editor canvas. You can toggle back and forth between the process designer and simulation canvases using the drop down list in the process editor toolbar as shown in Figure 6–10.

Note: This drop down list only appears after you have defined at least one simulation model for a process.

4. Hover the mouse over the flow object whose simulation information you want to configure, then click the **Edit** icon of the flow object where you want to define resources.

   The simulation definition pane appears as shown in Figure 6–11.

Figure 6–11 The Simulation Model Editor Showing the Resource Definition Panel



5. Edit the simulation parameters for the flow object by clicking the tab and editing the appropriate parameters. See Section 6.1.2.4, "Start Event Parameters" and Section 6.1.2.5, "Activity Parameters" for information on the parameters you can configure for specific flow objects.

6. After editing the parameters, click outside the popup window to save your changes.

6.5 Running Simulations

After defining a simulation model and simulation definitions, you can run the simulation to view the performance of your BPM project.

6.5.1 How to Run a Simulation

To run a simulation, you must have created at least one simulation definitions and simulation models for each of the processes you want to test.

To run a simulation:

1. From the Project Welcome page toolbar, click **Simulate**.
2. Click the **Run** icon.

6.5.2 What Happens When You Run a Simulation

The animation of the simulation appears in the simulation definition editor.

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

6.6 Analyzing the Results of a Simulation

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

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

- Date and Time
- Process
- Instance
- Instance path

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

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

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

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

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.

7

Using Process Player

The chapter describes how to use process player to test the business processes within a BPM project.

This chapter contains the following sections:

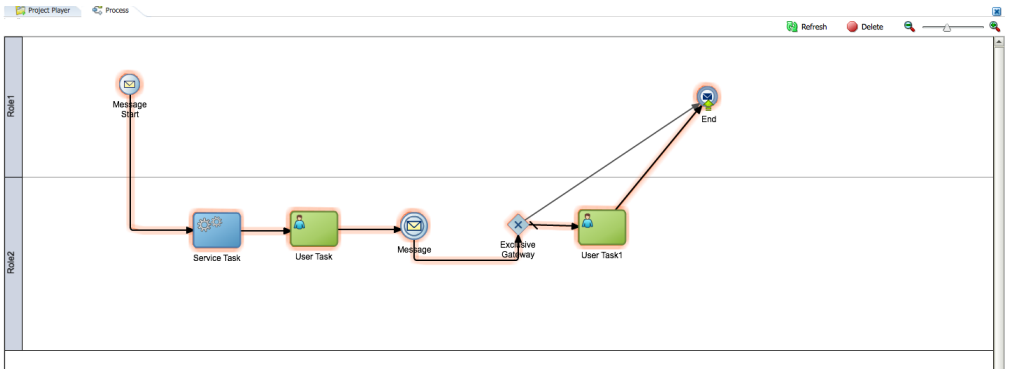
- Section 7.1, "Introduction to Process Player"
- Section 7.2, "Using Process Player to Test the Behavior of Business Processes"

7.1 Introduction to Process Player

Process player provides a quick and easy way of testing business processes. It provides a runtime environment, accessible from Business Process Composer, that emulates the real-world behavior of business processes. This enables process designers to easily create, test, and revise processes without having to save and deploy the BPM project and view it in Process Workspace.

Process player provides an animated view of the behavior of your process. Figure 7–1 shows the process player viewer.

Figure 7–1 The Process Player



The red outline shows the path the process instance takes through the flow objects and sequence flows of your process. The specific path an instance takes through your process depends on the input data you provide for various flow objects. See Section 7.1.1, "How Process Player Handles the Flow Objects of Your Process" for more information.

When you run process player on a business process, it validates the project and deploys the current draft version of the BPM project to a test partition of the Oracle

BPM runtime environment. When using process player, you do not have to save or manually deploy the project to see view changes while designing a process.

You can use process player to test the creation and behavior of process instances. You can create multiple instances of each process within your project

Note: You can only run process player if you are currently editing the BPM project.

7.1.1 How Process Player Handles the Flow Objects of Your Process

As process player runs through a process, it emulates the run time behavior of some of the flow objects in your process.

- User tasks

  When process player reaches a user task in a process, the action it performs depends on whether or not the human task has a form assigned.

  If no form is assigned, process player pauses to enable you to input the simulated user you want to perform the task. It prompts you to select one of the outcomes defined for the human task. Approve and Reject are provided as default outcomes. However, the list of possible outcomes depends on how outcomes are configured in the human task. See Section 11.3.4, "How to Configure Basic Task Properties" for more information. After selecting an outcome, process player continues to the next flow object of your process.

  If a web form is assigned to the human task, process player give you the option of launching the web form or selecting the outcome. If you choose to launch the web form, Business Process Composer deploys the web form and displays it in a viewer.

  If an ADF form is assigned to the human task, you must deploy it to the run time environment using Oracle JDeveloper to be able to view it using process player. If the ADF form is deployed, Business Process Composer is able to access it when deploying to the process player partition.

  After viewing the web form or ADF form, you must manually close the form viewer window to continue running your process.

- Message send events and send tasks

  When process player reaches a message send event or a send task within a process, it performs these automatically. It then continues to the instance of the process being called and pauses at the corresponded message catch event or receive task.

  In both cases, you must manually return to the parent process. For example, If the send/receive pair is creating an instance on a different process of the same project you can return to the process player home, select the new instance for this process, run the child process, then return to the parent process.

  If the send/receive pair calls an external web-service you must manually enter the required web service message to continue running the process.

- Timer events

  When process player reaches a timer event within a process, it pauses and waits until you click the **Run** icon of the flow object. Process player moves to the next flow object in the process.

- Call activities

When process player reaches a call activity calls the child process and creates a new instance of the process. Click the "drill-down" icon to view the child process.

- End events

  When process player reaches an end event, it pauses and displays the "drill-up" icon. Clicking this icon causes process player to return to the parent process. If the current process has no parent, process player returns to the process player home and deletes the process instance.

- Other flow objects

  When process player reaches another flow object that causes the instance to wait for some operation or external event, process player pauses. To continue running the process click the **Refresh** icon located at the top of the process player home.

7.1.2 Enabling Process Player in Business Process Composer

Before process designers can use process player to test the processes of a BPM project, a system administrator must enable process player. See Section 16.4, "How to Define Administrator Credentials for Process Player"for more information.

7.2 Using Process Player to Test the Behavior of Business Processes

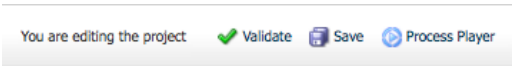
7.2.1 How to Access Process Player

There are two ways of accessing process player in Business Process Composer:

- From the project toolbar.

  When project player is enabled and you are editing the project, the project toolbar displays an icon for accessing process player as shown in Figure 7–2.

Figure 7–2 The Project Editor Toolbar Showing the Process Player Icon



- From the project main menu.

  When project player is enabled and you are editing the project, the project menu displays a menu option for accessing process player as shown in Figure 7–3.

Figure 7–3 The Project Menu Showing the Process Player Menu Item

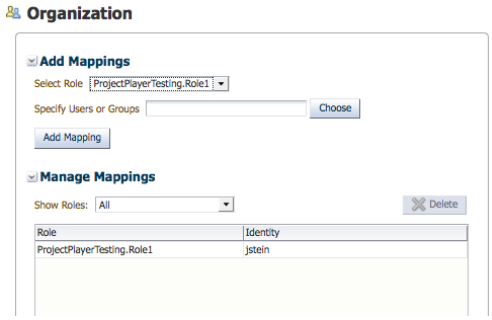
7.2.2 How to Map the Roles Defined in Your Process to Users in Your Organization

Before using process player, you must map the roles defined in your process to the users or groups of the organization infrastructure defined in your runtime environment. Process player uses the information of your organization to mimic the behavior of your business processes in real-world situations.

To map roles to users in your organization:

1. Open your project.
2. Start process player. See Section 7.2.1, "How to Access Process Player" for more information.
3. In the **Organization** pane, select the role of your process you want to map from the drop down list. The drop down list displays all the roles defined in your process. Figure 7–4 shows the **Organization** pane.

Figure 7–4 Process Player - Organization Pane



4. Select the user or group you want to map.
   a. Click **Choose**.
   b. Select user or group from the drop down list.
   c. Enter the name of the user or group you want to search for, then click **Search**. To see a list of all users or groups, leave the text area blank, then click **Search**.
   d. In the table, click the checkbox next to each user or group you want to map. After selecting a user or group, it appears at the bottom of the chooser window.
   e. Click **OK**.
5. Click **Add Mapping**.

The users or groups you mapped to the process role appear in the mappings table.

> **Note:** You must map at least one user or group for each role in your process. If process player encounters a user task with an unmapped role, it cannot continuing running the process past the human task.

7.2.3 How to Use Process Player to Run a Business Process

Before using process player to test the behavior of your business processes, ensure that you have mapped all the roles in your process to at least one user or group within your organization. See Section 7.2.2, "How to Map the Roles Defined in Your Process to Users in Your Organization" for more information.

To use process player:

1. Open your project.
2. Access process player. See Section 7.2.1, "How to Access Process Player" for more information.
3. Select a process.
4. Click the **Play** icon located on the start even of your process as shown in Figure 7–5.

Figure 7–5 The Process Player Play Icon on a Start Event

Process player begins running the process. As it passes through each flow object and sequence flow, it outlines the path it takes through the process in red.

As process player continues running through your process, it stops when the process instance reaches one of the following flow objects:

- User tasks
- Call activities
- Message events
- Timer events

You must provide input for each of theses flow objects before process player continuances running the process. See Section 7.1.1, "How Process Player Handles the Flow Objects of Your Process" for more information.

5. If process player pauses on a user task:
   a. Click the **Play** icon on the user task as shown in Figure 7–6.

Figure 7–6 The Process Player Play Icon on a User Task

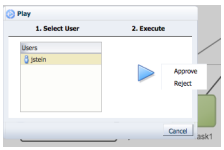
   b. Select the user you want to perform the task.

> **Note:** If the list of users appear empty, you must ensure that you have correctly mapped all of the roles of your process.

   c. Click **Run**, then select the outcome from the list. The possible outcomes are defined by the human task associated with the current user task.

   Figure 7–7 shows an example where the user jstein is selected to perform the approve outcome.

Figure 7–7 Selecting the User and Outcome for a Human Task



After you select the outcome, process player continues to the next flow object in your process.

6. If process player pauses on a message catch event: or a receive task, it creates an instance of the child process.
   a. Click the **Run** icon.
   b. Select the **Project Player** tab.
   c. In the **Instances** table, select the newly created instance.

      Business Process Composer asks if you want to close the process player tab for the original process. Closing this window has no effect on the process instances.
   d. Click **OK**.

      Process player opens the new process instance and begins running the process from the message start event called from the parent process.
   e. Click the **Run** icon for any flow objects that pause process player as outlined in previous steps.
   f. When process player reaches the message end even of the process, click the drill-up arrow to return to the parent process as shown in Figure 7–8.

Figure 7–8 The Drill-up Icon on a Message End Event

Process player closes the tab for this process and removes the process instance from the list of instances.

   g. From the list of process instances, open the process instance of the parent process.

After reopening the process instance of the parent process, process player continues running through the process from the point where the child process was called.

7. When process player reaches an end event in your process, click the drill-up icon, as shown in Figure 7–8, to finish the process instance. Process player returns to the process player editor and deletes the process instance.

8

Working with the Project Life Cycle

This chapter describes some of the advanced features related to BPM projects and how to work with them within the development life-cycle.

This chapter includes the following sections:

- Section 8.1, "Importing and Exporting Projects"
- Section 8.2, "Introduction to BPM Project Templates"
- Section 8.3, "Working with Project Templates"
- Section 8.4, "Working with Project Snapshots"

8.1 Importing and Exporting Projects

Business Process Composer enables you to import and export BPM projects as .exp files. This enables you to share projects directly with other Business Process Composer and BPM Studio users directly through the files system without having to use the BPM repository.

8.1.1 How to Import a Project from Your Local File System

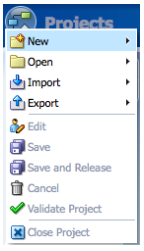
You can import a BPM project that was previously exported and saved as a .EXP file. The imported project is stored in the BPM repository.

To import a project:

1. Access the application welcome page.
2. From the main menu, select **Import**, then select **Import Project**

   The main menu is accessible from the upper left-hand corner of the Business Process Composer user interface as shown in Figure 8–1.

Figure 8–1 The Application Composer Main Menu



3. Click **Browse**, then select the project file you want to import.

4. Click **OK**.
5. Enter a name for the project.

   After you select a project file to import, the name field is automatically propagated.
6. Enter an optional description.
7. If you want to optionally select a folder within the BPM repository where the imported project will be created, click **Browse**, then select the folder.
8. Click **OK** to import the project into the BPM repository.

8.1.2 How to Export a Project to Your Local File System

Projects exported from Business Process Composer can be imported into Oracle BPM Studio. Exporting a project to your local file system enables you to share projects without using the Oracle BPM MDS repository.

For information on importing a project into Oracle BPM Studio see the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

To export a project as an EXP file:

1. Open the project you want to export.
2. From the main menu, select **Export**, then select **Export Project**.
3. Choose a location on your local file system and click **Save**.

Your exported project is saved as a.EXP file on your local file system.

To export a project as an Oracle Tutor file:

1. Open the project you want to export.
2. From the main menu, select Export, then select **Export Oracle Tutor**.
3. Select one of the following:
   - **Active process**: Selects the currently active process. This option is only displayed if a process has already been opened and focus is currently on that process.
   - **All open processes**: Exports only the processes that are currently open.
   - **All processes of project**: Exports all of the processes of the current project.
4. Click **OK**.

The file is saved as a ZIP file on your file system.

8.2 Introduction to BPM Project Templates

Project templates enable business users to quickly create custom Oracle BPM applications and deploy them to runtime without assistance from developers. Using Oracle Business Process Composer business users can create new BPM projects based on project templates. These projects contain BPMN process flows and can be deployed directly to runtime.

Project templates enable you to incorporate reusable components and services including Human Tasks, Business Rules, and Adapters. These services are stored as part of the business catalog.

Business Process Composer users who have administrative privileges can also import project templates directly from their local file system. See Chapter 16, "Performing Administrative Tasks" for more information.

Using Oracle BPM Studio process developers can convert a normal project to a template and publish it to the Oracle BPM repository. After a template is available in the repository, you can create new projects based on the template using Business Process Composer.

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

Project templates are based on normal Oracle BPM projects and generally have all the required implementation and services defined in the business catalog. However, they often do not have all of the required services assigned to the necessary flow objects. After creating a new project based on a project template, a business analyst can assign business catalog components to the necessary flow objects.

The specific services required for each activity are defined by the editor policies of the project template. After the process analyst incorporates the required services, the project can be deployed to Oracle BPM runtime.

8.2.1 Introduction to Edit Policies

Project templates also allow you to define edit policies for processes and flow objects within a process. Edit policies determine what parts of a process can be changed or edited when creating a new project based on a project template. Edit policies are defined for the entire process. However, you can also define edit policies for individual flow objects.

See the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for information on defining edit policies in a process template.

Edit policies allow the creator of a project template to define what elements of a process can and cannot be changed when a project is created from a template.

> **Note:** You cannot change the edit policy settings of processes and elements using Business Process Composer.

8.2.1.1 Process Level Edit Policies

Within a project template, each process contains an edit policy which determines the changes you can make to the process using Business Process Composer.

Table 8–1 describes the process level edit policies.

Table 8–1 Process Level Edit Policies

Edit Policy	Description
Activity Sealed	User cannot make any changes to the activities within the process.
Flow Sealed	The overall flow of the process cannot be changed. A user can edit specific implementation details, but cannot change the process flow

8.2.1.2 Component Level Edit Policies

Within a process, there are also edit policies that apply to the flow objects within a process. Component level edit policies can be configured for the flow objects within a process.

Table 8–2 describes the edit policies values that can be configured for component level edit policies.

Table 8–2 Component Level Edit Policies

Edit Policy	Description
Sealed	The flow object cannot be modified
Can modify implementation	The user may redefine this flow object if necessary.
Must implement	The user is required to assign a component from the business catalog to this flow object for it to function correctly.
Use process permission	Uses the default edit policy defined by the process.

8.2.2 About Using Data Objects and Variables in Project Templates

A project template can define the data objects used within a project. These can be the Oracle BPM default types or complex data objects created by process template developers within Oracle BPM Studio.

When editing a project based on a project template in Business Process Composer, you can add and create new data objects as necessary. However, you can only create new data objects based on types that are previously defined in the project template. You cannot create new types of complex data objects.

You can use any of the data objects defined in a project template in data associations and expressions. See Section 12, "Handling Data in Your Business Processes" for more information.

8.3 Working with Project Templates

Using Business Process Composer you can create new projects based on project templates.

To create a new project from a project template:

1. Launch Business Process Composer.
2. From the main menu select **New**, then select **New Project**.
3. Enter a name for you project
4. Select **Use Template**, then click **Choose**.
5. Select the template you want to use.
6. Optionally, choose the folder where you want to store the new project.
7. Click **Next**
8. From the drop-down list, choose the type of approval routing you want to configure for the project.

> **Note:** You can change the type of approval routing after you create the new project.

9. Click **Choose**, then select **Users** or **Groups** from the drop-down menu in the browser.
10. Click **Search** to see a list of available users or groups.
11. Select an item from the **Available** list, then click **Move selected items to other list**.
12. Click **OK**.
13. Click **Finish** to create the new project.

8.4 Working with Project Snapshots

A project snapshot is a read-only copy of a project at a particular moment. Since snapshots are read-only, they cannot be modified or opened for editing. You can view the contents of a project snapshot as well as export and deploy a project based on a snapshot.

8.4.1 How to Create a New Project Snapshot

You can create a new project snapshot from the Project Welcome Page.

To create a new project snapshot:

Users with owner and editor permissions on a project can create new snapshots.

1. Go to the Project Welcome Page, then expand **Snapshots**.
2. Click **New**.
3. Enter a name for your snapshot, then click **Create Snapshot**.

The snapshot appears in the list of snapshots defined for this project, including the date the snapshot was created and the user ID of the snapshot creator.

8.4.2 How to View the Contents of a Project Snapshot

Viewing the contents of a project snapshot enables you to view and compare previous version of a project with the current one.

To view the contents of a project snapshot:

1. Go to the Project Welcome Page.
2. Expand **Snapshots**, then click the name of the snapshot you want to view.

Within the snapshot view, you can view the state of the processes, rules, and human tasks associated with the project.

8.4.3 How to Return to the Active Version of a Project

To return to the active version of a project from a project snapshot, click the **Back to Active Version** button at the top of the Project Welcome Page.

8.4.4 How to Delete a Project Snapshot

Using Business Process Composer a user who is granted the editor role of a project can delete the snapshots they created. A user who is granted the owner or administrator role of a project can delete any snapshot created by any user.

1. Open your project.

2. From the Project Welcome Page, expand **Snapshots**.
3. Select the snapshot you want to delete from the list, then click **Delete**.
4. Click **Yes** to confirm that you want to delete the project snapshot.

   Once you delete a project snapshot, it cannot be recovered.

8.4.5 How to Export a Project Snapshot

You can use Business Process Composer to export a project snapshot. Project snapshots are exported as normal BPM projects which you can deploy, share with other users, etc.

To export a project snapshot:

1. Open your project.
2. View the project snapshot you want to export.

   See Section 8.4.2, "How to View the Contents of a Project Snapshot" for more information.
3. From the main menu, select **Export**, then **Export Project**.
4. Choose a location on your local file system and click **Save**.

   The exported project snapshot is saved as a .EXP file on your file system.

8.4.6 How to Deploy a Project Snapshot

You can use Business Process Composer to deploy a project snapshot directly to runtime. This enables you to test or revert back to an older version of a project.

To deploy a project snapshot:

1. Open your project.
2. View the project snapshot you want to deploy.

   See Section 8.4.2, "How to View the Contents of a Project Snapshot" for more information.
3. From the main menu, select **Deploy Project**.
4. Provide the information as shown in Figure 15–1.
5. Click **Deploy**.

Part III

Defining How Users Interact with Your Business Processes

This part describes how to use Oracle Business Process Composer to define how users interact with your business processes. It contains chapters for creating and using web forms, creating form rules, and defining human tasks.

This part contains the following chapters:

- Chapter 9, "Working with Web Forms"
- Chapter 10, "Working with Web Form Rules"
- Chapter 11, "Working with Human Tasks"

9

Working with Web Forms

This chapter describes how to create web forms using Business Process Composer. It provides an introduction to the web forms designer and describes procedures for creating web forms, adding web form controls, and editing web form control properties.

This chapter contains the following sections:

- Section 9.1, "Introduction to Forms in Oracle BPM"
- Section 9.2, "Introduction to the Web Forms Designer"
- Section 9.3, "Introduction to Web Form Controls"
- Section 9.4, "Introduction to Data Sources"
- Section 9.5, "Walkthrough: Creating a Web Form Using the Form First Method"
- Section 9.6, "Walkthrough: Creating a Web Form Using the Data First Method"
- Section 9.7, "Working with Web Forms"

9.1 Introduction to Forms in Oracle BPM

Oracle Business Process Management provides functionality that defines how end users interact with the applications defined by BPMN processes. There are three components within Oracle Business Process Management that work together to define this user interaction:

- User tasks: are the BPMN flow objects that specify where in your process user interaction is required. Oracle BPM supports different types of user tasks that determine the approval process required.
- Human tasks: define how users interact with a process-based application. They define the user interface, data structures, and connectivity information. See Chapter 11, "Working with Human Tasks" for more information.
- Forms: define the interface that enables users to interact with your application. For business applications created with Oracle BPM these forms are displayed in Oracle Business Process Management Workspace.

Oracle BPM supports the following types of forms:

- Web forms: define the user interface for a human task. They are based on standard technologies, including XHTML, CSS, and Javascript. Web forms are created by process analysts using Business Process Composer. They are included within a BPM project. This chapter describes how to create and use web forms.

- Oracle Application Development Framework (ADF) task forms: are ADF components that define the user interface for a human task. ADF task forms are generally designed by process developers. They are created in Oracle JDeveloper and can be added to a BPM project using Oracle BPM Studio.

  See "Getting Started with Human Workflow" in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management for more information*.

9.1.1 Introduction to Web Forms

Web forms define the user interface that enables end users to interact with your business processes. Business Process Composer provides an editor for creating web forms. This enables process analysts to design the way users interact with a business process and also define the underlying data structure required by the application.

Web forms are based on standards, including XHTML, CSS, and Javascript which ensures compatibility across multiple platforms and browsers. At its core, a web form is an XHTML file. However, the form designer enables you to create and design a web form without interacting directly with the XHTML code. After creating a new form, use the web form designer to customize the form's appearance and behavior.

You can add the following to a web form:

- Form controls: are components that you can add directly to a form. Form controls define the graphical elements of a web form and their layout. They also display data to users and receive data input. See Section 9.3, "Introduction to Web Form Controls" for information about the types of form controls supported by Oracle BPM.

  When you add, arrange, or remove a form control from a web form, the form editor automatically updates the underlying XHTML code.

- Form rules: are pieces of Javascript code that define the behavior of a web form or web form controls. Business Process Composer provides a form rules editor that enables you to create and edit form rules. See Chapter 10, "Working with Web Form Rules" for more information.

9.1.2 Form First and Data First Design

There are two use cases for creating web forms in Oracle BPM:

- Form first

  In this use case, you create a web form first before any data elements are defined. This enables a business user to define the required user interface using Business Process Composer. In this use case, user-interface elements are created first without considering the required underlying data model.

  After the form is created, Oracle BPM automatically generates the schema that defines the data required by the web form. This data schema is based on the web form controls added to the web form. When you assign the web form to a human task, this schema is automatically used to define the human task payload.

  See Section 9.5, "Walkthrough: Creating a Web Form Using the Form First Method" for procedural information about this use case.

- Data first

  In this use case, you create a web form based on an existing human task payload. Oracle BPM generates data sources based on the payload. You can automatically

add web form controls to a form based on these data sources. One advantage of the data first model is that it enables you to reuse a schema across multiple forms.

After adding form controls from a data source, use the form designer to rearrange them within the form to define the necessary look and feel of the user interface.

See Section 9.6, "Walkthrough: Creating a Web Form Using the Data First Method" for procedural information about this use cases.

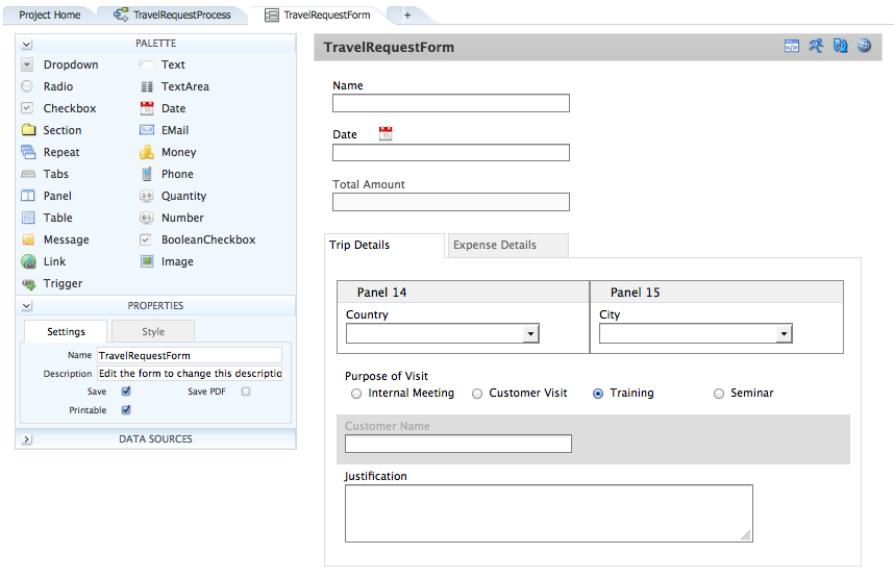
9.2 Introduction to the Web Forms Designer

Web forms define the user interface of your process based application. Business Process Composer provides a web form editor for creating web forms. Using the web forms editor you can:

- Add web form controls to a web form.
- Customize the appearance and layout of a web form.
- Create form rules that define the behavior of the controls on a web form.

Figure 9–1 shows an example of the web form designer.

Figure 9–1 Business Process Composer Web Form Designer



When you open a web form, the web form designer appears as a tabbed pane within Business Process Composer. The web forms designer is divided into the following areas:

- Web forms component palette
- Web form toolbar
- Forms canvas
- Properties editor
- Data sources

Note: Data sources are only available in web forms created via the "data first" use case. See Section 9.6, "Walkthrough: Creating a Web Form Using the Data First Method" for information about creating web forms based on a human task payload.

Each area of the web forms designer is described in the following sections.

9.2.1 Introduction to the Web Forms Component Palette

The component palette contains the controls you can add to a forms to define how your users interact with your business application. The form palette appears on the left side of the web form editor as shown in Figure 9–1. You can add a form control to a web form by dragging the control from the palette to the form canvas.

See Section 9.3, "Introduction to Web Form Controls" for a description of the web form controls that are available in Oracle BPM. See Section 9.7.1, "How to Add Controls to a Web Form" for information dragging controls to a web form.

9.2.2 Introduction to the Web Form Editor Toolbar

The web form editor contains a toolbar that provides access to form-related features. Figure 9–2 shows the web form editor toolbar.

Figure 9–2 The Web Form Editor Toolbar

The web form editor toolbar contains the following buttons:

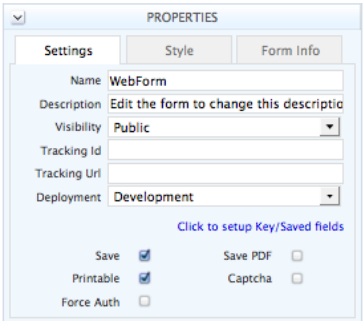
- Form rules: click to access the form rules editor. This button toggles between the form rules editor and the web form editor.
- Test web form: opens the form in a browser window. In this window, you can view how the web form appears to end users. You can also test the behavior of form rules.
- Refresh: refreshes the external data source (human task payload) of the form. If you make changes to the human task payload, you must refresh the data source of the web form. This button is only applicable to web forms created using the data first method. See Section 9.6, "Walkthrough: Creating a Web Form Using the Data First Method" for more information.
- Localization: opens the resource bundle editor. This editor enables you to localize components of the web form and web form controls.

9.2.3 Introduction to the Property Editor

Use the property editor to define the properties of a web form or web form control. When you click on a control in your form, the **Properties** area displays the control's properties so you can view and edit them.

The properties editor contains tabbed panes that display the properties of the web form or web form control grouped by function. Figure 9–3 shows an example of the properties editor.

Figure 9–3 The Web Form Designer Properties Editor



See Section 9.7.4, "How to Edit the Properties of a Web Form and Web Form Controls" for procedures about editing a web form and web form control properties. See Appendix C, "Web Form and Web Form Control Property Reference" for information about each property.

9.2.4 Introduction to the Data Source Panel

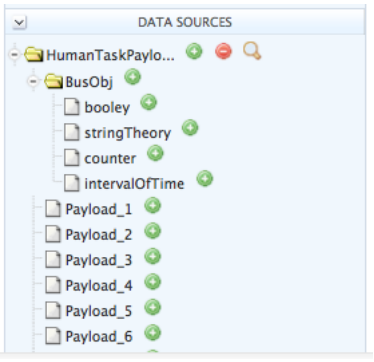
The Data Source panel displays the list of data elements that can be incorporated into your form. See Section 9.4, "Introduction to Data Sources" for more information.

From the data source panel, you can generate web form controls based on all the data elements in the payload or you can generate web form controls based on specific data elements.

> **Note:** Data sources are only available in web forms created based on a human task payload. See Section 9.6, "Walkthrough: Creating a Web Form Using the Data First Method" for information.

Figure 9–4 shows how the data source panel appears in a web form created based on a human task payload.

Figure 9–4 The Data Source Panel

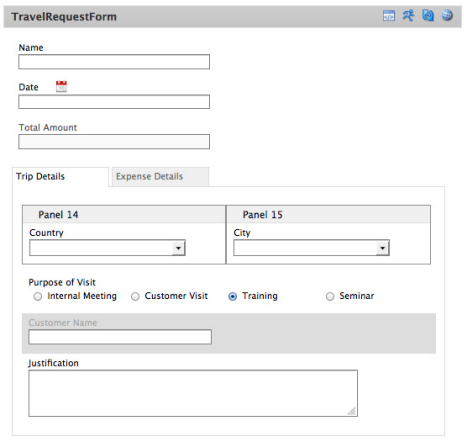


See Section 9.4, "Introduction to Data Sources" for more information about data sources in Oracle BPM. See Section 9.6, "Walkthrough: Creating a Web Form Using the Data First Method" for information on creating a web form based on a human task payload.

9.2.5 Introduction to the Form Canvas

The right side of the web form designer is the form canvas. You can create a web form by dragging controls from the component palette to the form designer canvas. Figure 9–5 shows an example of the form canvas containing several controls.

Figure 9–5 The Form Canvas



9.3 Introduction to Web Form Controls

Web form controls are the user interface components that you add to a form to define the user interface of your business application. Web form controls are defined in HTML, but using the web form editor, you do not have to edit any of the underlying code.

Web form controls are described in the following sections. For information on the properties that you can configure for each web form control, see Appendix C.2, "Web Form Control Properties".

9.3.1 Input Controls

Input controls enable users to enter data (text, dates, numbers, etc.) into a form. Input controls automatically prevent users from entering the wrong data types. For example, if a user enter letters into a number control, the form displays an error message and the Submit button will not be enabled until valid data is supplied. This validation happens automatically. The function of each input control is described below.

To validate the content in each input control, Oracle BPM assigns a default XML schema data type to each control. The table below shows the default data types for each input control.

Input Control	XML Schema Type
Text	xsd:string
Date	xsd:date
EMail	types:emailType, an xsd:string restriction pattern defined in types.xsd.
Money	types:number, an xsd:double restriction defined in types.xsd.
Phone	types:phoneType, an xsd:string restriction pattern defined in types.xsd.

Quantity	xsd:integer
Number	types:number, an xsd:double restriction defined in types.xsd.
Boolean	xsd:boolean

9.3.1.1 Text

The text control enables users to enter text and is primarily intended for short, one-line entries.

9.3.1.2 TextArea

The textarea control enables users to enter any text and is designed for longer, multi-line submissions. When a user enters data, a scroll bar appears, if necessary, to accommodate the text. The textarea control uses a property called **# of Rows** that controls the default number of lines visible in the input area.

In HTML there is no property for setting the maximum length on a textarea control. Therefore, this control does not have the maxlength property like the text control. If you need to specify a maximum length, you should use the text control, or use a form rule to limit the amount of text a user can enter. See Chapter 10, "Working with Web Form Rules" for more information.

9.3.1.3 Date, Time, Date/Time

The date control enables users to enter information as a date, time or date plus time control. Use the Control Type property to select from a drop down of choices: date, time or date/time.

9.3.1.3.1 The Date Control

Use the date control type to display the date input field with a calendar icon. If a user clicks the calendar to choose the date, the date selected is displayed in the default mm-dd-yyyy format, for example 09-01-2008. If a user manually enters a date, it can be in any of the formats listed below. However, the date is reformatted to the mm-dd-yyyy format.

Valid date formats use three different separators (. / -) and three date formats (DD/MM/YYYY, MM/DD/YYYY, and YYYY/MM/DD). The following examples show how dates can be specified:

- mm-dd-yyyy (07-26-1966)
- mm/dd/yyyy (07/26/1966)
- mm.dd.yyyy (07.26.1966)
- dd-mm-yyyy (26-07-1966)
- dd/mm/yyyy (26/07/1966)
- dd.mm.yyyy (26.07.1966)
- yyyy-mm-dd (1966-07-26)
- yyyy/mm/dd (1966/07/26)
- yyyy.mm.dd (1966.07.26)

The date is converted to the standard xsd:date format of yyyy-mm-dd in the submissions XML document using the following format:

```
<Order Date>2012-03-06</Order Date>
```

9.3.1.3.2 The Time Control

Use the time control type to display only the time input field on your form. The Time Format property drop down menu appears, enabling a user to select between four variations of military and standard conventions. The Time Format control defaults to hh:mm using military time, for example 18:30. Other format options include standard time (AM/PM).

Changing the control type to Time does not display the date picker.

The Time Format control has four standard format using both military and standard time. There are two separators, the dot and the colon. Examples are shown below:

- hh:mm (18:30)
- hh.mm (18.30)
- hh:mm - AM/PM (06:30 PM)
- hh.mm - AM/PM (06.30 PM)
- hh:mm:ss (18:30:15)
- hh.mm.ss (18.30.15)
- hh:mm:ss - AM/PM (06:30:15 PM)
- hh.mm.ss - AM/PM (06.30.15 PM)

The local time is converted to UTC format in the submissions XML document. Here is an example: **<Order Time>15:42:00Z</Order Time>**. The capital "Z" after the time is necessary for proper initialization.

9.3.1.3.3 The Date/Time Control

Use the **Date/Time** control to display two input fields: one for the date and another for the time. A property drop down menu enables you to select from nine valid formats for the date of the control and four choices in military and standard time for the time. See Section 9.3.1.3.1, "The Date Control" and Section 9.3.1.3.2, "The Time Control" for more information.

Default formats for date and time remain mm-dd-yyyy and hh:mm.

- When you enter a date in the date part (or select with the picker), it automatically completes the time portion of the control with a value for 12:00 AM. The value displayed depends on the time format selected:
  - **00:00** displays if the selected time format is hh:mm
  - **12:00 AM** displays if the selected time format is hh:mm - AM/PM
  - **00:00:00** displays if the selected time format is hh:mm:ss
  - **12:00:00 AM** displays if the selected time format is hh:mm:ss - AM/PM
- You cannot enter a time value without a date value.
- Changing the Control Type to Time does not display the date picker.
- The time input field cannot be labeled. You must ensure that the label for the date portion is descriptive enough to include the time portion. If date and time labels for the appropriate input fields are required, two separate controls must be used or the label for the date portion should be extended over the time input field.

The local time is converted to and saved in UTC format and the date will be converted to the standard xsd:date format of yyyy-mm-dd In the submission XML document, for example:

```
<OrderDate>2012-01-01TO05:00:00Z</OrderDate>
```

The capital "Z" after the time is necessary for proper initialization.

Form rules can be applied to the Date/Time control in all variations. Form rules should execute in the form time zone.

9.3.1.3.4 Time Zones

In general, form data is displayed in the browser's time zone and saved in UTC in the submission document. The form's time zone is calculated automatically from the browser's time zone. The time zone, once set, cannot be changed for that form instance. You can override the browser's time zone by appending the **&_formTz=<tz>** parameter to the form URL.

Modern Time Zone Strings must be used with the &_formTz parameter. Other time zone formats are not supported.

9.3.1.4 Email

The email form control enables users to enter a valid EMail address. The address must conform to the following syntax: <name>@<name>.<string>.

9.3.1.5 Money

Use the money form control to enable users to enter U.S. currency. Users can enter commas or decimal point. If a user does not enter them, the form displays these symbols automatically. For example if a user types 4000, the form will display the value as 4,000.00. The form also rounds all entries to two decimal places.

If a user types more than two digits after the decimal place the XML submissions document will store as many digits as the user entered but will not include the dollar symbol, decimal point or commas.

9.3.1.6 Phone

The phone web form control enables users to enter a phone number using any of the following formats: xxx.xxx.xxxx, xxx-xxx-xxxx, xxx.xxxx, or xxx-xxxx. To enforce one of the 10-digit formats (to require an area code), edit the control's Pattern property.

9.3.1.7 Quantity

The quantity web form control enables users to enter quantities or any whole numbers (integers). The form displays an error message if users enter decimal points, commas, or anything other than an integer.

9.3.1.8 Number

The number web form control enables users to enter decimal numbers. Users may enter any number of digits after the decimal place.

9.3.2 Selection Controls

Selection controls enable user to select from a list of several options. Oracle BPM supports the following selection controls:

- Dropdown
- Radio
- Checkbox

- True/False

Note: The web form control palette does not include a combo box control which is a drop down list in which a user can type a new value or option. Combo boxes are not part of standard XHTML.

However, it is possible to use a combination of drop downs, radio buttons, or check boxes with an "other" option in combination with form rules. If a user selects "other," the rule then displays a text control (such as details or new entry) the user can enter.

9.3.2.1 Dropdown

The dropdown control adds a drop down list to a form. By default the first choice in the drop down list is blank. You must define the other choices by editing the control's Options properties.

9.3.2.2 Radio

The radio control adds mutually exclusive radio buttons. You must define the number of radio buttons and the specific choices by editing the control Options properties.

After selecting a radio option as the default, to change it, you must remove that option from the control and tab out of the options property so that it is removed from the control. You can then add the option back to the control's option list.

9.3.2.3 Checkbox

The checkbox control adds a set of check boxes enabling the user to select one or more options. Like other selection controls, you edit the control's Options properties to define the number of check boxes and define the options available.

9.3.2.4 BooleanCheckbox

The booleancheckbox control enables a user to select a true or false value in a control instead of entering data. The control options default to "true=Yes" and "false=No". You can change the option labels to other values if necessary. However, the option values cannot be changed and remain true or false.

Selecting the "Yes" checkbox stet the value to true value in the XML document for the control data while leaving it deselected defines a false value. No value appears in the XML document.

9.3.3 Group Controls

Use group controls to organize your forms based on the types of information you need to present to your users.

9.3.3.1 Sections

Use the sections control to create groups of controls that users can expand and collapse.

In a web form, users can click the expand icon to expand and collapse sections. You can specify a default value to determine if the section is initially expanded or collapsed.

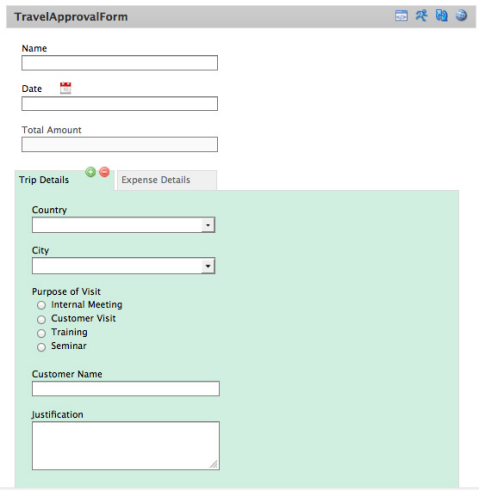
After dragging a section control into your form, you can drag any other controls inside it, including panels and other section controls. If you have a required control inside a

collapsed section, the section label turns red to cue users that they must expand the section and supply the required information before they submit the form. If you delete a section control while designing a form, all controls within the section are deleted also.

9.3.3.2 Tabs

You can use the tabs group control to create a tabbed view as shown in Figure 9–6.

Figure 9–6 A Web Form Containing Multiple Tabs



By default, when you drag the tab control into a form there are three individual tabs. To add or remove a tab, click the tab and then click the Add (+) or Delete (-) icon. Additional tabs are added to the right of the tab from which you clicked the add icon.

To rearrange the tab order, drag one tab on top of another tab. The tab you dragged will move to the right of the tab upon which it was dropped. You can drag in other controls, including other group controls, into any individual tabs. Users see only those controls in the currently selected tab.

To move a group of tabs to another area of a form, click the area to the right of the tab drag the entire group to the desired location.

9.3.3.3 Panels

The panels control creates columns within a web form. You can add multiple columns to a form by dragging in as many panels as necessary.

By default a panel's width is set to 49%. When you drag two panels into your form, the second panel is automatically aligned with the first panel. Panels have a 1px border so to make their boundaries visible. Therefore, in a two-column layout you cannot make both widths 50%.

Since panels are group controls, you can drag other controls inside them. If you want to rearrange the order of your panels, you should remember that the drag-and-drop restriction that prevents you from dropping a control below a group control. If you have three columns and want the middle column at the far right, you must drag the middle column above whichever control is directly beneath the panel.

In a three or four column layout, to move one of the middle columns or the right-most column to the far left, you must drag and drop it above your left-most column.

Although panels have labels you can edit during design, the panel labels and panels themselves are not visible to users at runtime or when testing a form. Only the controls inside a panels are visible. These controls are organized visually according to the width of the panels.

If you delete a panel, any controls you've dragged inside it are automatically deleted.

9.3.3.4 Tables

Use the tables layout control to conserve space within a web form. This control enables you to arrange the form controls in a grid pattern.

You can edit the table name and column names. You can also drag and drop new controls from the palette, and set the widths of the columns. You can control the minimum and maximum number of rows in the table. The **Add** and **Remove** icons automatically appear to the left. You can also use form rules to calculate values, enable or disable fields, show or hide fields, etc.

When you drag the table control into a form, by default, it has three rows and three columns. The columns contain the default names col 0, col 1 and col 2. All of the cells in the table are defined as required. The Add and Delete icons are displayed for each row in the table, enabling you to add and delete rows, depending on the Min/Max property values.

Rearrange table columns by clicking on the green arrow that appears when you click in the column heading of the column that you wish to move. Columns move to the right until the last column position in the table is reached.

Table controls have the following limitations:

- Table controls cannot be used in a repeat control.
- Table cell borders are not shown in print view.
- Table column names cannot be edited from the work area. Use the property editor.

9.3.3.5 Repeats

Repeating controls dynamically display multiple copies of a web form control. This is useful when you want to enable users to enter multiple information of the same type, for example, a phone number. A repeat control can dynamically display as many controls as need rather than having to explicitly add extra input controls. Repeat controls are used to repeat data elements and sections within a form.

You can add a repeat control to a web form, then drag and drop additional controls into the repeat control in the same way that you add controls to a tab, panel, or section. You can surround an individual control with a repeat control, or it can include an entire section control. This section control should contain all of the web form controls that need to be repeated.

After adding a control to a repeat control, you can select the control to view and edit its properties. Web form controls added to a repeat control contain two additional properties:

- **Min#**: Specifies the minimum number of times the control can be repeated. The default value for Min# is 0
- **Max#**: Specifies the maximum number of times the control can be repeated. The default value of Max# is 1.

You must edit this value to define how many times the control is repeated. IF you do not edit the value, the control does not repeat.

These properties define how the control appears to end users. When the value of these properties is greater than one, the user sees an add icon that automatically creates another control. In the form editor, clicking on the control automatically increments the property.

The Min# value defines the minimum number of controls that must be added and filled-in by the end user. If the user does not fill in data for the minimum number of controls, the Submit button is not enabled.

If a users adds controls beyond the number defined by Min#, they must fill in the data for the first controls up to the value of Min#. For example, if Min# is defined as 3 and the user adds 5 controls, the first 3 controls, in order starting from the top of the form, must contain data in order for the Submit button to be enabled. If a user adds controls up to the number defined by Max#, the add icon is no longer displayed.

To explain to users how to add additional data elements, you can provide a message using a message control The following example shows one way of doing this:

```
<center>Click on the <img src="/frevvo/images/icons/add.gif"/> icon to addmore
phone numbers to the list.<br/></center>
```

When you edit the label or any properties of a control inside a repeat control, your changes are applied to every instance of the control inside the repeat control. If you delete a control inside the repeat control, all instances of the control are deleted.

Like panel controls, repeat controls are hidden when testing or using forms. Only the controls within the repeat control are visible.

There are some key differences between repeat controls and the other grouping controls:

- You cannot add button, message, or image controls.
- A repeat control can contain only one web form control.l You cannot add tab controls, panels or other repeat controls. To add multiple controls to a repeat control, you can add multiple controls to a section control, then add the section to a repeat control.

  If you need to add more controls to the section, you must drag the section out of the repeat control, add the additional controls, then drag the section control back into the repeat control. However, you can add a section control to a panel control without having to remove the section out of the repeat control.

When adding a repeat data element that is part of a data source, BPC automatically generates a repeating item in the control The schema of the repeat control automatically enforces the minimum and maximum requirements.

9.3.4 Other Controls

In addition to input, selection, and group controls, Oracle BPM provides other web form controls to define how users interact with a web form.

9.3.4.1 Message Control

Use the message control to add static text on your form. You must provide the text in the control's Message property. You can include XHMTL with your text. For example you may want two lines with different font sizes or colors, for instance.

The browser will format the XHMTL when users access the form. For example, if you wanted to create a header in a web form, you can use a combination of panels and message controls as shown in the following example.

```
<center style="font-weight:bold;">
Connecticut Surgeons, LLC
</center>
<center style="line-height:1.2em;">
82 Anderson Road<br/>
Branford, CT 06180<br/>
</center>

<center style="font-weight:bold;">
Connecticut Surgeons, LLC
</center>
<center style="line-height:1.2em;">
82 Anderson Road<br/>
Branford, CT 06180<br/>
</center>
```

9.3.4.2 Link Control

Use the Link Control to include a URL in your form. When a user clicks the link, the target URL opens in a separate browser window.

9.3.4.3 Button Control

Use the button control to add a button to your form. This control is primarily used in conjunction with form rules.

Note: The button control does not work in repeating items.

9.3.4.4 Image Control

Use this control to include an image (picture, logo, etc.) in your form. The image control enables you to add JPG, GIF, and PNG files or any other image type that your browser supports.

When you drag in the image control, a **Browse** button and an **Upload Image** button appear. Click Browse, navigate to the image you want, and click Upload Image. After uploading the image, the **Browse** and **Upload Image** buttons are no longer available. To change the image you must delete the image control and drag in a new one.

Before uploading images, you should ensure that they are sized to fit within a form. The standard form size is 600 pixels. You can resize an uploaded image by selecting the image control in the designer, clicking on the style tab in the properties panel and setting the width.

Units are px, % or em. You must include the units. For example, specifying a value of 50% will resize the image to half the width of the form or panel.

9.4 Introduction to Data Sources

Data sources enable you to create web form controls based on existing data structures. In Oracle BPM these data structures are defined by a Human Task payload. After creating a human task payload, you can create a new web form based on the payload data. See Section 9.6, "Walkthrough: Creating a Web Form Using the Data First Method" for information about creating a web form based on payload data.

9.4.1 Web Form Controls Generated by Payload Data Types

Table 9–1 shows the type of web form control generated for each data type supported by human task payloads.

Table 9–1 Human Task Data Types and Their Corresponding Web Form Control

Data type	Web form control
String	Text control
Binary	Text control
Boolean	Booleancheckbox. If checked, the value is set to 'true' and if unchecked it is set to 'false' if the element is required. No value is specified if the control is optional. You can set the label of the true and false options via the Labels property. However only the first option which maps to the 'true' choice is displayed. If you add more then two labels the extra labels are automatically removed
Int / Int64	Text control
Real / Decimal	Text control
Interval	Text control
Time	Date control
Components	Section Human task payload data created based on a business object is added to a section form control. Individual data elements within the component generate the same form controls as other data types. These are added to the section control. After adding them to a form, you can rearrange them as necessary.

Most of the controls generated from a human task payload are created as text controls. However all validation for the controls will be based on the original data type datatype. For example an xsd:integer type will only allow numeric values.

Schema elements that specify element restrictions are generated as selection controls. The control is created as a dropdown if there are four or more valid choices and a radio button if there are fewer than four valid values. You can change the display type between radio and dropdown and checkbox using the Display As property.

9.4.2 Modifying Web Form Controls Generated From Data Elements

After generating web form controls from data elements you can:

- Update the data payload of the human task and recreate the web form.
- Edit the controls in the web forms editor.

9.4.2.1 What You Can Edit Using the Web Form Editor

You can perform the following on a web form control after generating it from a data element:

- Edit the web form control's name. You can edit the name that is displayed in the web form editor. However, the underlying XSD is not changed.
- Create or edit web form rules used by the web form control.
- Set default values for the web form control.

- Edit the layout of the web form controls within the web form. You can move controls in sections or panels as required. However, you cannot add or remove a generated web form control from a repeat control.
- Edit the Display As property of the web form control. See Section 9.4.3, "Introduction to the Display As Property" for more information.
- Edit the web form control's label.

9.4.2.2 What You Cannot Edit Using the Web Form Designer

The following changes cannot be made when generating web form controls based on data elements:

- Make edits to web form controls that alter how the web form or web form controls are validated.
- Alter the number of repeats of a repeating control.
- Add additional web form controls that require changes to the underlying data source.

9.4.3 Introduction to the Display As Property

The Display As property is only supported by web form controls generated from a data source. Use the Display As property to change the way your control appears on a web form. For example, a text control can be changed to a dropdown control or vice versa. This changes the control's appearance on the form, but does not change how the control validates data. To modify how the control is validated, you must redefine the human task payload data and recreate the web form.

To see the control's underlying data type and the global element to which it is bound, hover over the property tab. The expanded Data Source s displays the XSD path as a tool tip.

If you set the Display As property to a select web form control (radio, dropdown, or checkbox), the Label property is enabled. By adding strings into the labels property you can restrict the values allowed to be entered into this control when users use your form. When the form is submitted the value will be equivalent to the selected option label.

Another way to set the labels is via a rule to dynamically initialize the options when the form loads. See Chapter 10, "Working with Web Form Rules" for more information.

The Display As property is not available for the following controls when based on a data type:

- Message
- Repeat
- Date, Time, or Date/Time
- Boolean

9.5 Walkthrough: Creating a Web Form Using the Form First Method

You can use Business Process Composer create a web form that is not based on pre-existing data. See Section 9.1.1, "Introduction to Web Forms" for information about the use cases for creating web forms.

Use the form first use case, you create a web form directly from the **Project Home Page**. After creating the web form, you use the web form designer rearrange the form controls as required.

To create and edit a web form with no previously defined data:

1. Create a new web form using the **Project Welcome Page**.
   a. Ensure that you are editing the project.
   b. From the **Project Welcome Page**, select the **Web Forms** tab.
   c. Click **New Web Form**, then enter a name.
   d. Click **Create**.
2. Add controls to a web form.

   After creating a new web form, you can add form controls as necessary to define how users interact with your application. See Section 9.7.1, "How to Add Controls to a Web Form" for more information.
3. Edit web form and web form control properties.

   You can edit the properties of your web forms and its controls to determine their behavior and appearance. See Section 9.7.4, "How to Edit the Properties of a Web Form and Web Form Controls" for more information.
4. Add form rules to a web form.

   You can add form rules to your web form to further customize and control its behavior. See Section 10.2, "Working with Form Rules" for more information.
5. Test your web form.

   You can use the web form designer to preview how your web form will appear to end users. See Section 9.7.7, "How to Test a Web Form" for more information.
6. Save your web form.

   To save a web form, you must save the entire project. See Section 4.4.7, "How to Save Changes to a Project" for more information.
7. Assign your web form to a human task.

   After completing your web form, you can assign it to a human task using the human task editor. When you assign the web form to a human task, the human task editor automatically generates the payload information based on the controls you added to the web form. If the human task already has an existing payload defined, it is replaced by a new payload defined by the web form.

   See Section 11.3.11, "How to Specify the Presentation of a Human Task" for information about assigning a web form to a human task.

9.6 Walkthrough: Creating a Web Form Using the Data First Method

You can use Business Process Composer to create a new web form based on an existing human task payload. See Section 9.1.1, "Introduction to Web Forms" for information about the use cases for creating web forms.

Human task payloads are created using the human task editor. When creating web forms based on a payload, the payload becomes read-only. The following conditions apply:

- You cannot update the payload signature after you create the web form. In other words you cannot add or remove data objects nor change their type. However, if you have defined complex data types in the human task payload, you can modify them by editing the underlying business object.
- After creating a web form based on a human task payload, you cannot add web form controls to a form that require underlying data structures. Any additional data required by the adding web form controls to the web form using the form editor is not be added to the underlying schema and will not be available in the human task payload.

If you have not created the human task payload based on complex data objects, to modify the payload you must dissociate the human task from the web form, make the modifications to the payload, and create a new web form based on this new payload.

To create and edit a new web form based on an existing payload:

1. Create a new human task and define its payload.
   a. Create a new human task. See Section 11.3.2, "How to Create New Human Task" for more information.
   b. Create the payload you want to use as the base for the new web form. See Section 11.3.10, "How to Create and Configure the Data Payload for a Human Task" for more information.
   c. Open the human task. See Section 11.3.3, "How to Open a Human Task" for more information.
   d. Select the **Basic** tab.
   e. In the **Presentation** area, select **Web Form**, then click the **Add** icon.
   f. Enter the name of the web form, then select **Based on payload**.
   g. Click **Create**.

      The web form is created and assigned to the current human task.
   h. Click **Edit** to edit the web form.

   The web form editor displays a blank form.
2. Generate web form controls from payload data.

   After creating a web form based on a payload, you can use the payload data to generate the corresponding web form controls. The payload data appears in the Data Structures pane on the left side of the web form editor.

   See Section 9.7.2, "How to Add Controls Based on Data Sources" for more information.
3. Add additional web form controls.

   You can add additional form controls as necessary to define how users interact with your application. See Section 9.7.1, "How to Add Controls to a Web Form" for more information.

   Note: You can add additional controls to refine the look and feel of your web form. However, any data controls that you add will not be reflected in the underlying payload.
4. Edit web form and web form control properties.

You can edit the properties of your web forms and its controls to determine their behavior and appearance. See Section 9.7.4, "How to Edit the Properties of a Web Form and Web Form Controls" for more information.

5. Add form rules to a web form.

   You can add form rules to your web form to further customize and control its behavior. See Section 10.2, "Working with Form Rules" for more information.

6. Test your web form.

   As you are designing a web form, you can test its appearance and the behavior of form rule. See Section 9.7.7, "How to Test a Web Form" for more information.

7. Save your web form.

   To save a web form, you must save the entire project. See Section 4.4.7, "How to Save Changes to a Project" for more information.

9.7 Working with Web Forms

After creating a new web form, you can use the web form editor to add web form controls, edit form control properties, and test the web form.

9.7.1 How to Add Controls to a Web Form

You can add controls to a web form by dragging and dropping from the web form control palette to the web form canvas.

To add controls from the component palette:

1. Open the web form where you want to add controls.
2. In the component palette click the control you want to add, then drag it to the form canvas.

   As you begin to drag a control from the component palette, the position cursor changes depending on its position in the form canvas. The different position cursors are:

3. Release the mouse button to add the new control.

9.7.2 How to Add Controls Based on Data Sources

After creating a web form based on the payload data of a human task, you can automatically generate web form controls for the data elements defined in the payload. You can generate form controls from all of the data elements or individual data elements.

See Section 9.6, "Walkthrough: Creating a Web Form Using the Data First Method" for more information.

To generate form controls for all the data elements in a payload:

1. Open the web form you created based on a human task payload.
2. In the Data Sources pane, click the green **Add (+)** icon to the right of the **HumanTaskPayload** node in the data elements tree as shown in Figure 9–4.

   Business Process Composer generates a web form control for each of the data elements in the payload. The controls are added to the web form above the currently selected control. See Section 9.4, "Introduction to Data Sources" for

information about which web form control is generated for each data type of the payload.

3. Edit the layout of the controls as necessary by dragging and dropping them within the web form.

> **Note:** You cannot add a control generated from a data element into a repeat control.

To generate form controls for individual data elements in a payload

1. Open the web form you created based on a human task payload.
2. In the Data Sources pane, click the bullet to the left of the **HumanTaskPayload** node as shown in Figure 9–4.

   Business Process Composer displays a tree showing a list of the data elements within the payload.
3. For each of the data elements that you want to add to your form, select the green **Add** (+) icon to the left of the data element.

   Business Process Composer generates a form control for the data element. When adding data elements individually, the form controls are added directly at the beginning of the form. They are not added to a form panel.
4. Edit the layout of the controls as necessary by dragging and dropping them within the web form.

> **Note:** You cannot add a control generated from a data element into a repeat control.

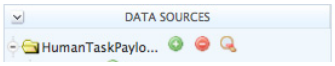
9.7.3 How to Show Which Web Controls Were Created from a Data Source

The web forms editor enables you to view which web form controls were automatically generated from a data source.

To show which web controls were created from a data source:

1. Open your web form.
2. Expand the **Data Sources** panel.
3. Click the **Show Form Controls** icon as shown in Figure 9–7.

Figure 9–7 The Show Form Controls Icon



Each generated web form control is highlighted in blue in the web form canvas.

9.7.4 How to Edit the Properties of a Web Form and Web Form Controls

You can edit the properties of a web form to configure general settings and style for the web form. You can also edit the properties of a web form control to configure how the control appears in your web form.

To edit the properties of a web form:

You can edit the properties of a web form to configure settings for the form as well as general style properties. See Section C.1, "Web Form Properties" for information on the properties supported by Oracle BPM.

1. Open the web form whose properties you want to edit.
2. Click the web form header at the top of the form canvas.

   The properties tabs for the web form appear in the properties editor.
3. Edit the properties for the web form in the properties editor.

To edit the properties of a web form control:

You can edit the properties of individual form controls to configure its behavior and style. See Section C.2, "Web Form Control Properties" for information on the properties available for each form control.

1. Open the web form containing the control you want to edit.
2. In the form canvas, select the control.

   When you select a control in the form canvas, it is highlighted.
3. In the properties editor select either the Settings or Style tab, then edit the properties as necessary.

   See Appendix C.2, "Web Form Control Properties" for a description of each supported web form control property.

9.7.5 How to Delete a Web Form

You can delete a web form your BPM project from the **Project Home Page**.

> **Note:** You cannot delete a web form that is associated with a human task. If you want to delete a web form associated with a human task, you must first disassociate it using the human task editor.

To delete a web form:

1. Go to the **Project Home Page**.
2. Select the **Web Forms** tab, then position the cursor over the web form you want to delete.
3. Click the **Delete** icon, then click **OK** to confirm that you want to delete the web form.
4. Save your project. If you do not save your project to ensure that deletion of the web form is preserved in your project.

9.7.6 How to Remove a Control from a Web Form

You can remove form controls from a web form using the web forms designer.

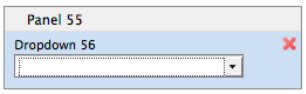
> **Note:** After deleting a control you it cannot be undone. You must add a new control to the web form and configure its properties.

To remove a control from a web form or organizational control:

1. Open the web form whose control you want to delete.
2. Click on the control to select it.

   When you select a control, it is highlighted as shown in Figure 9–8.

Figure 9–8 Selecting an Individual Control



> **Note:** Ensure that you select the individual control you want to delete and not the organizational control.

3. Click the **Click to remove** button.

To remove an organizational control from a web form:

1. Open the web form containing the organizational control you want to delete.
2. Click on the organizational control you want to delete.

   When you select an organizational control, the control and all of the controls it contains are selected as shown in.

Figure 9–9 Selecting an Organizational Control

3. Click the **Delete** button.

> **Note:** When you delete an organizational control, all the controls it contains are also deleted.

9.7.7 How to Test a Web Form

While designing a web form, you can test its behavior directly in the web form editor.

To test a web form:

1. Open your web form.
2. Click the **Test Form** button in the web form editor toolbar.
3. Test your web form by entering date, selecting items from dropdown controls, etc.
4. Click the Close (x) button when you are done testing.

10

Working with Web Form Rules

This chapter describes how to create and edit form rules within a web form using Oracle Business Process Composer. Form rules are pieces of Java-script code that enable you to define how users interact with your web forms. This chapter assumes that you are familiar with the basics of Javascript.

This chapter contains the following sections:

- Section 10.1, "Introduction to Form Rules"
- Section 10.2, "Working with Form Rules"

10.1 Introduction to Form Rules

You can use form rules to define the behavior of a web form. Typical uses of form rules include:

- Adding dynamic behavior to a web form, including showing or hiding elements or enabling or displaying elements. For example, you can add form rules that will show or hide certain form controls or entire sections based on the state of other form controls. Displaying form fields in a context-sensitive manner reduces clutter and makes it easier for users to navigate your forms.
- Performing complex calculations. For example, you can compute the total invoice price on an order form from values of other form fields such as item quantity and price.
- Performing complex validations.
- Populating a dynamic drop down list.

Form rules are pieces of server-side Javascript code that enable you to define how users interact with your web forms by defining how web form controls are displayed and defining the types of data users can enter.

The following sections describe the basic functionary of Javascript within form rules. See Appendix D, "Web Form Rules Examples" for information about specific use cases of form rules.

10.1.1 Form Rule Javascript Syntax

In Oracle BPM, a form rule is a JavaScript code. Form rules are saved and runnable after you save your BPM project.

Form rules generally have the following form:

```
if (condition)
{
```

```
    true actions;
}
else
{
    false actions;
}
```

You can create more advanced form rules primarily for the purpose of looping over repeating items. Following are some basic characteristics of JavaScript that you should be aware of when writing form rules:

- Case-sensitivity: Javascript commands are case-sensitive. For example, var color1 and var Color1 are two different variables
- Loosely typed variables: In Javascript variables are loosely typed. Variable types do not have to be explicitly declared. For example, var color = 'red'; and var num = 33 cause the variable color to be a string type and num to be a numeric type.
- End of line semicolons: End-of-line semicolons are optional. However you should use semicolons to terminate lines as part of good coding practice. This often prevents mistakes.
- Comments: You can add comments to your code using either // or /**/ syntax

Oracle BPM does not support the following Javascript syntax:

- switch statement

Additionally, Oracle BPM supports the following syntax with some limitations:

- try-catch

For example, in the following example, the value of the control named "FN" is set to 'got exception' because the JavaScript variable named x is undefined in this form rule.

```
if (form.load) {
    try {
        x.randomproperty = 'blah';
    } catch (e) {
        FN.value = 'got exception';
    }
}
```

However, catching an exception when a form control is not defined is not supported and may behave unexpectedly. This is because Oracle BPM catches the problem while parsing the form rule before the JavaScript interpreter catches the error.

In the following example, the control named Color does not exist in the form.

```
if (form.load) {
    try {
        Color.value = "red";
    }   catch(e) {
        msg1.value = "error caught: " + e;
    }
}
```

10.1.1.1 Control Name

Form rules often need to reference form controls. You must assign a control a name using the control's name property.

Names are case sensitive. If your control is named FirstName then you must write the form rule as FirstName.value. firstname.value will not work.

When using a control in a form rule, you must ensure that the control has a unique name. If multiple controls have the same name, the run time environment cannot determine which control the form rule refers to. Form controls added to a form from palette are usually guaranteed to have a unique name. The web form editor does not allow you to change the name of a control to one that already exists in your form.

However there are several contexts where you can have multiple controls with the same name:

- Controls added from XSD data sources
- Controls added from the custom palette
- Controls nested in Sections

If there are two controls with the same label and at the same level, the control's name will automatically be made unique. If you try to edit the name such that it would no longer be unique, the web forms designer prevents you from making the change.

When a control is dropped inside a section control, it is at a different nesting level then a control dropped outside a section. Also two controls, one inside a section and another outside the section are also at different nesting levels. The web form designer enables you provide identical names to the controls.

However, if non-uniquely named controls are used in form rules there may unexpected results. If you encounter errors in a form, you can edit the control names names to make them unique.

Note: Editing the name of a from xsd schema control has no effect on the xml instance document created when the form is submitted nor on the xml validation

---

10.1.1.2 Form Rule Identifiers

Form rules refer to form controls using the Name property. If you have a control where the Name property is defined as "MyControl," you can refer to properties of this control in form rules using the name as an identifier.

Form rules identifiers must always be of the following form:

```
Name.<property>
```

The following form rule identifier properties are supported:

- visible: Set to false to hide a control and true to make the control visible.
- value: Read or set the value of a control. This property is not applicable to sections, tabs and other controls that do not have values displayed to the user.
- enabled: Set to false to disable (grey out) a control so that a user can not change its value and true to enable it. This is not applicable to sections and tabs.
- expanded: Set to false to collapse a group control (sections controls only) and true to expand a group control.
- selected: Set to true to designate a tab the selected tab (tab controls only).
- valid: The value of this property is true if the control contains a valid value otherwise its value is false. Validity is based on the control's type. For instance a

numeric control will be invalid if the user enters a string value that cannot be converted to a number. This property can be set and read.

- required: Set to true to make a control required and display the red asterisk. This property only affects palette controls and not controls generated from XSD schema data source.

  This property is also valid for section controls. Setting a section required to false automatically sets all inner controls to not required.
- options: Enables dynamic setting select control options (radio, dropdown and checkbox controls only).
- label: Sets the label seen on any control including sections.
- help: Sets the help text.
- hint: Sets the hint seen on hover.
- status: Sets the error message display whenever the control's value is invalid.
- clicked: Used by the trigger controls. Its initial state is false. When a user clicks a trigger its state turns true.
- printable: Set to false to remove the control from both the printable view and PDF submission document.
- itemAdded: Used by repeat controls. Its initial state is false. When a user clicks "+" to add a repeat item AND when a repeat item is added via a Document URI as the form loads its state turns true.
- itemRemoved: Used by repeat controls. Its initial state is false. When a user clicks "-" to delete a repeat item its state turns true.
- itemIndex: Used by repeat controls. When an itemAdded or itemRemoved event fires the value of itemIndex is set. For itemRemoved events itemIndex will return -1. For itemAdded events itemIndex will give you the index of the added item
- form.load: This property is true when the form is first loading. It is useful for setting default values via form rules that you need to be set before the user starts interacting with the form.
- form.unload: This property is true when the users clicks the form's submit button. It is useful for setting control values just prior to the form's Doc Actions and Form Actions are executed.

Examples of identifiers used in form rules are:

- FirstName.value
- BillingAddress.visible
- Email[1].value
- Email[i].visible

The latter two are examples of repeating controls. We will discuss repeating controls in more detail below. Note that the case of the properties is important. FirstName.value is a valid form rule identifier but FirstName.Value or FirstName.vAlUe are not.

10.1.1.3 Strings and Numbers

Because Javascript is a loosely typed language there may be situations where you are need to add field values and the form rule performs string concatenation instead. There are several way tell the form rule to perform mathematical caluclations instead of string manipulation. One simple way is by adding a *1 to the variable. id = id*1 + 1;

will ensure that id equals the current value plus one rather than the current value with a 1 appended. Ex: If the current value was 4 and you didn't write id*1 the result may be "41" rather than 5.

You may also encounter a situation in a form rule in which money controls perform a string concatenation rather than addition. To correct this:

- Open the form with the form rule in the Designer, change the money controls to text controls, and then save the form.
- Open the form, change the text controls *back* to a money controls, and then save the form again.

10.1.1.4 Writing Conditions

One of the most common conditions is a form rule that executes as soon as the user enters a value in control. The test for this condition depends on if the field is a string type or a numeric type.

String Types: text, textarea, date, phone

```
if (name.value.length > 0)
```

Numeric Types: money, quantity, number

```
if (name.value != null) or if (name.value > 0)
```

are both common test conditions.

Many times the condition name.value.length > 0 can be dropped altogether and the form rule can be simplified. This form rule executes whenever a user enters a value into either the control named firstname or lastname.

```
fullname.value = firstname.value + ' ' + lastname.value;
```

10.1.1.5 Select Controls

Radio controls, dropdowns and checkboxes are all examples of select controls. Radio and dropdown controls are single select. That is, if one item in the dropdown is selected then all other items in the dropdown are deselected. The same is true for a radio. Only one radio button can be depressed at any time. Thus the ID.value of radios and dropdowns are similar to the other input and output controls. The value is a single item.

Checkbox controls are multi-select. Multiple items can be selected at any given time. Thus the ID.value of a checkbox is an array. Therefore on checkboxes a valid expression is ID.value.length which returns the number of items in the value array. And ID[0].value would retrieve the 1st item in the list and ID[1].value the 2nd and so on. If you have a checkbox control with only one checkbox and that checkbox is unchecked, the array will contain no elements. For an checkbox control a useful expression is ID.value.length == 0. Note that ID.length is not a valid expression. Also since a checkbox control is an array it is not a valid expression to write ID[0].value == . *Because if at option in the checkbox control is unchecked, it's value will not be an empty string. It will simply not exist in the array.*

10.1.1.6 Initial Control State

Every control in your form has an initial default property states for the visible, expanded, value, valid and enabled properties. However you can change a controls initial state in the form designer Edit tab. A control's initial state can be modified several ways. One way is by simply tying a value into an input control. This sets the

default value for the control when the form is first opened in use mode. Another way is by expanding or collapsing group controls. This sets the initial expanded state. The default state for the visible and enabled properties is set via the controls edit property panel. The edit property contains checkboxes for visible and enabled for controls on which those properties make sense like input controls.

10.1.1.7 Form Rules and Repeating Controls

If you have a repeating control in a form that itself contains a repeating control, you cannot apply a form rule to the "inner" repeating control, since there's no way to tell which of the inner repeating items goes with which outer repeating item.

10.1.2 Using Dynamic Content in Form Rules

10.1.2.1 Dynamic Content

Real forms often require dynamic content in dropdown list. Often based on the value entered into one form field, the values in other fields or options available in select controls need to be dynamic.

Form rules enable invocation of http gets that return X-JSON headers with JSON objects. This allows complete flexibility in assignment of default values populated into form fields such as text controls and select (radios, dropdown, checkbox) controls. You can also use http.post(), http.delete(), and http.put() in form rules, although you must use URL parameters with them, as they do not all support payloads.

Here is an example that shows the syntax of the http.get. This form rule invokes the http get which must return a JSON object. The method on the servlet can do whatever necessary such as querying a database given the itemName to retrieve the itemPrice. In this example the JSON object returned contains a field called price. The eval converts and assigns the JSON object to the javascript variable x. Then x can be used in the form rule as necessary. In this case it is used to set a value to the form field called Price.

```
eval('x=' + http.get('http://<webhost>/test/json/getPrice?itemName=' +
itemName.value));
Price.value = x.price;
```

Imagine another example where the JSON object returned in the http.get response contains an array called clients. This array can then be used to set the options in a dropdown list.

```
eval('x=' + http.get('http://<webhost>/test/json/getClients'));
Clients.options = x.clients;
```

Here is another example of a servlet that returns a JSON object after authenticating a user/password.

```
@Override
public void doGet (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
{
    try {
        String u = request.getParameter("username");
        String p = request.getParameter("password");
        if (authenticate(u, p) == null)
                response.addHeader("X-JSON", "{auth:false}");
        else
```

```
                response.addHeader("X-JSON", "{auth:true}");
    } catch (Exception e) {
        throw new ServletException(e);
    }
}
```

This servlet could be used in a form rule as follows:

```
if (signForm.clicked)
{
    eval('x=' + http.get('http://<webhost>/MYservices/signForm?username=' +
u.value + '&password=' + p.value));
    if (x.auth)
    {
        m.value = "<center>Authenticationn Succeeded</center>";
    } else
    {
      m.value = "<center>Invalid username or password</center>";
      }
}
```

It is important to note that the http.get is accessing your http service via a URL. Certain characters must be encoded in order to correctly pass through via an HTTP URL You may not always be able to control the values your users enters into your form fields. If the value may contain one of these characters that value must be encoded.

For example, this http.get contains a password parameter. Since user passwords may contain characters such as '#' you should encode the value. The built-in javascript method encodeURIComponent() makes this easy.

```
eval('x=' + http.get('http://<webhost>/MYservices/signForm?username=' +
                          u.value + '&password=' + encodeURIComponent(p.value)));
```

You may also need to decode the URL parameter in your HTTP service. For example:

```
import java.net.URLDecoder;
String p = request.getParameter("password");
p = URLDecoder.decode(password, "UTF-8");
```

10.1.2.2 Reusing Dynamic Content

Fetching dynamic content from your backend system can be the source of degrading form performance. If your form needs to use the dynamic content multiple times you can improve performance by fetching the content once from the http.get() and saving the returned JSON string in a hidden text form control. Then you can later read the text control value and eval it again rather than having to call your backend system multiple times.

For example add a hidden text control to your form named jsonUserData. Add a single line to the form rule that retrieves the user Data from your backend system via the http.get():

```
jsonUserDate.value = x;
```

In other form rules that also need the user data, rather than calling http.get() to your backend system again, add the following line to your form rule:

```
var val = jsonUserDate.value;
    eval('x' = val);
```

This will have the affect of setting x to the same string as if you had again fetched content from your backend system.

10.1.3 Using Data and Built-in Methods in a Form Rule

You can access built-in data and methods within your web form rules.

10.1.3.1 Built-in Data

Oracle BPM provides certain data to your form rules. This includes information about the person currently using your form, the tenant and form information. You retrieve this data in your form rule using the _data.getParameter('<data name>') syntax.

The following is a list of the available data:

- subject.id - logged in user's username.
- subject.first.name - Logged in users's First Name
- subject.last.name - Logged in users's Last Name
- subject.roles - A list of all the roles for the logged in user (available in v4.1.5)

You can use a form.load form rule to pre-populate fields in your form with information about the currently logged in user. For example, if you have controls in your form named Id, FirstName, LastName, Email and Roles.

Note: Setting the value of a control to an array or to a random JavaScript object is not allowed.

10.1.3.2 Built-in Methods

Oracle BPM provides built-in helper methods for common functionality required by form rules. Here is the list of available methods for working with dates and times:

- Time frevvo.currentTime(form) - returns the current time in the user's local time zone. This method should only be used to set the value of a Time control.
- Date frevvo.currentDate(form) - returns the current date in the user's local timezone. This method should only be used to set the value of a Date control.
- DateTime frevvo.currentDateTime(form) - returns the current date and time in the user's local timezone. This method should only be used to set the value of a Date/Time control.

Here is an example of setting a Time control named Tm, a Date control named Dt and a Date/Time control named DtTm.

```
Tm.value = frevvo.currentTime(form);
  Dt.value = frevvo.currentDate(form);
  DtTm.value = frevvo.currentDateTime(form);
```

The currentTime(), currentDate() and currentDateTime() will not work in a form.load form rule unless you specify a timezone on the form's Url via the _formTz Url parameter. This is because the form server needs to know the timezone in which to return the date and time. If you do not specify a _formTz the methods will return null and the control values will remain blank. For example, to specify Eastern time: &_*formTz=America/New*York.

Use the following methods to work with users and roles:

- boolean isUniqueUserId (String userId, String tenantId) - returns true if this user does not exist in the tenant and false if it does
- boolean isUniqueRoleId (String roleId, String tenantId) - returns true if this role does not exist in the tenant and false if it does

10.1.4 Understanding How Form Rules Work at Runtime

When using form rules within a web form, it is important to understand how form rules behave at runtime.

10.1.4.1 When are Form Rules Executed?

When you create or edit a form rule, Oracle BPM determines the list of controls and properties of those controls that the form rule depends upon. The form rule will be automatically executed whenever there is a state change that is relevant to the form rule. Form rules are also executed sequentially in a top to bottom order as they are seen in the form rules panel.

Note that form rules can trigger the execution of other form rules. So, if a form rule, R1, sets the value of a control with Name A, and there is a form rule R2, that depends on A.value, then form rule R2 will be triggered and executed.

A form rule will typically refer to one or more form controls and their properties and it will be executed if any of those properties change value. Note that form rules are not fired when the page is loaded. For example, the form rule below will only be executed when N1.value changes its value.

```
if (N1.value > 0 || N2.value > 0) {
    T.value = N1.value + N2.value;
}
```

Now let's assume a use case where you want to show a message if a user enters an invalid email. The form has a required email input control (Name=E) and an action should be executed if the email control 'valid' property is 'false'. One could write the form rule:

```
if (!E.valid) {
// code to show message here.
}
```

The code above would not work as expected. E is a required field and therefore E.valid initial value is 'false' since the field is in initially empty. When the user enters an invalid email address, E.valid would still have the value 'false' and the form rule would not execute since there is no state change. The code below would work properly.

```
if ((E.value.length > 0) && (!E.valid)) {
// code to show message here.
}
```

Now, the form rule depends on both the value of E.valid and E.value.length and therefore, when a string longer than zero characters is entered the form rule will be executed and the message will be shown.

10.1.4.2 Infinite Loops

It's easy to create form rules that enter a loop condition. For example, a form rule that updates A.value based on B.value and another form rule that updates B.value based on A.value. They could continually trigger each other.

The Oracle BPM server will prevent this from happening by setting an execution time limit for each form rule. Any form rule that takes longer than 5 seconds to execute will be forcibly stopped. Note that this is a very lengthy period of time for most computational tasks and the vast majority of form rules will not be impacted by this constraint. However, since Oracle BPM is a hosted site that may be simultaneously used by numerous users, there is a time limit imposed on these computations.

10.1.5 Debugging Form Rules

This section describes ways of debugging forms.

10.1.5.1 Debugging Duplicate Control Names

When you're designing forms, the forms designer generally prevents you from giving controls duplicate names, but you can give controls the same name if the controls are contained within different section controls. For example, you can have two sections in a form named Home'' and Office, and each section can have a text control name Address. However, if you want to use either of the Address controls in a form rule, you have to give them unique names.

One way to tell if a form rule is breaking because of duplicate names is to look at the log in the Tomcat console. If you see a message similar to the one below, it's probably a duplicate control name problem.

16:50:35,390 WARN RuleObserver:455 - Error evaluating rule [Translate.Translate Form Rule]: Java arrays have no public instance fields or methods named "value." ([Translate.Translate Form Rule]#2) if (form.load) { Name.value = 'Nancy'; }

The phrase *Java arrays have no public instance fields or methods...* means Java is interpreting the controls with the same name as an array, not single controls.

10.1.5.2 Form Rule Profiling

Form rule execution can be profiled to determine how much time each form rule takes to execute. This can help you tune and improve performance in forms that use a lot of form rules. To turn on profiling set the rule-debug=profile on the url.

If your form rules are executing database queries to populate dynamic select controls (dropdown, radio, checkbox), form rule profiling can help you identify if the db queries are taking too long. You will see HTTP calls taking a long time to execute.

10.2 Working with Form Rules

The following procedures describe how to create and test form rules.

10.2.1 How to Create a Form Rule

Business Process Composer provides an editor that enables you to create and edit a form rule.

To create a new form rule:

1. Open the form where you want to create a new form rule.
2. In the form toolbar, click the **Rules** button.
3. Click **Create New Rule**.
4. Click **Edit**, then edit the following fields as required:

Element	Description
Name	Defines the name of the form rule. You should change this to something meaningful.
Enabled	Select to enable the form rule within your form. If you deselect this option, the form rule will not be applied to the form at runtime.
Description	Provides a description of the form rule. You should modify the default to describe the behavior of the form rule and how it functions within the context of the form controls and other form rules within the form.
Rule	Defines the form rule.

5. After you have finished creating and editing the form rules, click **Edit** in the toolbar to return to the web forms designer.

10.2.2 How to Test a Form Rule

You can test the behavior of form rules while testing a web form. See Section 9.7.7, "How to Test a Web Form" for more information.

11 Working with Human Tasks

This chapter describes how to use human tasks to define how end users interact with you process-based application. It describes how to create and edit human tasks using Oracle Business Process Composer.

This chapter includes the following sections:

- Section 11.1, "Introduction to Human Tasks"
- Section 11.2, "Introduction to the Human Task Editor"
- Section 11.3, "Working with Human Tasks"
- Section 11.4, "Assigning a Human Task to a User Task"

11.1 Introduction to Human Tasks

Human tasks are a component of Oracle Human Workflow. Oracle Human Workflow is a component of the Oracle SOA Suite and Oracle BPM Suites that provides a runtime environment for managing user interaction with a process-based application. Human tasks define the user interaction, including the user interface and workflow.

Using Business Process Composer you can create new human tasks and configure its basic properties. However, for more advanced configuration of human tasks, you must use Oracle BPM Studio. See "Getting Started with Human Workflow" in the *Oracle BPM Modeling and Implementation Guide* for more information.

Human tasks are defined as services within a BPM project and are stored in the business catalog.

Human tasks define the following:

- **Participants**: specify the users or groups of users that perform the work of the human task.
- **Approval patterns**: specify the order the work of the human task is performed.
- **Access privileges**: determine who has access to a task.
- **Payload**: defines the data structures used by the human task. The payload data is passed from the user task of a BPMN process to the human task.
- **Expiration**: defines the duration of a human task.
- **Presentation**: defines the user interface participants use to interact with a BPMN process. Presentations are defined by a form. Oracle BPM supports two types of forms: web forms and Oracle Application Development Framework (ADF) task forms.

See Chapter 9, "Working with Web Forms" for more information about creating and using web forms using Business Process Composer. See "Getting Started with Human Workflow" in the *Oracle BPM Modeling and Implementation Guide* for about creating ADF task forms using Oracle BPM Studio.

11.1.1 Introduction to Participant and Routing Types

Human tasks enable you to determine the order users perform different tasks within your application. This order is defined by the routing of the human task. Participants are the users or groups that are responsible for performing each task. You can use the human task editor specify the routing flow and participant for a human task.

11.1.1.1 Participant Types

Human tasks support the following patterns for common routing scenarios:

- Single approver

  This is the simple case where a participant maps to a user, group, or role. See Section 11.1.2, "Introduction to Participant Assignment" for more information.

  For example, a vacation request is assigned to a manager. The manager must act on the request task three days before the vacation starts. If the manager formally approves or rejects the request, the employee is notified with the decision. If the manager does not act on the task, the request is treated as rejected. Notification actions similar to the formal rejection are taken.

- Parallel

  This participant indicates that a set of people must work in parallel. This pattern is commonly used for voting.

  For example, multiple users in a hiring situation must vote to hire or reject an applicant. You specify the voting percentage that is needed for the outcome to take effect, such as a majority vote or a unanimous vote.

- FYI

  This participant also maps to a single user, group, or role, just as in single approver. However, this pattern indicates that the participant just receives a notification task and the business process does not wait for the participant's response. FYI participants cannot directly impact the outcome of a task, but in some cases can provide comments or add attachments.

  For example, a regional sales office is notified that a candidate for employment has been approved for hire by the regional manager and their candidacy is being passed onto the state wide manager for approval or rejection.

- Serial

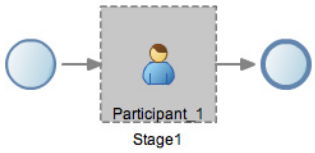
  This participant indicates that a set of users must work in sequence. While working in sequence can be specified in the routing policy by using multiple participants in sequence, this pattern is useful when the set of people is dynamic. The most common scenario for this is management chain escalation, which is done by specifying that the list is based on a management chain within the specification of this pattern.

When you create a human task, it contains a default simple participant. You can use the human task editor to change the participant type or add additional participants. See Section 11.3.6, "How to Change the Default Participant" for more information.

11.1.1.2 Routing Types

Figure 11–1 show a basic routing that contains only one participant.

Figure 11–1 Basic Routing with a Single Participant

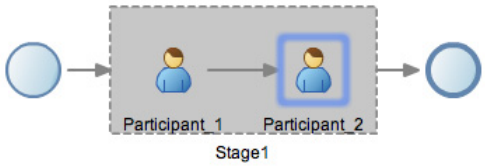


In addition to participant types, human tasks support more complex types of routing that give you more control. These routing types are:

- Sequential

  In sequential routing different participants act on a task sequentially. Figure 11–2 shows an example of sequential routing. In this example, Participant_1 is the first participant in the routing flow. After Participant_1 finishes acting on a task, the task is passed to Particpant_2. After Participant_2 finishes acting on the task, the human task is completed.

Figure 11–2 Example of Sequential Task Routing

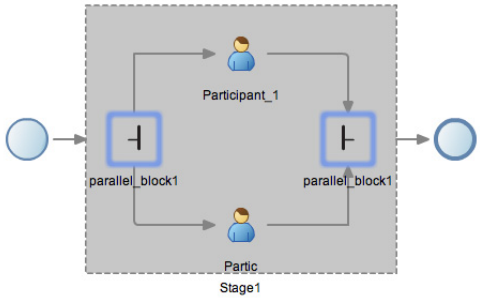


- Parallel

  In parallel routing, participants act on a task simultaneously. Figure 11–3 shows an example of parallel routing. In this example, both Participant_1 and Participant_2 act on the task at the same time.

  Parallel routing allows you to define an outcome for the routing that determines how the task flow continues to the next participant after the parallel block. See Section 11.1.1.3, "Outcome" for more information.

Figure 11–3 Example of Parallel Task Routing



11.1.1.3 Outcome

Outcome specifies possible outcome arguments of the Human Task. Oracle BPM Worklist and Process Workspace display the possible outcomes you select as the available tasks to perform at runtime.

You can specify a voted-upon outcome that overrides the default outcomes selected in the Default Outcomes list. This outcome takes effect if the required percentage is reached. Outcomes are evaluated in the order listed in the table as shown in.

You can define the following outcomes for a human task:

- Defer
- Yes
- Approve
- Accept
- Reject
- No

See Section 11.3.4, "How to Configure Basic Task Properties" for information on configuring the outcome for a human task. See Section 11.3.9, "*How to Configure the Outcome for Parallel Routing" for information about configuring outcomes in a parallel routing.

11.1.2 Introduction to Participant Assignment

Participant assignment is the process of mapping human task participants to the people in your organization that will use your application. This is done by mapping each participant to one of the following:

- Users

  You can assign individual users to act upon tasks. For example, you may assign users jlondon or jstein to a particular task. Users are defined in an identity store configured with the SOA Infrastructure. These users can be in the embedded LDAP of Oracle WebLogic Server, Oracle Internet Directory, or a third party LDAP directory.

  As with users, groups are defined in the identity store of the SOA Infrastructure.

- Groups

  You can assign groups to act upon tasks. Groups contain individual users who can claim and act upon a task. For example, users jcooper and fkafka may be members of the group LoanAgentGroup that you assign to act upon the task.

- Application Roles

  You can assign users who are members of application roles to claim and act upon tasks.

See Section 11.3.8, "How to Assign Users, Groups, and Roles to a Participant" for procedures on assigning users, groups, or roles to a human task participant.

11.1.3 Introduction to Duration

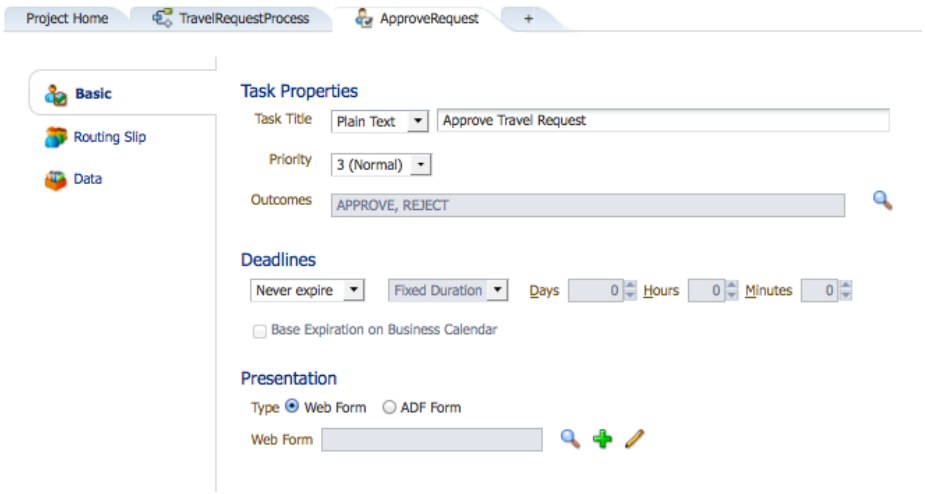
Duration defines how long a human task can remain idle before some other action is performed. Duration can be defined globally for the human task or for each human task participant.

See Section 11.3.5, "How to Configure the Deadline (Duration) for a Human Task" for information on defining duration globally for a human task.

11.2 Introduction to the Human Task Editor

Oracle Business Process Composer provides a human task editor that enables you to create and edit human tasks. The human task editor consists of a tabbed pane as shown in Figure 11–4.

Figure 11–4 The Human Task Editor



The human task editor contains the following tabs:

- **Basic**: enables you to configure the basic properties of a human task, define deadlines, and configure the presentation of the human task. See Section 11.3.4, "How to Configure Basic Task Properties" and Section 11.3.11, "How to Specify the Presentation of a Human Task" for more information.
- **Routing slip**: enables you to define and configure the participants and routing types of the human task. See Section 11.3.6, "How to Change the Default Participant" and Section 11.3.8, "How to Assign Users, Groups, and Roles to a Participant" for more information.
- **Data**: defines the data payload used by the human task. See Section 11.3.10, "How to Create and Configure the Data Payload for a Human Task" for more information.

11.3 Working with Human Tasks

The following sections describe how to create, configure, and edit human tasks.

11.3.1 Walkthrough: Creating and Configuring a Human Task

The following procedures describe the general steps required to create and configure a human task.

To create and configure a human task:

1. Create a new human task. You can create a new human task directly from the Project Welcome page. See Section 11.3.2, "How to Create New Human Task" for more information.
2. Open the human task. See Section 11.3.3, "How to Open a Human Task" for more information.
3. Configure the basic properties and deadline for the human task.
4. Define the payload data for a human task.
5. Assign a form to the human task.
6. Assign the human task to a user task.

11.3.2 How to Create New Human Task

Oracle Business Process Composer enables you to create new human tasks directly from the Project Welcome Page. Human tasks are defined as services and stored in the business catalog of a BPM project.

To create a new human task:

1. Open the project where you want to create a new human task.
2. Ensure the project is in edit mode.
3. From the Project Welcome page, select the **Human Tasks** tab.
4. Click **New Human Task**.
5. Provide a name for the new human task, then click **Create**.

The new human task is displayed in list of human tasks.

11.3.3 How to Open a Human Task

You can open an existing human task for viewing or editing.

To open an existing human task:

1. Open the project containing the human task you want to view or edit.
2. From the Project Welcome Page, select **Human Tasks**.
3. Click the name of the human task you want to open.

The human task is opened in the human task editor. See Section 11.2, "Introduction to the Human Task Editor" for more information.

11.3.4 How to Configure Basic Task Properties

Use the **Basic** tab of the Human Task editor to configure the basic properties of the human task, including the task title, task priority and outcomes.

To configure basic task properties:

1. Open the human task.
2. In the human task editor, select the **Basic** tab.
3. From the drop down list next to **Task Title**, select one of the following:

- **Plain text**: defines the task title using a text box. After selecting this option, enter the task title in the text box.
- **XPath**: defines the task title based on an XPath function.
- **Translation**: defines a translated version of the task title.

The task title defines the title of the human task that is displayed in Process Workspace.

4. From the drop down list next to **Priority**, select a priority for the human task. Valid values are between 1 (highest) and 5 (lowest). This value is displayed in Process Workspace.
5. Define the outcome for the human task:
   a. Click the magnifying glass icon next to the **Outcomes** field.
   b. Select the outcomes you want to configure. You must select at last one outcome. See Section 11.1.1.3, "Outcome" for more information.
   c. Click **OK**.
6. Save the project to save your changes.

11.3.5 How to Configure the Deadline (Duration) for a Human Task

Use the task editor to define the duration for

To configure the deadline (duration) for a human task:

1. Open the human task.
2. In the human task editor select the **Basic** tab.
3. From the drop down list under **Deadlines**, select one of the following:
   - **Never expire**: indicates that the human task has no deadline or expiration.
   - **Expire after**: indicates that the human task expires after a specified duration. When the option is selected, you can configure the following:
     - **Fixed duration**: defines the duration of the human task. When this option is selected, you can define a specific number of days, hours, and minutes.
     - **By expression**: enables you to use an expression to define the duration of the human task.
     - **Base expiration on Business Calender**: select this option to base the duration of the human task on a business calendar.
   - **Renew after**: extends that expiration date of a human task. When this option is selected, you can configure the following:
     - **Fixed duration**: defines the how long the expiration of the human task is extended. When this option is selected, you can define a specific number of days, hours, and minutes.
     - **By expression**: enables you to use an expression to define how long the expiration of the human task is extended.
     - **Maximum renewals**: defines the number of times the expiration deadline can be renewed.
     - **Base expiration on Business Calender**: if selected, this option bases the duration of the human task on a business calendar.

- **Escalate after**: escalates the task to a manager after the duration expires. When this option is selected, you can configure the following:
  - **Fixed duration**: defines the duration of the human task. When this option is selected, you can define a specific number of days, hours, and minutes.
  - **By expression**: enables you to use an expression to define the duration of the human task.
  - **Maximum escalation levels**: defines the number of levels the task can escalate after expiration.
  - **Highest approval title**: defines the title of the highest level in the management chain the escalation reaches.
  - **Base expiration on Business Calender**: if selected, this option bases the duration of the human task on a business calendar.

4. Save the project to save your changes.

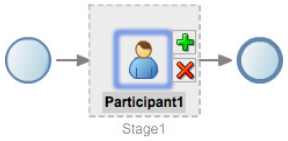
11.3.6 How to Change the Default Participant

When you create a new human task, it contains a default simple participant. Use the human task editor to change the type of the default participant.

To change the default participant of a human task:

1. If you are working on a shared project, ensure that you are in edit mode.
2. Open the human task, then select the **Routing Slip** tab.
3. Click the participant, then click the **Delete** (X) icon as shown in Figure 11–5 to delete the participant.

Figure 11–5 The Add and Remove Icons of a Participant



4. Click the routing slip box.
5. Click the **Add** (+) icon, then select a participant type. See Section 11.1.1, "Introduction to Participant and Routing Types" for information about each participant type.

The new participant appears in the routing slip box. See the following procedures for information about adding additional participants to a human task.

11.3.7 How to Add Participants and Routing to a Human Task

Use the human task editor to add participants or routing types to a human task.

To add participants to a human task:

1. If you are working on a shared project, ensure that you are in edit mode.
2. Open the human task, then select the **Routing Slip** tab.
3. Click on the routing slip block.

4. Click the + icon, then select a participant type. See Section 11.1.1, "Introduction to Participant and Routing Types" for information about each participant type.

The new participant is added to the routing slip in the far right position.

11.3.8 How to Assign Users, Groups, and Roles to a Participant

Within a human task, participants abstractly define the users who are responsible for performing the work of a human task and its work flow. You can define how the participants and routing defined in the human task are mapped to the real-world participants of your organization by configuring participant selection.

The following ways of selecting participants are available:

- Names and expressions
- Lane participants
- Parametric roles

> **Note:** Although you can configure a human task to use parametric roles, you cannot create or configure the parametric roles using Business Process Composer. These must be created and assigned to the human task using Process Workspace or Oracle BPM Studio.

Each of these is covered in the following procedures.

To select participants using names and expressions:

1. If you are editing a shared project, ensure that you are in edit mode.
2. Open the human task, then select **Routing Slip**.
3. Double click on the participant.
4. In the drop down box next to **Build a list of participants using:** select **Names and Expressions**.
5. Under **Participant Names**, click the **Add** icon, then select one of the following:
   - Add user:
   - Add group:
   - Add application role

   After selecting one of these options, it appears in the table. You can change the value, if necessary, using the drop down list in the table.
6. In the **Data Type** column, select either **By name** or **By Expression** from the drop down list.
7. Select a value for the participant:
   a. Click the **Search** icon in the **Value** column.
   b. From the drop down list select either **User**, **Groups**, or **Application Roles**.
   c. Enter the name you want to search, then click **Search**. To view a list of all names, groups, or roles, leave the text field empty, then click **Search**.
   d. Select the check boxes next to the users, groups, or roles you want to add.
   e. Click **OK**.

The name appears in the **Value** column.

To select participants using lane participants:

1. If you are editing a shared project, ensure that you are in edit mode.
2. Open the human task, then select **Routing Slip**.
3. Double click on the participant.
4. From the drop down box next to **Build a list of participants using,** select **Lane Participants**.
5. Select one of the following:
   - Previous lane participants
   - Current lane participants

To select participants using parametric roles:

1. Ensure that you have created parametric roles for the project using Oracle BPM Studio.
2. If you are editing a shared project, ensure that you are in edit mode.
3. Open the human task, then select **Routing Slip**.
4. Double click on the participant.
5. From the drop down box next to **Build a list of participants using,** select **Parametric roles**.
6. In the **Data Type** column, select either **By name** or **By Expression** from the drop down list.
7. Select a value for the participant:
   a. Click the **Search** icon in the **Value** column.
   b. From the drop down list select either **User**, **Groups**, or **Application Roles**.
   c. Enter the name you want to search, then click **Search**. To view a list of all names, groups, or roles, leave the text field empty, then click **Search**.
   d. Select the check boxes next to the users, groups, or roles you want to add.
   e. Click **OK**.

11.3.9 *How to Configure the Outcome for Parallel Routing

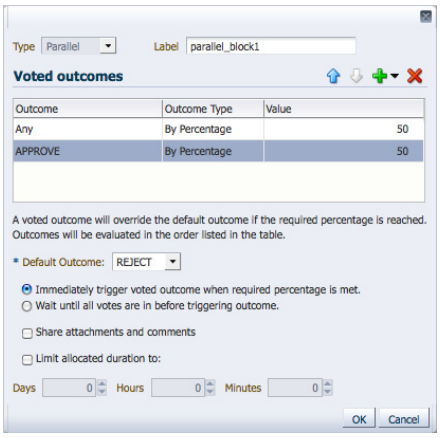
You can configure the routing outcome for parallel blocks.

To configure the outcome for parallel routing:

1. If you are editing a shared project, ensure that you are in edit mode.
2. Open the human task, then select **Routing Slip**.
3. Select the **General** tab.
4. In the **Participants** editor, double-click on one of the intersections of the parallel block.

   The Voted Outcomes editor appears as shown in Figure 11–6.

Figure 11–6 Voted Outcomes Editor



5. Click the Add icon, the select one of the following outcomes:
   - Approve
   - Reject
6. In the table, enter a value for the outcome.
7. Select a value for Default Outcome from the drop-down list.
8. Click **OK**.

11.3.10 How to Create and Configure the Data Payload for a Human Task

Using the human task editor, you can define the data types used within the human task. This data is used to store the information entered during user interaction.

To create task data for a human task:

1. If you are editing a shared project, ensure that you are in edit mode.
2. Open the human task.
3. Select the **Data** tab.
4. Click the Add button.
5. Select one of the following:
   - String
   - Int
   - Int(64)
   - Decimal
   - Real(32)
   - Real
   - Bool
   - Time Interval
   - TaskExecutionData

   The task data appears in Task Data table.

6. Enter a name for the new task data in the table

11.3.11 How to Specify the Presentation of a Human Task

The presentation of a human task defines the user interface of your application. Presentations are associated with a form which define the user interface, data structures and connectivity information for a human task. Oracle BPM supports two types of forms: web forms and ADF task forms. See Section 9.1, "Introduction to Forms in Oracle BPM" for more information about using forms.

To specify the presentation of a human task using an ADF task form:

1. Open the human task.
2. Select the **Basic** tab.
3. Under **Presentation**, select **ADF Form**.
4. Enter the following connectivity information for the ADF form:
   - **Hostname**: Specifies the hostname of the server where the presentation is deployed.
   - **HTTP Port**: Specifies the port of the server where the presentation is deployed.
   - **HTTPS Port**: Specifies the secure port of the server where the presentation is deployed.
   - **URI**: Specifies the Uniform Resource Indicator of the presentation.
5. Click **OK**.

To specify the presentation of a human task using a web form:

1. Open the human task.
2. Select the **Basic** tab.
3. Under **Presentation**, select **Web Form**.
4. Click the **Browse** button.

   The web form browser appears.
5. Click the name of the web form you want to select.

11.4 Assigning a Human Task to a User Task

After creating a human task, it becomes part of the business catalog of a BPM project. To incorporate a human task within a BPMN process, you must assign it to a user task within the process.

To assign a human task to a user task:

1. If you are editing a shared project, ensure that you are in edit mode.
2. Open your business process.
3. Add a user task to the process.
4. Right-click on the user task, then select **Implement**.
5. Click the **Browse** icon next to the **Human Task** text box.

   The human task browser appears.

6. Click the name of the human task you want to select.

   After clicking the name, the human task appears in the human task text box.

7. Click **Apply Changes** in the Implementation editor.

Part IV

Implementing and Deploying a BPM Project

This part describes advanced functionality of Business Process Composer that is targeted towards process developers who need to make changes to the implementation details of a BPM project.

This part contains the following chapters:

- Chapter 12, "Handling Data in Your Business Processes"
- Chapter 13, "Using Oracle Business Rules"
- Chapter 14, "Communicating with other Processes and Services"
- Chapter 15, "Deploying a BPM Project"

12

Handling Data in Your Business Processes

This chapter describes how to handle different types of data used within an Oracle BPM application. It provides information on using data objects and business objects to define the data structures used within your BPM project. It describes how to manipulate data using expressions and data associations. This chapter also provides information on how to define the data used by Process Analytics.

This chapter includes the following sections:

- Section 12.1, "About Handling Data Used by Your Business Processes"
- Section 12.2, "Introduction to Data Objects"
- Section 12.3, "Working with Data Objects"
- Section 12.4, "Introduction to Business Objects"
- Section 12.5, "Working with Business Objects"
- Section 12.6, "Introduction to Data Associations"
- Section 12.7, "Introduction to Expressions"
- Section 12.8, "Working with Expressions"
- Section 12.9, "Working with Business Indicators and Counter Marks"
- Section 12.10, "Measuring Process Performance Using Measurement Marks"

12.1 About Handling Data Used by Your Business Processes

Most business applications require users to create and manipulate data. In the Sales Quote example project, application users enter the data related to the quote which includes information about the customer, quote, and other types of data. Additionally, an application may need to create and manipulate other data that is only used internally as part of the overall function of the application.

When creating business processes you must define the data the application uses. In Oracle BPM, data is stored using data objects. Data objects can be defined based on simple types that are similar to those found in most programming languages.

Data objects can also be defined based on a business object. Business objects enable you to group together related data. For example, if you were creating an application that needed to store information about an employee, you may need to create a business object that stored the name, address, salary, and other information about the employee. Business objects are similar to the concept of classes used in object-oriented programming languages like Java.

12.1.1 How to Define the Data Used by an Oracle BPM Application

Defining how data is stored and manipulated is part of the overall design and development of an Oracle BPM application. The following high-level task outlines the typical process for defining the data used within an Oracle BPM application.

To Define the Data Used by Your Oracle BPM Application

1. Define the business objects required by your project.

   The first step in defining the data used within an Oracle BPM application is to define the required business objects. Business objects enable you to define the data structures used within your application. See Section 12.4, "Introduction to Business Objects" for more information.

2. Create the data objects used within your project
3. Define the expressions used to manipulate the data in your process.
4. Define how information is passed between BPMN flow objects using data associations.
5. Define the inputs and output for your process.

12.2 Introduction to Data Objects

Data objects are the variables used to store the information used by your business processes and Oracle BPM application in general. They are defined during the design and implementation stage of a process. Before deploying a BPM project you must define all of the data objects that the running application will need.

At runtime, new data objects are not created, but the value of the information they store is altered as users interact with your application. Running processes can store, access, and manipulate data. The values of data can also determine process branching.

12.2.1 Introduction to Basic and Complex Data Objects

Oracle BPM supports two types of data objects: basic and complex. Which type of data object you use depends on the type of data it needs to handle.

- Basic data objects

  Basic data objects are based on simple data types. These are the core data types common in most programming languages. Table x-x lists the basic data types supported by Oracle BPM.

Table 12–1 Simple Data Objects

Type	Description
Bool	Represents the logical values true or false.
Int	Represents an integer. For example: 23, -10, 0
Decimal	Represents a number than can be expressed in decimal notation. For example: 3.14, 62, 0.023.
Real	Represents a floating-point numeric value. For example: 2e-1, 2.3E8.
String	Represents a sequence of characters. For example: "This is a string."
Time	Represents a specific time expressed as: year-month day hour:minute:second. For example: 1995-02-03 13:30:28-08:00

Table 12–1 (Cont.) Simple Data Objects

Type	Description
Interval	Represents a duration of time expressed as a number in years, months, days, hours, minutes, and seconds. For example: 1d3h30m.
Binary	Used to store binary data, including images or videos.

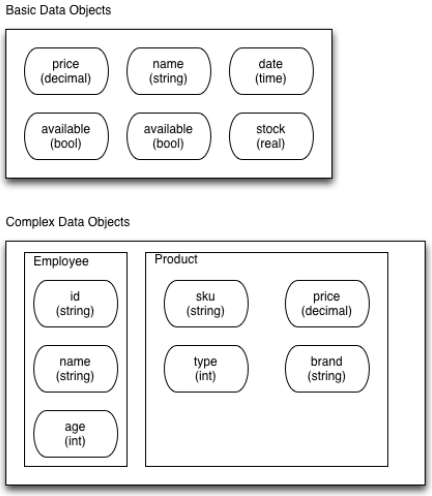
- Complex data objects

  Complex data objects enable you to create data structures that group together different types of data. Complex data objects are based on business objects. Business objects enable you to create data structures based on basic data objects.

  For example, you can create a complex data object called employee that contains different types of data for employee like id, name, and age. The relationship between business objects and complex data objects is analogous to the relationship between classes and instances in the Java programming language.

Figure 12–1 shows the relationship between basic and complex data objects.

Figure 12–1 Relationship Between Basic and Complex Data Objects



Before creating a complex data object, you must first define the business object that defines the data structure. See Section 12.4, "Introduction to Business Objects" for more information.

12.2.2 Introduction to Process and Project Data Objects

In addition to defining data objects based on type and complexity, you can also define data objects based on scope. In Oracle BPM, the scope of a data object refers to where in the process or project it can be accessed from.

In Oracle BPM, there are two types of data object scope: process and project. Process data objects are data objects that are defined for a specific process. Similarly, project data objects are defined for an entire BPM project. Both project and process data objects can be created from both basic and complex types.

12.2.2.1 Process Data Objects

Process data objects enable you to define data objects that are only used within a single process. Process data objects can only be used within the process where they are defined.

When designing a process-based application, if you know that a data object is only required within a single process, it is better to define it as a process data object in order to conserve system resources.

12.2.2.2 Project Data Objects

Project data objects allow you to share data between processes. For example, within an Oracle BPM application, both a purchase order process and an approval request process may track the data of the employee that created the request. The data created or modified in one process can be accessed by the other.

Project data objects can be used to ensure that all processes within an application have access to the same data. However, each process must assign and update the value of its data.

> **Note:** Although project data objects allow you to define data objects that are used by all processes in a project, they are not global data objects. Each process within your project uses its own version of the data object. Project data objects are not used to share data between processes.

Another benefit of defining project data objects is that after publishing your project you can configure Oracle Business Process Management Workspace views to display the values of the variables. This is only possible when using project data objects.

12.3 Working with Data Objects

This section describes the procedures for working with data objects, including how to create new data objects and edit or delete existing ones.

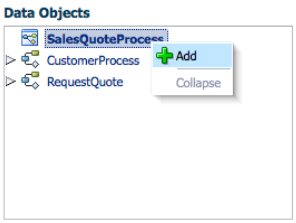
12.3.1 How to Create a Data Object

The procedures in this section describe how to create a new process or project data object. If you need to create a data object based on a business object, you must create the business object first. See Section 12.5, "Working with Business Objects" for more information.

To create a data object from the Project Welcome Page:

1. Ensure that you are editing your project.
2. Go to the **Project Welcome Page**.
3. If you are creating a process data object, ensure that you have created a process in the project.
4. In the **Data Objects** panel, right-click the name of the project or process, then select **Add** as shown in Figure 12–2.

Figure 12–2 Adding a New Data Object



5. Enter a unique name for your data object.
   - If you are creating a basic data object, select any of the basic types from the drop-down list.
   - If you are creating a complex data object, select **<component>** from the drop-down list, the select the business object the complex data object is based on.

   See Section 12.2.1, "Introduction to Basic and Complex Data Objects" for information on the data object types supported by Oracle BPM.
6. Click **OK**.

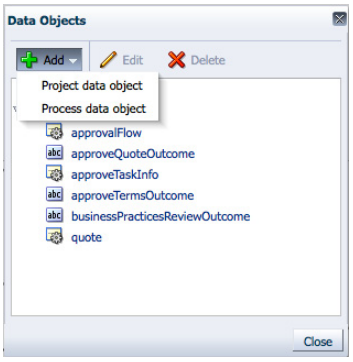
To create a data object from the process editor:

1. Ensure you are editing the project.
2. Open the business process where you want to create the data object.
3. Click the data objects icon in the process editor toolbar as shown in Figure 12–3.

Figure 12–3 The Process Editor Toolbar Showing the Data Object Icon

4. Click **Add**, then select either **Project data object** or **Process data object** as shown in Figure 12–4.

Figure 12–4 The Data Object Editor



5. Enter a unique name for your data object.
   - If you are creating a basic data object, select any of the basic types from the drop-down list.

- If you are creating a complex data object, select **<component>** from the drop-down list, the select the business object the complex data object is based on.

See Section 12.2.1, "Introduction to Basic and Complex Data Objects" for information on the data object types supported by Oracle BPM.

6. Click **OK**.

12.3.2 How to Edit or Delete a Data Object

You can edit or delete process or project data objects.

To edit a data object from the Project Home Page:

1. Open your project.
2. In the **Data Objects** pane of the **Project Welcome Page**, the name of the project or process whose data object you want to edit.
3. Right-click the name of the data object, then click **Edit**.
4. Change the name or type of the data object as required.
5. Click **OK**.

To edit or delete a data object from the process editor:

1. Ensure you are editing the project.
2. Open the process where you want to delete a data object. If you want to delete a project data object, you can open any process in the project.
3. Click the data objects icon in the process editor toolbar as shown in Figure 12–3.
4. In the list of data objects, select the project or process data object you wan to delete or edit, then:
   - To edit the data object, click **Edit**, then provide new name and change the data type as necessary.
   - To delete the data object, click **Delete**.
5. Click **OK**.

12.3.3 What Happens When You Delete or Edit a Data Object

After editing or deleting data objects, validate your project to verify that there are no references to changed or deleted data objects.

After editing a data object, you must ensure that all references to it are still valid. For example, if you change a data object from type int to type string, you must verify that all of the expressions that use the data objects will still function correctly.

After deleting a data object, you must ensure that all references to it are removed. This includes any data associations and expressions that use the data object. If you do not remove references to the deleted data object, the project does not validate.

12.4 Introduction to Business Objects

You can use business objects to create the data structures required in your Oracle BPM application. Business objects enable you to group related types of data together. This can help make your processes more manageable and readable to other users.

You can use Business Process Composer to create business objects manually or based on an XML schema.

The structure of a business object is composed of three components:

- Modules

  Modules are containers that enable your to create a hierarchical structure in you business object. Each business object must contain one top-level module. Within a module, you can create business objects or other modules.

- Business objects

  Within a module you can define one or more business object. Can contain other business object or modules.

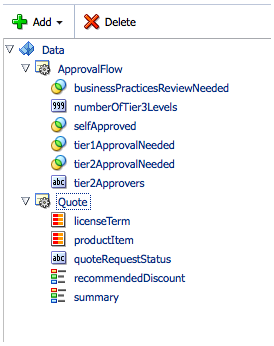
- Attributes

  Attributed are the lowest level component of a business object. Attributes define the data types of the business object.

You can also add documentation to your business objects and attributes. Adding documentation makes your data structures more understandable to other users who are collaborating on your BPM project.

The Sales Quote example process defines two different business objects to handle the required data. Figure 12–5 shows how these appear in the business objects editor.

Figure 12–5 The Business Objects Defined in the Sales Quote Example Project



12.5 Working with Business Objects

Using Business Process Composer you can create, edit, and delete business objects and their modules and attributes.

12.5.1 How to Create a Business Object Manually

You can manually create business objects from the business objects editor. When manually creating a business object, you can define the hierarchical relationship between modules and objects and the attributes used in the business object. A business object can include the following:

- Simple data types
- Arrays of data types
- Complex data types that use other business objects as a component

Note: After creating a module, business object, or attribute, you cannot change their names.

To create a new business object manually:

1. From the **Project Welcome Page**, click the **Business Objects** tab as shown in Figure 12–6.

Figure 12–6 The Business Objects Tab

2. Click **New BO**, then enter a name.
3. Select a parent module from the drop down list.

   You must select a parent module when creating a new business object. If no module currently exists, you must create a new one. To create a new module:

   a. Click the "**+**" icon.

   b. Enter a name for the new module, then click **OK**.
4. Click **Create**.

   The new business object appears in the **Business Objects** tab.
5. In the **Business Object** tab, click business object you just created.

   The business object editor opens.
6. If necessary, create a new module within the business object:

   a. Right click on the business object, then select **New Module**.

   b. Enter a name for the new module, then select **OK**.

      After creating a new module, you can create additional sub-modules if necessary.
7. If necessary, create a new business object:

   a. Right-click on a module, then select **Add New Business Object**.

   b. Enter a name, then click **OK**.

      The new business object appears within the list of business objects. You can create additional business objects or sub-modules within the business object if necessary.
8. Add a new attribute.

   a. Select a business object, then select **Add New Attribute** from the menu.

   b. Enter a name for the attribute and select a type from the drop down menu.
9. Click **OK**.

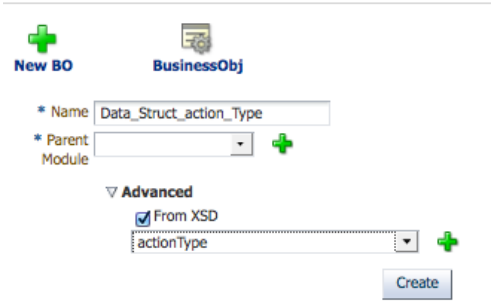
12.5.2 How to Create a Business Objects Based on an XML Schema Definition (XSD)

You can create a business object based on an XML Schema Definition. Like a business object, an XSD provides a hierarchical definition of a data structure.

To create a business objects based on an XSD:

1. From the **Project Welcome Page**, click the **Business Objects** tab, then click **New BO**.
2. Enter a name for the new business object.
3. Expand **Advanced**, then click **From XSD**.
4. Select an XSD from the drop down list as shown in Figure 12–7.

Figure 12–7 Creating a New Business Object from an XSD



5. Click **Create**

The new business object appears in the **Business Objects** tab. You can edit the new business object as required. See Section 12.5.4, "How to Edit and Delete a Business Object" for more information.

12.5.3 What Happens When You Create a Business Object

After you define a business object, it appears in the list of business objects in the business objects editor. You can use the business object to create new complex data objects that are based on it. You can also use the business object to define other business objects.

12.5.4 How to Edit and Delete a Business Object

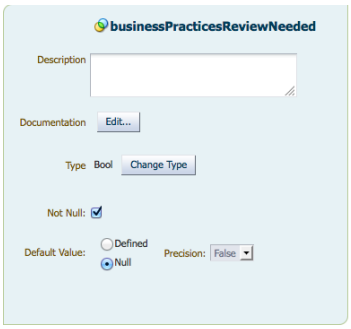
After creating a business object, you can change the type of its attributes or add documentation. You can also delete a business object or its modules or attributes.

To Change the attribute type of a business object or add documentation:

1. From the **Project Welcome Page**, click **Business Objects Editor** as shown in Figure 12–6.
2. Select the attribute you want to edit or add documentation to in the business objects tree.

   When you select the attribute, the attribute editor pane appears in the right-hand pane as shown in Figure 12–8.

Figure 12–8 The Attribute Editor Pane of the Business Object Editor



3. To change the data type of the attribute:
   a. Click **Change Type**, then select a new data type from the drop-down list.
   b. Click **OK**.
4. To add documentation:
   a. Click **Edit**. The documentation editor opens in the lower portion of the application.
   b. Add the required documentation in the documentation editor.

      See Section 5–5, "The Documentation Editor" for more information.
   c. Click **Apply Changes**.

To Delete a Business Object, Module, or Attribute:

1. From the Project Welcome Page, click **Business Objects** tab as shown in Figure 12–6.
2. Hover the cursor to the right of the business object you want to delete.
3. Click the **Delete** icon, then click **OK**.

> **Note:** When you delete a business object, module, or attribute it cannot be recovered. When deleting a module or business object that contains other components, the components are also be deleted. However, if the deleted business object or module contains an attribute based on a business object, that business object is not deleted.

12.6 Introduction to Data Associations

Data associations determine how information stored in data objects is handled in the following contexts:

- To and from another process or service invoked from a BPMN process
- To and from a Human Task service
- To and from an Oracle Business Rule
- To and from a script task. This BPMN flow object is used to pass data objects through data associations.

Table Table 12–2 lists the flow objects where you can define data associations. It also lists the objects implemented.

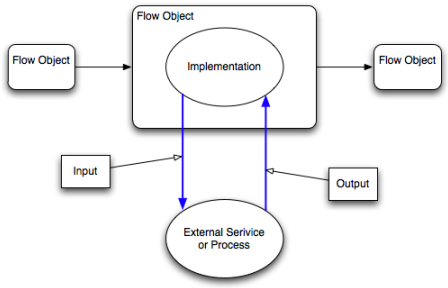
Table 12–2 Flow Objects that Accept Data Associations

Flow Objects	Implementation
Message start and end events	Services and other BPMN processes
Message throw and catch events	Services and other BPMN processes
Send and receive tasks	Services and other BPMN processes
Script tasks	Do not contain an implementation, are used to pass data objects through data associations.
User tasks	Oracle Human Tasks
Business rule tasks	Oracle Business Rules
Service Tasks	Services and BPMN processes

Data associations are used to define the input and output arguments from a flow object to an external service or process. Figure 12–9 shows the relationship between a flow object, its corresponding implementation, and external processes or services.

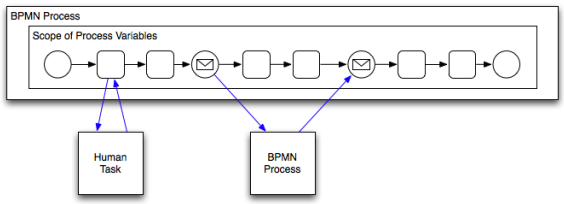
The blue arrows represent the input and output arguments to the external process or service. These are defined using data associations.

Figure 12–9 Relationship between a Flow Object, Implementation and 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, they define values that are passed to the systems or services being called. These systems and services exist outside of the business process as shown in Figure 12–11.

Figure 12–10 Data Associations between a process



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

12.6.1 Introduction to the Data Associations Editor

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

Figure 12–11 The Data Associations Editor

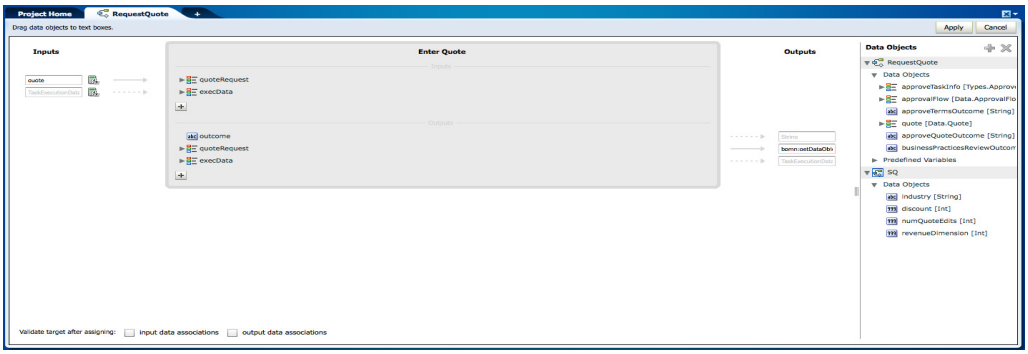


Table 12–3 describes the different areas of the data associations editor.

Table 12–3 The Data Associations Editor User Interface

UI Area	Description
Inputs	Contains text boxes that display the data objects assigned as inputs to the service or process implemented in the flow object. Next to each text box is an icon that launches the expression editor
Flow Object Interface	Lists the expected input arguments for the service or process implemented. The flow object interface also contains an expandable list of the data objects supplied as input and output. Within the flow object area, you can expand complex data objects to map to specific basic data objects within a complex data object.
Outputs	Contains text boxes that display the data objects assigned as outputs from the service or process implemented in the flow object.
Data Objects	Displays a list of all the data objects. This list is divided between process and project data objects.

12.6.2 How to Configure Data Associations for a Flow Object

You can configure data associatons for flow objects.

To configure a data association for a flow object:

1. Open the process where you want to configure data associations.
2. Right-click a flow object that enables data associations, then select **Data Associations**.

   See Table 12–2 for the list of sequence flows that allow data associations.
3. From the data objects column on the right, select the data object you want to map as an input argument.
4. Click and drag the data object to an input text field.

12.7 Introduction to Expressions

Expressions allow you to perform calculations on data objects.Using Business Process Composer, you can define and edit expressions in the following contexts:

- Conditional sequence flows
- Complex gateways
- Timer events
- Data associations
- Notification tasks

Expressions do not allow you to directly reassign the values to data objects. However, you can use expressions to change the values passed to and from the implementation of a sequence flow. See Section 12.6, "Introduction to Data Associations" for more information.

12.7.1 Introduction to the Expression Editor

The expression editor provides a simple way of creating expressions by allowing you to select data objects and operators from a list and insert them into your expression. You can also enter the expression manually if necessary.

Figure 12–12 shows the expression editor user interface.

Figure 12–12 The Oracle Business Process Composer Expression Editor

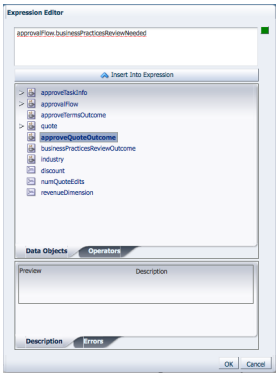


Table 12–4 The Expression Editor User Interface

Area	Description
Expression field	Contains the text of the expression. You can edit this field directly, or use the Insert Into Expression tool.
Insert Into Expression	Inserts the selected data object or operator into the expression.
Data Object and Operator Chooser	Contain tabbed panes that allow you to select the data object or operator you want to insert into the expression.
Description tab	Provides a description of the selected operator.
Errors	Displays errors in the current expression.

12.7.2 Types of Expressions

Oracle Business Process Composer supports the following types of expressions:

- Simple

- Plain text
- XML Literal

12.7.3 Simple Expressions

Simple expressions are defined using a basic expression language supported by Oracle BPM.

12.7.3.1 Operator Types

Simple expressions support the following operator types:

- Arithmetic Operators
- Unary Operators
- Equality and Relational Operators
- Conditional Operators

You can use these operators to write expressions and conditions to define your process flow. Generally these expressions perform their calculations based on the data objects in your process. You can write expressions and conditions using the value of the data objects, but you cannot modify their value.

The following examples of expressions use operators:

- totalAmount - discount
- activationCount > 3
- unitsSold <= 1200

Table 12–5, Table 12–6, Table 12–7, and Table 12–8 describe the supported operators in the simple expression builder.

Table 12–5 Arithmetic Operators

Operator	Name	Description
+	Addition	Adds numeric data types. Concatenates Strings.
-	Subtraction	Subtracts numeric data types.
*	Multiplication	Multiplies numeric data types.
/	Division	Divides numeric data types.
rem	Remainder	Calculates the remainder of a division in which the divisor does not exactly divide the dividend.
( )	Precedence	Indicates the order of evaluation of an arithmetic expression.

Table 12–6 Unary Operators

Operator	Name	Description
+	Plus	Has no effect in the value of the numeric operand. Use it to indicate explicitly that a certain value is positive.
-	Minus	Negates an arithmetic expression.
*	Not	Logical complement operator. Negates the value of a boolean expression.

Table 12–7 Equality and Relational Operators

Operator	Name	Description
= or ==	Equal	Returns true is the first operand equals the second operand.
!=	Not Equal	Returns true is the first operand is not equal to the second operand.
>	Greater Than	Returns true if the first operand is greater than the second operand.
>=	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 12–8 Conditional Operators

Operator	Name	Description
and	Conditional And	Returns true if both operands evaluate to true.
or	Conditional Or	Returns true if either operand evaluates to true.

12.7.3.2 Operator Precedence

Operator precedence indicates the order in which the compiler evaluates them. You can change operator precedence in an expression by using parenthesis.

In Oracle BPM the operator precedence is:

- Addition, Subtraction
- Multiplication, Division, Remainder
- Plus and Minus
- Less than, Greater Than, Less Than or Equal to, Greater Than or Equal to
- Equal, Not Equal
- Not
- Conditional And
- Conditional Or

12.8 Working with Expressions

The following sections describe how to define expressions using Business Process Composer.

12.8.1 How to Define a Simple Expression for a Conditional Sequence Flow

Using Business Process Composer, you can create and edit expressions for conditional sequence flows. Conditional sequence use expression to determine the flow of your process.

To define an expression for a conditional sequence flow:

1. Open your process.

2. Ensure that the project is in edit mode.
3. Click the edit icon for the conditional sequence flow you want to edit.
4. Click **Implementation**.
5. Click **Edit**.

   The expression editor window displays.
6. Add any required data objects and operators.

   To add a data object to an expression:

   a. Select the **Data Objects** tab.
   b. Select a data object from the list.

      If you add a basic data object that is part of a complex data objects, expand the complex data object and select the basic data object.
   c. Click **Insert Into Expression**

   To add an operator to an expression:

   a. Select the **Operators** tab.
   b. From the expandable list, select the operator you want to add.
   c. Click **Insert Into Expression**.
7. Click the **Error** tab, then verify that there are no errors in your expression.
8. Click **OK**.
9. Click **Apply Changes** in the **Implementation** tab.

12.8.2 How to Define a Simple Expression in Data Associations

Using Business Process Composer, you can create and edit expressions for Data associations. Data associations use expression to alter the values data objects passed as inputs. and outputs.

To define an expression within a data association input or output:

1. Open your process.
2. Ensure that the project is in edit mode.
3. Right-click a flow object within your process then select **Data Associations**.
4. Click **Launch Expression Builder**.
5. Add any required data objects and operators.

   To add a data object to an expression:

   a. Select the **Data Objects** tab.
   b. Select a data object from the list.

      If you add a basic data object that is part of a complex data objects, expand the complex data object and select the basic data object.
   c. Click **Insert Into Expression**

   To add an operator to an expression:

   a. Select the **Operators** tab.

b. From the expandable list, select the operator you want to add.

c. Click **Insert Into Expression**.

6. Click the **Error** tab, then verify that there are no errors in your expression.

7. Click **OK**.

12.9 Working with Business Indicators and Counter Marks

This section describes how to create business indicators and counter marks using Business Process Composer.

12.9.1 Introduction to Business Indicators and Counters

Business Indicators are project data objects you use to store the value of the key performance indicators of your process. Although Oracle BPM allows you to create business indicators using different types of data objects, within Business Process Composer you can only create business indicators used as counter marks.

Counters keep track of the number of times an instance completes a certain activity. You must use them with counter marks. The counter variable does not store the actual value, its value is always 1. The value that specifies the number of times an instance completes an activity is updated directly in the Process Analytics databases. To monitor the value of a counter business indicator, you must create a dashboard based on a counter mark that is configured to track this counter business indicator.

12.9.2 Introduction to Counter Marks

Counter marks enable you to update the value of the counter business indicators defined for your process. A counter mark may update multiple counter mark business indicators. When a token arrives at an activity that has a counter mark defined, the BPM Service Engine updates the value of its associated counters in the Process Analytics databases. Each time the BPM Service Engine updates a counter business indicator, it adds one unit to the current value.

Note: The actual value of the counter variable is stored in the Process Analytics databases. You must not use the counter variable in your process to perform any calculations because its default value never changes. The value of the counter variable is always equal to 1.

You can use counter marks for the following:

- Auditing: The number of activities the instance completed combined with other performance measurements are important information for auditing the process.
- Identifying performance issues: You can use a counter to identify performance issues within your process. Your process might be taking longer than expected because the instances are following a different path than expected or because the loop in an activity is running more times than it should. You can identify these situations by comparing the actual number of completed activities to the number you expected.
- Identifying the process path the instance followed: You can mark different paths using different counter business indicators. When the instance reaches the end of the process, the path the instance followed has the greatest number of completed activities.

Typically you define one counter business indicator for each of the process paths you want to monitor. Then you add counter marks in all the activities that are part of that process path. Finally you associate the counter business indicators that correspond to the paths that activity is part of, to the counter mark.

12.9.3 How to Add a New Counter Mark to a Process

You can add new counter marks to activities and tasks within your process.

To add a new counter mark to a process

1. Right-click on the task or activity where you want to add a counter mark.
2. Select **Implement**.
3. If required, create a new business indicator.
    1. Click the **Add** button.
    2. Provide a name for the business indicator.
    3. Click **OK**.
4. In the list of business indicators, select the check box next to the indicators you want to use for this flow object.
5. Click **Apply Changes**.

12.9.4 How to Delete a Counter Mark

You can delete the counter marks you have defined for a project.

To delete a counter mark:

1. Right-click on the activity or task where you want to edit a counter mark.
2. Select **Data Associations**.
3. In the list of Data Objects, expand the name of the project. This contains a list of all the project data object.
4. Select the counter mark you want to delete.
5. Click **Remove Data Object**.
6. Click **Yes**.
7. Click **Apply**.

12.10 Measuring Process Performance Using Measurement Marks

You can measure process performance using measurement marks. Measurement marks enable you to measure a business indicator of type measure at a certain point in the process or in a section of the process.

For more information on using measurement marks and the Process Analytics database, see "Using Process Analytics" in the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management*.

A measurement mark stores the following data into the Process Analytics databases:

- The value of the process default measures
- The value of the measure business indicators associated with that measurement mark

- The value of the dimensions defined in the process

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

When storing the value of a measure business indicator, the BPMN service engine also stores the value of the dimensions you defined in your process. Later on, when you build the dashboards to monitor your process, you can use these dimensions to group the values into different categories. For example, in the Sales Quote example you might want to view the total number of quotes approved by region.

The types of measurement marks you can define are:

- Single measurement
- Interval start
- Interval stop

12.10.1 How to Add a Measurement Mark to a Process

You can add measurement marks to your business processes by dragging the from the component palette to the process editor canvas.

To add a single measurement mark to a process:

1. Open the BPMN process.
2. In the **Component Palette**, double-click the **Measurement** icon, then click and drag one of the following.
   - Measurement mark (Snapshot)
   - Start measurement mark
   - End measurement mark
3. Place the measurement mark near the sequence flow where you want to add a business indicator. When the sequence flow turns blue, release the mouse measurement mark.
4. Right-click the measurement mark and select **Properties**, then click **Implementation**.
5. In the **Name** field, enter a name to identify the measurement mark.
6. In the Business Indicators section, select a business indicator from the list of available business indicators and move it to the Selected list using the arrows between the two lists.

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

   Note: If you do not select a business indicator, the measurement mark only stores the value of the default business indicators. If you want to add a business indicator without leaving the Measurement Mark Properties dialog, then click the **New** button under the **Selected** list.

7. Click **OK**.

13

Using Oracle Business Rules

This chapter describes how to use Business Process Composer to create and edit business rules. It contains a general introduction to Oracle Business Rules and provides tasks for working with them.

This chapter includes the following sections:

- Section 13.1, "Introduction to Oracle Business Rules"
- Section 13.2, "Introduction to the Business Process Composer Rules Editor"
- Section 13.3, "Creating and Editing Business Rules"
- Section 13.4, "Assigning a Rule to a Business Rules Task"
- Section 13.5, "Editing Oracle Business Rules at Run Time"

13.1 Introduction to Oracle Business Rules

This section provides a brief introduction to Oracle Business Rules.

Business rules are statements that describe business policies or describe key business decisions. For example, business rules include:

- Business policies such as spending policies and approval matrices.
- Constraints such as valid configurations or regulatory requirements.
- Computations such as discounts or premiums.
- Reasoning capabilities such as offers based on customer value.

For example, a car rental company might use the following business rule:

```
IF
Rental_application.driver age < 21
THEN
modify Rental_application(status: "Declined")
```

An airline might use a business rule such as the following:

```
IF
Frequent_Flyer.total_miles > 10000
THEN
modify Frequent_Flyer (status : "GOLD")
```

A financial institution could use a business rule such as:

```
IF
Application_loan.income < 10000
THEN
```

```
modify Application_loan (deny: true)
```

These examples represent individual business rules. In practice, you can use Oracle Business Rules to combine many business rules or to use more complex tests.

Oracle Business Rules allow process analysts to change policies that are expressed as business rules, with little or no assistance from a process developers. Applications using Oracle Business Rules support continuous change that enables the applications to adapt to new government regulations, improvements in internal company processes, or changes in relationships between customers and suppliers.

Business rules follow an if-then structure and consists of two parts:

- If part: a condition or pattern match.
- Then part: a list of actions.

Alternatively, you can express rules in a spreadsheet-like format called a decision table.

13.1.1 Introduction to Rule Conditions

The rule IF part is composed of conditional expressions, rule conditions, that refer to facts. For example:

```
IF Rental_application.driver age < 21
```

The conditional expression compares a business term (Rental_application.driver age) to the number 21 using a less than comparison.

The rule condition activates the rule whenever a combination of facts makes the conditional expression true. In some respects, the rule condition is like a query over the available facts in the Rules Engine, and for every row returned from the query the rule is activated.

13.1.2 Introduction to Rule Actions

The rule THEN part contains the actions that are run when the rule is fired. A rule is fired after it is activated and selected among the other rule activations using conflict resolution mechanisms such as priority. A rule might perform several kinds of actions. An action can add facts, modify facts, or remove facts. An action can run a Java method or perform a function which may modify the status of facts or create facts.

Rules fire sequentially, not in parallel. Note that rule actions often change the set of rule activations and thus change the next rule to fire.

13.1.3 Introduction to Decision Tables

A Decision Table is an alternative business rule format that is more compact and intuitive when many rules are needed to analyze many combinations of property values. You can use a Decision Table to create a set of rules that covers all combinations or where no two combinations conflict.

13.1.4 Introduction to Facts and Bucketsets

In Oracle Business Rules, facts are the objects that rules reason on. Each fact is an instance of a fact type. You must import or create one or more fact types before you can create rules.

In Oracle Business Rules a fact is an asserted instance of a class. The Oracle Business Rules runtime or a developer writing in the RL Language uses the RL Language assert function to add an instance of a fact to the Oracle Business Rules Engine.

In Rules Designer you can define a variety of fact types based on, XML Schema, Java classes, Oracle RL definitions, and ADF Business Components view objects. In the Oracle Business Rules runtime such fact type instances are called facts.

You can create bucketsets to define a list of values or a range of values of a specified type. After you create a bucketset you can associate the bucketset with a fact property of matching type. Oracle Business Rules uses the bucketsets that you define to specify constraints on the values associated with fact properties in rules or in Decision Tables. You can also use bucketsets to specify constraints for variable initial values and function return values or function argument values.

13.1.5 Introduction to Rulesets

A ruleset is an Oracle Business Rules container for rules and Decision Tables. A ruleset provides a namespace, similar to a Java package, for rules and Decision Tables. In addition you can use rulesets to partially order rule firing.

13.1.6 Introduction to Decision Functions

A decision function provides a contract for invoking rules from Java or SOA (from an SOA composite application or from a BPEL process). The contract includes input fact types, rulesets to run, and output fact types.

13.1.7 Introduction to Decision Points

Oracle Business Rules SDK (Rules SDK) provides APIs that let you write applications that access, create, modify, and run rules in Oracle Business Rules dictionaries (and all the contents of a dictionary). The Rules SDK provides the Decision Point API to access and run rules or Decision Tables from a Java application.

13.1.8 Introduction to Dictionaries

A dictionary is an Oracle Business Rules container for facts, functions, globals, bucketsets, links, decision functions, and rulesets. A dictionary is an XML file that stores the application's rulesets and the data model. Dictionaries can link to other dictionaries. Oracle JDeveloper creates an Oracle Business Rules dictionary in a.rules file. You can create as many dictionaries as you need. A dictionary may contain any number of rulesets.

13.2 Introduction to the Business Process Composer Rules Editor

The Business Process Composer rules editor enables you to view and edit a rules dictionary. Rules dictionaries are displayed in a tabbed window similar to the process editor and data association editor.

This window is divided into two main areas:

- A panel containing tabbed panes for globals, bucketsets, and rulesets.
- An editor panel showing detailed information for each tab.

Figure 13–1 shows the Globals tab displaying information for the globals contained in the Sales Quote example project.

Note: This tab only displays globals that were marked as final when the rules dictionary was created.

Figure 13–1 The Globals Tab of the Oracle Business Rules Editor

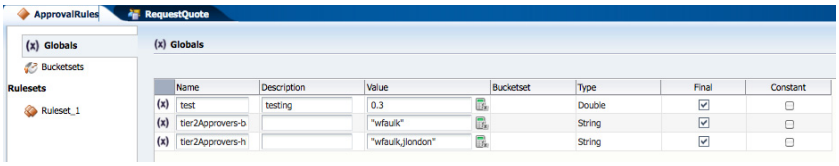


Figure 13–2 shows the Bucketsets tab displaying details for the bucketsets contained in the Sales Quote example project.

Figure 13–2 The Bucketsets Tab of the Oracle Business Rules Editor

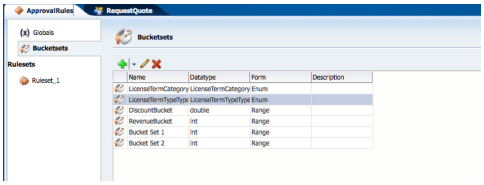
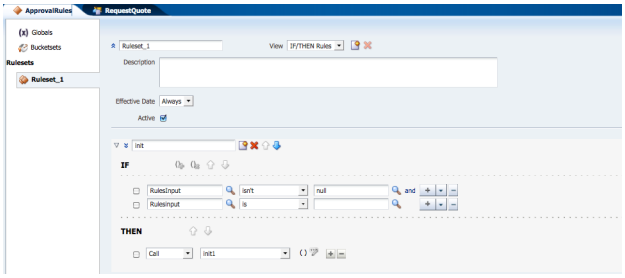


Figure 13–3 shows the **Rulesets** tab displaying details for the rulesets contained in the Sales Quote example.

Figure 13–3 The Rulesets Tab of the Oracle Business Rules Editor



13.3 Creating and Editing Business Rules

The following sections provide specific procedures for viewing and editing Oracle Business Rules using Business Process Composer.

13.3.1 How to Create a New Business Rule

Using Business Process Composer you can create new business rules.

To create a new business rule:

1. From the Project Welcome Page, select the **Business Rule** tab.
2. Click **New Business Rule**.
3. Enter a name and a package for the business rule.
4. Define the input and output data objects.

a. Click the **Add data object** button.

b. Select Input or Output from the drop down list on the right side of the window.

c. Enter a name for the data object.

d. From the drop down list, select a data type.

e. Click **Add**.

The data object appears in **Input and Output Data Objects** table. Add additional input or output objects as necessary.

5. Click **OK**.

13.3.2 How to Open a Business Rule

Oracle Business Rules can be included as part of the reusable business catalog, enabling you to use business rules when editing Oracle BPM projects created from project templates.

To open a business rule from the Project Navigator:

1. From the **Project Welcome Page**, expand **Rules**, then click the name of the rule you want to open.

The business rule appears in the rules editor. If you want to edit the rule, ensure that the project is in edit mode.

13.3.3 How to Add a Bucketset

Using Business Process Composer you can add new bucketsets to a rules dictionary.

To add a new bucketset:

1. Open the rules dictionary where you want to edit the bucketset.
2. Select the **Bucketsets** tab. This displays a table listing the bucketsets in the dictionary as shown in Figure 13–2.
3. Click the **Add Bucketset** drop-down list, then select the type of bucket set you want to create.
   - List of values
   - List of ranges
4. Select the bucketset from the list, then click **Edit Bucketset**.
5. Edit the bucket set as required, then click **OK**.

13.3.4 How to Edit an Existing Bucketset

In Business Process Composer, selecting the **Bucketsets** tab shows you a table listing the bucketsets in the dictionary. To edit a bucketset, select the appropriate row and click the **Edit** icon. Depending on the type of the bucketset, Range, Enum, or LOV, this displays a corresponding Edit bucketset page.

You can create a Range Bucketset by clicking Add in the menu bar and selecting a type. This adds a new row in the Bucketsets table. Adding a bucket automatically adds an end point for a range bucket and a value for an LOV bucket based on the data type. You can modify the newly added bucket end point or value. Note that the alias is modified when an end point or value is changed.

To delete a bucketset, select a row and click **Delete**.

To edit a bucketset:

1. Open the rules dictionary where you want to edit the bucketset.
2. Select the Bucketsets tab. This displays a table listing the bucketsets in the dictionary as shown in Figure 13–2.
3. Select the appropriate bucketset row and click the Edit Bucketset icon.
4. Use the Bucketset Editor to edit the appropriate fields in the bucketset.
5. Click OK to confirm the changes.

13.3.5 How to View Globals in the Oracle Rules Dictionary

When you open a rules dictionary, Business Process Composer displays the Globals tab. The Globals tab only shows final global variables (global variables with Final option selected).

You cannot create or delete global variables. From the Globals tab, in edit mode you can edit the Name, Description, and Value fields. For the Value field, you can use the expression builder to set the value.

To view globals:

1. Open the rules dictionary where you want to view globals.
2. Select the Globals tab. This displays a table listing the globals defined for this rules dictionary as shown in Figure 13–1.

13.3.6 How to Add a Rule to a Ruleset

Using Business Process Composer you can edit, add, and delete rules in a ruleset.

To add a rule to a ruleset:

1. Open the rules dictionary containing the ruleset where you want to add a rule.
2. Click the tab of the ruleset you want to edit. This displays a table listing the rulesets defined for this dictionary as shown in Figure 13–3
3. Click the **New Rule** icon.
4. Enter a name for the new rule.
5. Click the **Show Advanced Settings** icon.
6. In the IF area, use the controls, icons, and selection boxes, including the Left Value expression icon, drop-down list for an operator, and Right Value expression icon to modify the condition.
7. In the THEN area for the rule, next to the rule action click Add Action.
8. Click Save in the editor toolbar to save the changes to the rule dictionary.

13.4 Assigning a Rule to a Business Rules Task

The business rules task is an Oracle BPMN element that enables you to incorporate Oracle Business Rules within a process model.

When editing a project based on a project template containing business rules in the business catalog, you can assign business rules to business rules task.

To assign a business rule to a business rules task:

1. Open the process containing the business rules task you want to edit.
2. Right-click the business rules task, then select **Properties**.
3. Select the Implementation tab.
4. Click **Change**. The displays the business rules browser containing a table of available business rules.
5. Double-click a rule from the table.
6. Click **OK**.

13.5 Editing Oracle Business Rules at Run Time

You can use Business Process Composer to open deployed Oracle BPM projects. Opening a deployed project enables you to edit the Oracle Business Rules contained in the project and deploy your changes back to Oracle BPM runtime.

> **Note:** When editing a deployed project, you can only edit the Oracle Business Rules for that project. You can view other project resources, but cannot edit them.

To open a deployed project:

1. From the **Project** menu select **Open a Deployed Project**.

   If you are currently editing a project, your changes are automatically saved.
2. Select **Deployed Projects**, then select the project you want to open from the project list.
3. Expand Repository, then select the deployed project you want to open.
4. Click **Ok**.
5. In the Project Navigator, expand **Business Rules** then expand the rules dictionary where whose Oracle Business Rules you want to edit.
6. Click **Edit** in the rules editor.
7. **Edit** the rules as required, then click **Save**.
8. Click **Validate** to ensure the changes you made to the business s made are valid.
9. Click **Commit** to commit the changes to Oracle BPM runtime.
10. Click **Yes**.
11. From the **Project** menu, select **Close Project**.

14

Communicating with other Processes and Services

This chapter describes functionality of Business Process Composer related to technical implementation of a BPMN process. The functionality described in this chapter is generally performed by a process developer.

This chapter includes the following sections:

- Section 14.1, "Defining Process Input and Output"
- Section 14.2, "Defining Conversations"
- Section 14.3, "Working with Services"

14.1 Defining Process Input and Output

When you add operations to a BPMN process, you are defining points in the process that other processes or services can use to communicate with it. The communication between processes and other processes or services generally requires an input and returns an output. The flow events that you use you to define the BPMN process operations enable you to define input and output arguments. These input and output arguments define the process input and output.

14.1.1 How to Define the Input Arguments for a Process

When you create a process that begins with a message start event, you must define the input arguments to that are passed to that process.

To define the input arguments for a process:

1. Add a message start event to your process.
2. Right-click on the message start event, the select **Properties**.
3. Click the **Implementation** tab.
4. Select **Define Interface**.
5. Click the **Add** icon.
6. Determine the type of data object.
7. Click **OK**.

14.1.2 How to Define Data Associations for a Message Start Event

After you have defined the input arguments to your process, you must map them to data objects in your process.

To define data associations for a message start event:

1. Select the message start event of your process.
2. Click **Data Associations**.
3. Drag the data objects from the list on the right-hand side to the text boxes listed under **Outputs**.
4. Click **Apply**.

14.1.3 How to Define the Output Arguments for a Process

When you create a process that contains message end events, you must define the output arguments for each end event. These are the output arguments for the process.

To define the output arguments for a process:

1. Add a message end event to your process.
2. Right-click on the message end event, then select **Properties**.
3. Click the **Implementation** tab.
4. Select **Define Interface**.
5. Click the **Add** icon.
6. Determine the type of data object.
7. Click **OK**.

14.1.4 How to Define Data Association for a Message End Event

After you have defined the output arguments for your process, you must map them to the data objects in your process.

To define data associations for a message end event:

1. Select a message end event in your process.
2. Click **Data Associations**.
3. Drag the data objects from the list on the right-hand side to the text boxes listed under **Inputs**.
4. Click **Apply**.

14.2 Defining Conversations

The following sections describe how to edit conversations using Business Process Composer. For complete information see *Oracle BPM Modeling and Implementation Guide*.

14.2.1 Introduction to Conversations

Conversations group the message exchange between two or more processes. The message exchange between processes is called collaboration. Within a project you can define multiple conversations that you can reuse amongst the processes in that project.

Collaboration diagrams allows you to view the process flow together with the interactions your process has with other participants in the conversation.

Your BPM project defines a conversation by default. If you do not want to define multiple conversations you must use this default conversation to gather all the message exchange amongst the processes in your project.

You can only define one default conversation per project. However you can modify your project to use a different default conversation than the one it uses by default.

The different types of conversations allow you to specify the different types of interaction your process can establish with other processes or services. The following list describes the different types of conversations:

- **Define Interface**: use this type to define the operations that other services and processes can invoke to interact with a BPMN process.
- **Use Interface:** use this type to configure your process to use an interface from a component in the Business Catalog.
- **Process Call:** use this type to invoke another BPMN process.
- **Service Call**: use this type to invoke a service defined in your BPM project.

14.2.2 Working with Conversations

The following sections describe how to define and configure conversations using Business Process Composer.

14.2.2.1 How to define a conversation

1. Open your process.
2. Click the **Edit Conversation** button in the process editor toolbar.
3. Click the Add Conversation button.
4. Provide a name for the conversation.
5. Select the type of conversation you want to define.
6. Click **OK**.

14.2.2.2 How to set the default conversation

1. Open your process.
2. Click the **Edit Conversation** button in the process editor toolbar.
3. Select a conversation from the list, then click the **Edit** button.
4. Click the **Default Conversation** checkbox.
5. Click **OK**.

14.2.2.3 How to define a conversation for a BPMN flow object

1. Open your process
2. Right-click one of the following types of BPMN flow objects:

- Message events (throw and catch)
- Send and receive tasks
- Service tasks

You can only define conversations these BPM flow objects.

3. Select **Implement**.
4. In the **Implementation** tab, click the **Browse** button next to the **Conversation** text field.
5. Select a conversation from the list, then click **OK**.

14.2.2.4 How to view a collaboration diagram

To view the collaboration diagram for a process, click the **View Collaboration** button in the process editor toolbar.

Note: In collaboration view, you cannot make changes to the process. To return to normal process editing, click the **View Collaboration** button again.

14.3 Working with Services

The following sections describe how to create services using Business Process Composer.

14.3.1 How to Create New Services in the Business Catalog

Business Process Composer enables you to create new services in the business catalog. These services are based on standard Web Services.

Services based on web services are defined using a Web Services Description Language (WSDL) file. A WSDL file is an XML file used to describe how web services are implemented. When creating a new service using Business Process Composer, you can specify a WSDL file stored local on your machine or one that is available.

Process developers are responsible for providing a WSDL file or URL location.

To create a new service:

1. Open the project where you want to create a new service.
2. From the main menu, select **New** then **New Service**.
3. Provide the following information:
   - **Name**: Defines the name of the service as it appears in the business catalog.
   - **Type**: Defines the type of service being created.
   - **WSDL**: The source of the WSDL used to create the new service can be one of the following:
     - **URL**: Specifies the remote URL of the WSDL.
     - **File**: Specifies either a WSDL or ZIP file on your local file system.

       If using a ZIP file, it can include only WSDL and XSD files. The WSDL file should have valid references.

- **Port Type**: Port type specifies the service used. A WSDL file exposes one or more port types.
- **Callback Type**: Determines the call back type used for this web service. This is only applicable to asynchronous services.
- **Transaction Participation**:
- **Version**:

4. Click **OK**.

15

Deploying a BPM Project

This chapter describes how to deploy a BPM project from Oracle Business Process Composer. Users who have been granted permission can deploy projects directly to runtime.

This chapter contains the following sections:

- Chapter 15.1, "Configuring Approval Workflow for a Project"
- Chapter 15.2, "Deploying a Project"

15.1 Configuring Approval Workflow for a Project

The following sections describe how to configure the approval workflow for a project.

15.1.1 Introduction to Approval Workflow

Approval workflow defines the approval process that must be followed before an Oracle BPM project can be deployed to runtime using Business Process Composer. You can define the approval workflow when you create a new project, or you can configure it later. You can view the approval workflow defined for a project in the project properties box of the Project Welcome Page.

Table 15–1 describes the different types of approval workflow available in Oracle BPM.

Table 15–1 Approval Routing

None	No approval workflow is specified. Any user with the correct permissions can deploy the project directly to Oracle BPM runtime.
Simple	Specifies only one approver in the workflow.
Simple Sequential	Specifies a sequential list of approvers. All approvers must take action in order to approve deployment of the project.
Simple FYI	Sends notification that a project is being deployed, but approval workflow does not wait for approvers to process the task.
Parallel	Specifies a list of approvers who must approve the project deployment concurrently and in parallel.

15.1.2 Configuring Approval Workflow

The following sections describe how to configure approval workflow for a project and how to perform approvals as part of the workflow.

15.1.2.1 How to Configure Approval Workflow for a Project

Business Process Composer enables you to configure the type of approval workflow when you create a new project. You can also configure approval workflow after the project has been create.

To configure approval workflow for a project:

1. Go to the Project Welcome Page.
2. Ensure that you are editing the project.
3. Click **Edit** next to the approval workflow in the project information area.
4. Select the type of approval workflow, then click **Apply**.
5. Click **Choose**, then select **Users** or **Groups** from the drop-down menu in the browser.
6. Click **Search** to see a list of available users or groups.
7. Select an item from the **Available** list, then click **Move selected items to other list**.
8. Click **OK**.
9. Click **Apply**.

15.2 Deploying a Project

You can use Business Process Composer to deploy a project to the Oracle BPM runtime. Deployment is available only within the same environment where the Business Process Composer application is installed.

15.2.1 Who Can Deploy Projects?

Users who are granted project owner permissions can use Business Process Composer to deploy projects directly to Oracle BPM runtime.

15.2.2 How to Deploy a Project to Run Time

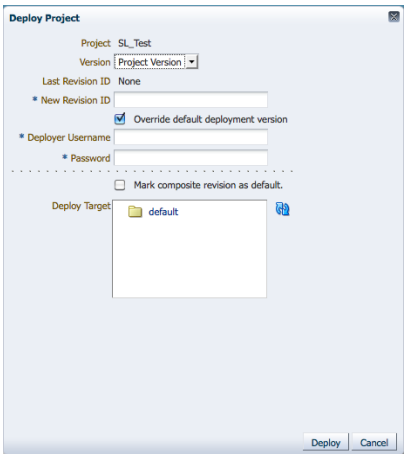
Project owners can deploy projects directly to Oracle BPM runtime. The following procedures define how to deploy a project when approval workflow is not enabled.

To deploy a project to runtime:

1. From the main menu select **Deployment**, then select **Deploy Project**.

   Business Process Composer validates the project. If there are no errors in the project, the deployment process continues.
2. Provide the required information in the **Deploy Project** dialog shown in Figure 15–1.

Figure 15–1 The Deploy Project Dialog



The properties in the Deploy Project dialog are described in the following table.

Table 15–2 The Deploy Project Dialog Properties

Project	Displays the name of the project
Version	Use to select the version of the project you want to deploy. This can be the current version of the project or a project snapshot.
Last Revision ID	Displays the revision ID of the most recent deployed version of the project.
New Revision ID	Specifies a revision ID for the deployed application. This ID can must be of the form: n0[.n1[.n2[.n3[.n4]]]][-milestone-name[milestone-number]/_ patch-number.
Override the default deployment version	Overrides the default deployment version.
Deployer Username	Used for deploying the Oracle BPM project to runtime.
Password	Specifies the password corresponding to the Deployer Username defined above.
Add MDS Connection for *oramds* Protocol	This option may be available, depending on how your server is configured. Select to enable a connection to a database using the *oramds* protocol. This is sometimes required when connecting to the database used for the Oracle BPM repository. If this is required, your system administrator must provide the following connectivity information for the database: ■ **Hostname**: specifies the database hostname or IP address. ■ **Port**: specifies the database port. ■ **SID**: specifies the SID of the database. ■ **Username**: specifies the database username used to connect to the Oracle MDS database. ■ **Password**: specifies the password for the database user.
Mark composite revision as default	Defines the current revision as the default revision.
Deploy target	Enables you to select the folder in Oracle MDS where the SOA composite application is deployed.

3. Click **Deploy**.

   Business Process Composer deploys the project to runtime. This may take a few minutes.

4. After the deployment is complete, click **OK**.

The project is deployed to Oracle BPM runtime. The project is available from the list of deployed projects in the project browser.

15.2.3 How to Deploy a Project Using an Approval Workflow

Business Process Composer enables you to specify an approval workflow. This workflow defines the users who must approve a project before the project is deployed to Oracle BPM runtime.

To deploy a project using approval workflow:

1. From the main menu select **Deploy Project**.

   Business Process Composer validates the project. If there are no errors in the project, the project appear in the Approval Workflow browser.

   Based on the type approval workflow defined for the project, required approvers must approve the deployment using Process Workspace. See the *Oracle Fusion Middleware Business Process Management User's Guide* for more information.

   You can monitor the approval status and progress using the Approval Workflow browser.

2. After the deployment has been approved open the Approval Workflow browser, then click **Deploy**.

15.2.4 How to Edit a Deployed Project

You can use Business Process Composer to open deployed Oracle BPM projects. Opening a deployed project enables you to edit the Oracle Business Rules contained in the project and deploy your changes back to Oracle BPM runtime.

Note: In order to edit a deployed project, you must be granted the SOA Designer role.

See Section 13.5, "Editing Oracle Business Rules at Run Time" for more information on editing Oracle Business Rules at runtime.

To open a deployed project:

1. Launch Business Process Composer.
2. From the main menu, select **Open a Deployed Project**.
3. Select a project from the Project navigator.
4. Click Refresh to ensure you see the latest contents of the Oracle BPM repository.
5. Click **OK**.

15.2.5 How to Generate a Project SAR File

You can generate a project as a SAR file from Business Process Composer. Your system administrator can use this file to deploy a project using the Oracle Enterprise Manager administration console.

To generate a project SAR file:

1. From the main menu, select **Deployment**, then select **Generate Project SAR file**.

   Business Process Composer validates the project. If the project contains errors, these are displayed in the project validation tab.

2. Select the project version you want to use to generate the SAR file. This can be the current version of the project or a project snapshot.
3. Enter a revision ID.
4. If required, select the following options:
   - Override the default deployment version
   - **Add MDS database connection for the oramds protocol**: Select to enable a connection to a database using the oramds protocol. This is sometimes required when connecting to the database used for the Oracle BPM repository.
5. Click **OK**.
6. If the project contains no errors, click OK to save the SAR file to your local file system.

15.2.6 How to Generate a Deployment Plan

A deployment plan is an XML configuration file that is used when deploying a BPM project from Oracle BPM Studio. Business Process Composer will generate any unexpected errors when generating the XML file of the deployment plan.

> **Note:** You should validate your project before creating a deployment plan. Business Process Composer does not perform any validation when generating the deployment plan.

To generate a deployment plan:

1. From the main menu, select **Deployment**.
2. Select **Generate Deployment Plan**
3. Select a location on your local file system, then click **OK**.

Part V

Performing Administrative Tasks Using Business Process Composer

This part describes how to manage projects within the Oracle BPM repository.

This part contains the following chapter:

- Chapter 16, "Performing Administrative Tasks"

16

Performing Administrative Tasks

This chapter describes how to perform administrative tasks using Oracle Business Process Composer. Administrators can assign global roles, manage projects and project templates, and enable the process player feature.

> **Note:** The procedures described in this chapter can only be performed by users who have been granted the Project Administrator security role.

This chapter includes the following sections:

- Section 16.1, "Introduction to Business Process Composer Administration"
- Section 16.2, "How to Assign Global Roles"
- Section 16.3, "Managing Projects and Project Templates"
- Section 16.4, "How to Define Administrator Credentials for Process Player"

16.1 Introduction to Business Process Composer Administration

Business Process Composer enables you to assign administrator privileges to a user. When a user has administrator privileges, the Administration option appears in the Business Process Composer toolbar as shown in Figure 16–1.

Figure 16–1 The Business Process Composer Toolbar Showing the Administration Menu Item

Administrators can perform the following tasks:

- Assign global roles to users.
- Delete shared projects and project templates.
- Release locks on shared projects.
- Configure permissions on projects.
- Import project templates.
- Configure administrator credentials for process player.

16.2 How to Assign Global Roles

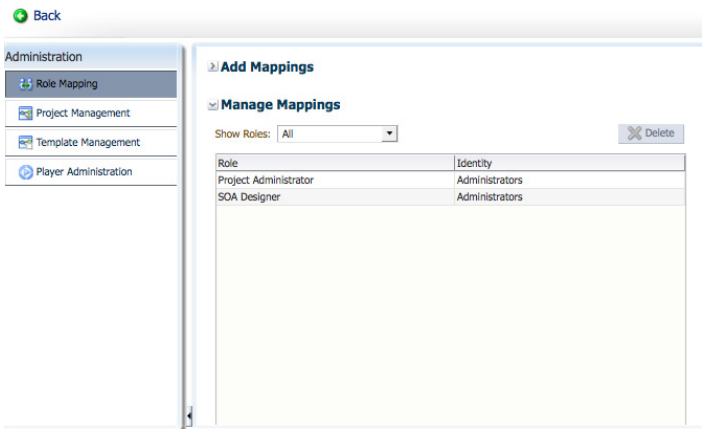
In addition to assigning permissions for individual projects, BPM Composer enables administrators to assign permissions globally. The following global roles are available:

- **SOA Designer**: The SOA Designer role is a security role shared with the SOA infrastructure. Business Process Composer uses this role to enable users and groups to edit Oracle Business Rules at runtime. See Section 8.4, "Editing Oracle Business Rules at Run Time" for more information.
- **Project Administrator:** Users or groups assigned this role can perform the following actions on all shared projects:
  - Remove projects
  - Deploy projects
  - Define project permissions
  - Create project snapshots

To assign a global role:

1. Login to Business Process Composer as a user with administrator privileges.
2. Click **Administration** in the Business Process Composer application toolbar as shown in Figure 16–1.
3. Select the **Role Mapping** tab. The role mapping editor appears as shown in Figure 16–2.

Figure 16–2 The Administration Console - Role Mapping Tab



4. Expand **Add Mappings**.
5. Select a role from the drop down list.
6. Click **Choose**:
   a. From the drop down list, select from the following:

      All: searches from both users and groups.

      Users: narrows searches to users.

      Groups: narrows searches to groups.

   b. Search of a user or group.

To search for a specific user or group, enter the name of a user or group, then click **Search**.

Click **Search** to view a list of all available users or groups

c. Click the check box next to the name of the users or groups you want to add.

d. Click **Ok**.

e. Click **Add Mapping**.

The new mapping appears in the table under **Manage Mappings**.

To remove a global role:

1. Login to Business Process Composer as a user with administrator privileges.
2. Click **Administration** in the Business Process Composer application toolbar as shown in Figure 16–1.
3. Expand **Manage Mappings**.
4. From the drop down list, select **All**.
5. In the role mapping table, select the mapping you want to delete, then click **Delete**.

16.3 Managing Projects and Project Templates

Use the Business Process Composer administration console to manage projects and project templates.

16.3.1 How to Delete a Project or Project Template

Administrators can delete shared projects or project templates.

Note: Administrators cannot perform tasks on private projects. Only the project owner has access to private projects.

To delete a shared project:

1. Login to Business Process Composer as a user with administrator privileges.
2. Click **Administration** in the Business Process Composer application toolbar as shown in Figure 16–1.
3. Select **Project Management** or **Template Management**.
4. Select a project or project template from the table.
5. Click **Delete**.
6. Click **Yes** to confirm that you want to delete the project or project template.

After deleting a project or project template, it is removed from the BPM repository.

16.3.2 How to Configure Sharing for a Project

Administrators can configure sharing for shared BPM projects.

To configure sharing on a project:

1. Login to Business Process Composer as a user with administrator privileges.

2. Click **Administration** in the Business Process Composer application toolbar as shown in Figure 16–1.
3. Select **Project Management**.
4. Select a project from the table.
5. Click **Share**:
   a. From the drop down list, select one of the following:

      Team members only: shares the project only with team members of the project owner.

      Public: shares the project with all Business Process Composer users.

   b. Click Choose, then specify a user or group.
   c. Select one of the following roles:

      Owner: enables full privileges on the project, including editing and deleting.

      Editor: enables the specified user or group to edit the project.

      Viewer: enables the specified user or group to view the project.
6. Click **Share**.

   The specified user or group appears in the table along with their role.
7. Click **Close** to return to the administration console.

16.3.3 How to Release the Lock on a Shared Project

Administrators can release the locks on shared projects that have been locked by another user.

To release the lock on a shared project:

1. Login to Business Process Composer as a user with administrator privileges.
2. Click **Administration** in the Business Process Composer application toolbar as shown in Figure 16–1.
3. Select the **Project Management** tab.
4. Select the project whose lock you want to release.
5. Click **Release Lock**.

Note: If you release the lock of a project, any changes made by the user who originally locked the project are lost.

16.3.4 How to Import a Project Template

Administrators can import a project template from their local file system.

To import a project template:

1. Login to Business Process Composer as a user with administrator privileges.
2. Click **Administration** in the Business Process Composer application toolbar as shown in Figure 16–1s.
3. Select **Template Management**.

4. Click **Import Template**.
5. Click Browse to locate the project template on your local file system.
6. Provide a name for the project template.
7. Click **OK**.

The imported project template is available to other Business Process Composer and Oracle BPM Studio users. See Section 8.3, "Working with Project Templates" for more information.

16.4 How to Define Administrator Credentials for Process Player

Process player enables you to test the behavior of a business process at design time using Business Process Composer. You can test a process as different users without having to deploy the BPM project and test the processes using Process Spaces. See Chapter 7, "Using Process Player" for more information.

When testing a business process, process player deploys a version of the BPM project to runtime using a special partition. This enables process player to run a process using the same environment as a normally deployed BPM project.

To enable the process player feature in Business Process Composer, you must first define the administrator credentials. Business Process Composer uses the administrator credentials to:

- Deploy a version of the project at runtime.
- Perform tasks on the process instance as different users.

After the administrator credentials are defined, any user who has edit privileges on a project can run process player.

When starting process player, Business Process Composer deploys the project to runtime using these credentials. When the current user wants to perform an activity or task as another user, Business Process Composer also uses these credentials.

16.4.1 How to Enable Process Player

Before Business Process Composer users can access process player, you must enable it by defining the administrator credentials.

To enable process player:

1. In the application toolbar, click **Administration**.
2. Select **Player Administration** from the menu at the left.
3. Enter the username, password, and password confirmation, then click **Save**.
4. Click **Back** to return to your BPM project.

16.4.2 What Happens When You Enable Process Player

After enabling process player, Business Process Composer users can us it to test the behavior of their business processes. See Section 7.2.1, "How to Access Process Player" for information.

Enabling process player makes it available to all Business Process Composer users. Any user who has edit privileges on a BPM project can run process player.

A

BPMN Flow Object Reference

This appendix provides a detailed description of each of the BPMN 2.0 flow objects that Oracle BPM supports.

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

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

- Section A.1, "Using Swimlanes to Organize Your Process"
- Section A.2, "Defining the Start and End Point of a Process"
- Section A.3, "Adding User Interaction to Your Process"
- Section A.4, "Communicating With Other Processes and Services"
- Section A.5, "Adding Business Logic Using Oracle Business Rules"
- Section A.6, "Controlling Process Flow Using Sequence Flows"
- Section A.7, "Controlling Process Flow Using Gateways"
- Section A.8, "Controlling Process Flow Using Intermediate Events"
- Section A.9, "Using Subprocesses in Oracle BPM"
- Section A.10, "Changing the Value of Data Objects in Your Process"

A.1 Using Swimlanes to Organize Your Process

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

A.1.1 Introduction to Roles

A key to designing a business process is determining the people and roles required to complete each of the tasks that require user interaction. Within your process, roles are used to model who is responsible for performing the work performed within your business processes. Roles enable you to define functional categories that represent job functions or responsibilities within your organization.

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

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

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

A.1.1.1 Roles in Context

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

The Sales Quote example defines the following roles:

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

See Section F.2, "Introduction to the Sales Quote Example" for more information on the Sales Quote example project.

A.1.2 Introduction to Swimlanes

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

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

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

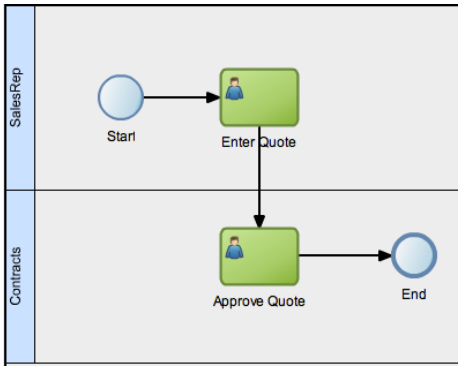
When you create a new process, Oracle BPM Studio and Business Process Composer create a default swimlane. You can add additional swimlanes to your process as necessary. When adding interactive and manual activities to a process, you must assign a role to the swimlane.

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

A.1.2.1 Swimlanes in Context

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

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



In a real-world business process, the combination of swimlanes and flow objects within them can be complex. The Sales Quote example project, as described in Appendix F, "The Sales Quote Example Process", is an example of a more complex process using multiple swimlanes.

A.1.3 How to Add Roles and Swimlanes to Your Process

You can add roles and swimlanes to your BPMN process.

To add a new swimlane to your process:

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

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

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

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

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

A.1.4 How to Edit Swimlane Properties

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

To edit swimlane properties:

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

To assign an existing role to a swimlane:

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

To assigning a new role to a swimlane:

1. Click the **Create** button.
2. .Enter the name of the new role in the text field.

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

A.1.5 Sharing Roles Between Business Process Composer and BPM Studio

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

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

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

A.2 Defining the Start and End Point of a Process

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

A.2.1 Introduction to Start and End Events

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

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

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

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

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

A.2.1.1 Specifying the Start Events for Different Types of Processes

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

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

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

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

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

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

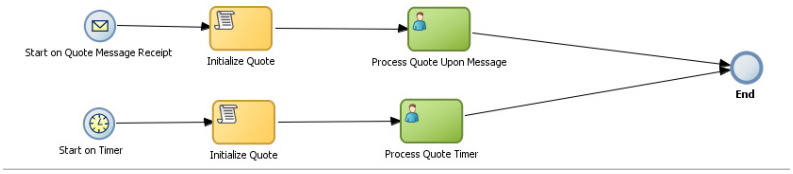
Event subprocesses contain a message start and none end event by default. However, you can change the start event to reflect the type of event you are handling.

A.2.1.2 Using Multiple Start Events in a Process

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

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

Figure A–2 Using Multiple Start Events within a Process



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

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

A.2.1.3 Using Multiple End Events in a Process

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

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

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

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

- Error end event: When an error end event is reached, all process activity is stopped. Like the error throw event, the error end event stops the flow of a process. See Section A.2.9, "Introduction to the Error End Event" for more information.
- Terminate end event: The terminate end event causes all work on a process to stop immediately. There is no error handling or other clean up of the running process. See Section A.2.11, "Introduction to the Terminate End Event" for more information.

A.2.2 Defining How a Process Instance is Triggered

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

- Using a message, signal, or timer start event. See the following sections for more information:
  - Section A.2.4, "Introduction to the Message Start Event"
  - Section A.2.5, "Introduction to the Signal Start Event"
  - Section A.2.6, "Introduction to the Timer Start Event"
- Using a none start event followed by a receive task. The receive task must be configured to create a process instance. See Section A.4.5, "Introduction to the Receive Task" for more information.
- Using a none start event followed by a user task defined with the initiator pattern. See Section A.3.2, "Introduction to the User Task" for more information.
- Using an event-based gateway that is configured to create a new process instance. See Section 4.5, "Using Guided Business Processes to Create Project Milestones" for more information.

A.2.3 Introduction to the None Start Event

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

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

Figure A–3 The None Start Event

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

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

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

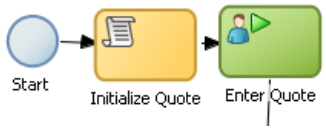
Similar to other start events, the none start event cannot have incoming sequence flows. It can only have default out-going sequence flows.

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

A.2.3.1 The None Start Event in Context

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

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



A.2.3.2 Data Associations

The none start event does not accept process input arguments.

A.2.4 Introduction to the Message Start Event

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

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

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

Figure A–5 The Message Start Event

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

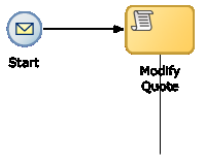
You can expose a BPMN process as a service that enables other processes and applications to invoke the process. To expose a process as a service, your process must begin with a message start event. Additionally, you must define the input arguments

to the process which are the data objects passed to the message start event. See Section 14.1, "Defining Process Input and Output" for more information.

A.2.4.1 The Message Start Event in Context

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

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



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

****

A.2.4.2 Using Process Input and Output Arguments

The message start event enables you to specify input and output arguments to a process. These arguments define the message that other processes or services must send to the process during invocation. See Section 14.1, "Defining Process Input and Output" for information on how to configure process input and output arguments.

A.2.5 Introduction to the Signal Start Event

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

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

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

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

Figure A–7 The Signal Start Event

A.2.5.1 The Signal Start Event in Context

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

A.2.6 Introduction to the Timer Start Event

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

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

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

Figure A–8 The Timer Start Event

A.2.7 Introduction to the Error Start Event

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

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

Figure A–9 The Error Start Event

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

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

A.2.8 Introduction to the None End Event

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

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

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

Figure A–10 The None End Event

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

A.2.8.1 The None End Event in Context

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

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

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

A.2.9 Introduction to the Error End Event

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

Errors end events are typically used with the error boundary event. The error boundary event is used to change the process flow based on a specific error. This flow usually ends using an error end event. See Section A.8.3, "Introduction to the Error Catch Event" for more information on using the error intermediate event.

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

Figure A–12 The Error End Event

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

A.2.10 Introduction to the Message End Event

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

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

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

Figure A–13 The Message End Event

For information on how to configure process output arguments using message end events, see Section 14.1, "Defining Process Input and Output".

A.2.11 Introduction to the Terminate End Event

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

A.3 Adding User Interaction to Your Process

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

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

- Section A.3.1, "Introduction to Human Workflow"
- Section A.3.2, "Introduction to the User Task"
- Section A.3.3, "Introduction to the Manual Task"

A.3.1 Introduction to Human Workflow

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

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

- Manual interactions with processes, including assignment and routing of tasks to the correct users or groups
- Deadlines, escalations, notifications, and other features required for ensuring the timely performance of a task.
- Organization, filtering, prioritization, and other features required for process participants to productively perform their tasks.
- Reports, reassignments, load balancing, and other features required by supervisors and business owners to manage the performance of tasks.

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

A.3.1.1 Introduction to Human Tasks

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

tasks through a variety of clients, including the worklist application, e-mail, portals, or custom applications.

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

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

- Roles and assignments
- Deadlines and escalations
- Presentations

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

A.3.2 Introduction to the User Task

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

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

Figure A–14 The User Task

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

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

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

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

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

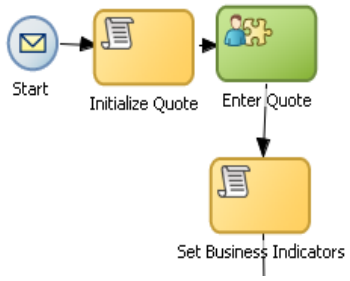
Similar to other flow objects, the user task may contain incoming and outgoing data associations.

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

A.3.2.1 The User Task in Context

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

Figure A–15 The Enter Quote User Task



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

A.3.2.2 Using Interactive Activities

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

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

Table A–2 Interactive Activities

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

A.3.2.3 Using the User Task in Project Templates

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

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

- Process developers must create any required human task services within Oracle BPM Studio before using the template in Business Process Composer. Human tasks can be implemented within a user task or this implementation can be performed when editing the project based on the project template.

- However, if the human task service is being implemented for a user task that is sealed within a project template, you must also perform the implementation before using the project template in Business Process Composer.

A.3.3 Introduction to the Manual Task

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

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

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

Figure A–16 The Manual Task

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

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

A.3.3.1 The Manual Task in Context

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

Figure A–17 Example of a Manual Task

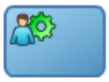
In this example, signing the formal contract is something that you may want to explicitly show as part of your business process. However, because it is not managed by the BPMN Service Engine, a manual task is used.

A.3.4 Introduction to the Update Task

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

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

Figure A–18 The Update Task



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

A.4 Communicating With Other Processes and Services

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

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

A.4.1 Introduction to the Service Task

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

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

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

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

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

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

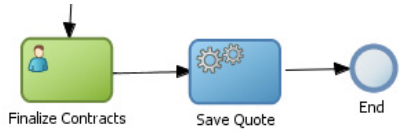
Figure A–19 shows the default notation for the service task.

Figure A–19 The Service Task

A.4.1.1 The Service Task in Context

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

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



A.4.1.2 Implementing Reusable Services in Project Templates

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

A.4.2 Introduction to the Notification Task

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

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

These different types of notification are:

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

Figure A–21 The Instant Message Notification Task

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

Figure A–22 The Email Notification Task

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

Figure A–23 The SMS Notification Task

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

Figure A–24 The Voicemail Notification Task

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

Figure A–25 The User Notification Task

A.4.3 Introduction to the Call Activity

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

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

Figure A–26 The Call Activity

See Section A.9.1, "Introduction to Reusable Processes (Reusable Subprocesses)" for more information.

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

A.4.4 Introduction to the Send Task

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

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

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

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

Figure A–27 The Send Task

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

A.4.4.1 The Send Task in Context

See Chapter A.4.6, "Using the Send and Receive Tasks to Communicate Between Processes" for information on using the send and received tasks to communicate between processes.

A.4.5 Introduction to the Receive Task

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

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

Figure A–28 The Receive Task

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

A.4.5.1 The Receive Task in Context

See Section A.4.6, "Using the Send and Receive Tasks to Communicate Between Processes" for information on using the send and receive tasks to communicate between processes.

A.4.5.2 Starting a Process with the Receive Task

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

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

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

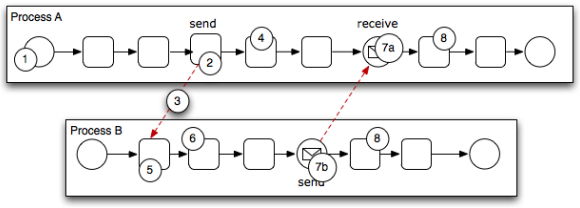
The following section describes how to use the send and receive tasks to communicate between processes.

A.4.6 Using the Send and Receive Tasks to Communicate Between Processes

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

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

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



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

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

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

   The receive task must have the Create Instance property defined. See Section A.4.5.2, "Starting a Process with the Receive Task.".
6. The newly created token proceeds through process B.
7. Depending on the specific behavior of your process, the following scenarios may occur:

   a. If the token of Process A reaches a receive task paired with a send task from Process B, the token of Process A waits until a response is received. After the response is received, the token of Process A continues to the next flow object.

   b. If the token of Process B reaches a send task paired with a receive task in Process A, Process B sends a response to Process A. The token of Process B continues to the next flow object.
8. Both processes continue running. You can use subsequent send and receive pairs to define subsequent communication between the two processes.

A.4.7 Introduction to the Message Throw Event

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

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

Figure A–30 The Message Throw Event

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

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

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

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

Message throw events are often used to invoke other BPMN processes by calling the message start event of another process. See Section A.2.4, "Introduction to the Message Start Event" for more information.

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

If your process receives a response synchronously, use the service task to invoke the process or service. See Section A.4.1, "Introduction to the Service Task" for more information.

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

A.4.8 Introduction to the Message Catch Event

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

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

Figure A–31 The Message Catch Event

The message catch event is frequently used with the message throw event to communicate with another BPMN process. See Section A.4.9, "Using Message Throw

and Catch Events to Communicate Between Processes" for information on how message throw events with message catch event.

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

A.4.9 Using Message Throw and Catch Events to Communicate Between Processes

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

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

  If you invoke a synchronous process use the service task. See Section A.4.1, "Introduction to the Service Task" for more information.

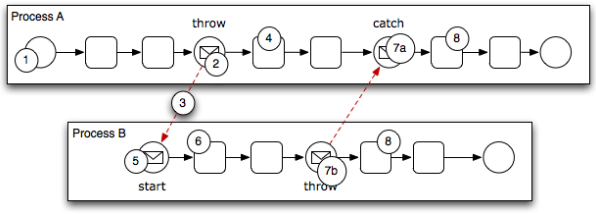
- The first time you use a message throw event it must be paired to the message start event of the other process. This is required to trigger the process instance. After the instance has been triggered, you can use subsequent message throw events that are caught by the second process.

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

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

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

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



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

1. Process A is invoked.
2. The token of Process A reaches a message throw event that is configured to invoke Process B.
3. The message throw event sends a message to the message start event of Process B.
4. The token of Process A proceeds to the next flow object.
5. The message start event triggers an instance of Process B.
6. The newly created token proceeds through Process B.

7. Depending on the behavior of your process, the following scenarios may occur:
   a. If the token of Process A reaches a catch event paired with a throw event from Process B, the token of Process A waits until the message is received. After the message is received, the token of Process A continues to the next flow object.
   b. If the token of Process B reaches a throw event paired with a catch event in Process A, Process B throws a message to Process A. The token of Process B continues to the next flow object.
8. Both processes continue running. You can use subsequent catch and throw event pairs to define subsequent communication between the two processes.

A.5 Adding Business Logic Using Oracle Business Rules

This section describes how to use the business rule task to incorporate Oracle Business Rules within your business processes. See Chapter 13, "Using Oracle Business Rules" for information on working with Oracle Business Rules using Business Process Composer.

A.5.1 Introduction to Oracle Business Rules

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

A.5.2 Introduction to the Business Rule Task

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

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

Figure A–33 The Business Rule Task

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

- Using structural rules

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

- Using operative rules

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

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

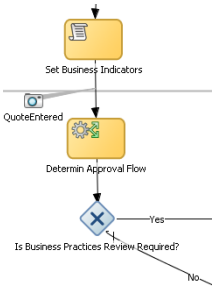
  See Section A.5.2.1, "The Business Rule Task in Context" for information on how an operation rule is used within the Sales Quote example.

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

A.5.2.1 The Business Rule Task in Context

Figure A–34 shows an example of the business rule task within the Sales Quote example.

Figure A–34 The Business Rule Task within the Sales Quote Example Process



A.6 Controlling Process Flow Using Sequence Flows

This section describes how to use sequence flows to define the behavior of your business process.

A.6.1 Introduction to Sequence Flows

Sequence flows define the order or sequence that work is performed within a process. Sequence flows connect the flow objects within your process and determine the path a process token follows through your process.

Incoming sequence flows are the sequence flows that lead into a flow object. Outgoing sequence flows are the sequence flows that determine the process path out of a flow object.

Most flow objects contain both incoming and outgoing sequence flows. Exceptions to this are start and end events. Start events can only contain outgoing sequence flows. End events can only contain incoming sequence flows. Additionally, event subprocesses do not have either incoming or outgoing sequence flows.

A.6.2 Introduction to Unconditional Sequence Flows

Unconditional sequence flows represent the typical path between two flow objects. Default sequence flows are displayed as a line with and arrow as shown in Figure A–35.

Figure A–35 The Unconditional Sequence Flow

Most flow objects can contain only one default out going sequence flow. Only parallel gateways can contain multiple unconditional sequence flows which represent the parallel paths of your process.

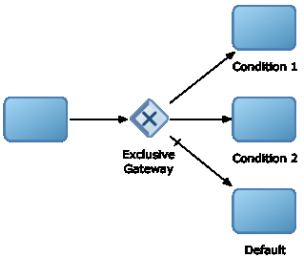
Exclusive, inclusive, and conditional gateways cannot have unconditional outgoing sequence flows. These gateways use conditional and default sequence flows to determine the flow of your process.

A.6.3 Introduction to Conditional Sequence Flows

Conditional sequence flows are used to control the flow of a process based on certain conditions. Like unconditional sequence flows, conditional sequence flows are displayed by an arrow lined arrow.

Figure A–36 shows two outgoing conditional sequence flows and a default sequence flow.

Figure A–36 Conditional and Default Sequence Flows



Not all flow objects can use outgoing conditional sequence flows. Only the following types of gateways can have outgoing conditional sequence flows:

- Exclusive gateways
- Inclusive gateway (split)

The conditions used within a conditional sequence flow are defined using expressions. See Section 12.7, "Introduction to Expressions" for information on using the expression editor to define expressions.

A.6.4 Introduction to Default Sequence Flows

Like conditional sequence flows, default sequence flows are used as outgoing sequence flows to exclusive, inclusive, and conditional gateways. Default sequence flows represent the path your process will take out of these gateways when none of the conditions evaluate to true.

Default sequence flows are represented by an arrowed line with a tic mark on one end as shown in Figure A–36.

A.7 Controlling Process Flow Using Gateways

This section describes how to use gateways to control process flow and behavior.

A.7.1 Introduction to Gateways

Gateways are flow elements that define the flow of your process. Gateways determine the path a token takes through a process. They define control points within your process by splitting and merging paths.

When possible, gateways are used for paths that are exceptions to or deviate from the default path of the process.

A.7.1.1 Split-Merge Pairs

The following gateways require a split-merge pair:

- Parallel gateway
- Inclusive gateway
- Complex gateway

When you add one of these gateways to a BPMN process, Oracle BPM Studio automatically creates the split and merge flow objects.

Although the merge portion of the gateway is required, you do not have to ensure that all paths out of the split return to the merge.

Although it is possible to have process paths that split at a gateway without merging through the gateway, this is not usually good practice. For more details on the merge behavior of gateways, see the following sections for each gateway type.

Note: If you delete the merge gateway from a process, the corresponding split gateway is also deleted.

A.7.2 Introduction to the Exclusive Gateway

The exclusive gateway enables you to split your process into two or more paths. However, the process only continues down one of these paths even if multiple outgoing sequence flows are present. Exclusive gateways can have conditional outgoing sequence flows and must have at least one default outgoing sequence flow.

You can define expressions that are used to determine if your process continues down a conditional sequence flow. See Section 12.7, "Introduction to Expressions" for more information.

If your process has multiple outgoing sequence flows for an exclusive gateway, you can define the order in which they are evaluated. The order of evaluation is configured in the properties of the exclusive gateway.

If you have an exclusive gateway where more than one conditional evaluates to true, the process will continue down the first conditional sequence flow determined by this order.

Unlike other gateways, the exclusive gateway does not require a corresponding merge to be explicitly defined in your process after splitting.

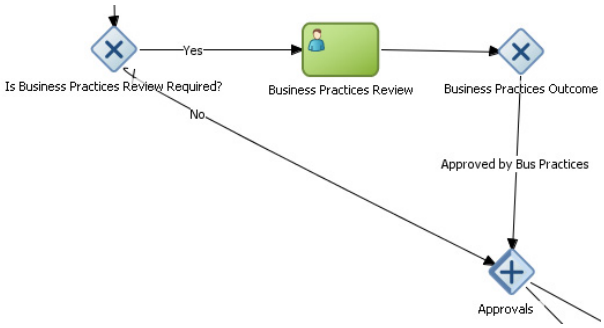
Figure A–37 shows the default notation for the exclusive gateway.

Figure A–37 The Exclusive Gateway

A.7.2.1 The Exclusive Gateway in Context

Figure A–38 shows an example of the exclusive gateway used within the Sale Quote example. Here, the exclusive gateway is used to evaluate whether a review of business practices is required.

Figure A–38 The Exclusive Gateway within the Sales Quote Example Process



This evaluation is determined by the expression defined for the outgoing conditional sequence flow. If this evaluates to true, then the process flow proceeds down the Yes path. If it evaluates to false, then the process flow proceeds down the path of the default outgoing sequence flow.

A.7.2.2 Splitting and Merging Exclusive Gateways

When a token reaches an exclusive gateway the outgoing conditional sequence flows are evaluated until one of them evaluates to true. You can define the specific order in which these are evaluated by configuring a property for the exclusive gateway.

Based on this configuration, when the first conditional sequence flow evaluates to true, the token moves down this outgoing sequence flow to the next flow object. If you have multiple outgoing conditional sequence flows, you can determine the order in which they are evaluated.

If none of the outgoing conditional sequence flows evaluate to true, then the token moves down the default outgoing sequence flow. Therefore, you must define a default outgoing sequence flow for the exclusive gateway.

The exclusive gateway can also merge incoming sequence flows. However, there is no synchronization with other tokens that may be coming from other paths within the process flow.

Note: If other tokens arrive at an exclusive gateway merge, then they are also passed through as is. If you are synchronizing tokens or perform evaluations on incoming sequence flows, you should use a different type of gateway.

A.7.3 Introduction to the Inclusive Gateway

The inclusive gateway enables you to split your process into two or more paths. Unlike the exclusive gateway, however, a token may flow down one or more of these paths depending on how the outgoing conditional sequence flows are evaluated.

You must define at least one default sequence flow for the inclusive gateway split. You can have multiple outgoing conditional sequence flows for an inclusive gateway split. The token will proceed along only those sequence flows that evaluate to true. If none of the sequence flows evaluate to true, then the token passes along the default sequence flow.

Figure A–39 shows the default notation for the inclusive gateway split.

Figure A–39 The Inclusive Gateway (Split)

Figure A–40 shows the default notation for the inclusive gateway merge.

Figure A–40 The Inclusive Gateway (Merge)

A.7.3.1 Splitting and Merging Inclusive Gateways

The inclusive gateway splits a process similar to the exclusive gateway, but enables tokens to proceed down multiple outgoing sequence flow. When a token arrives at an inclusive gateway, the expressions of its conditional sequence flows are evaluated.

Next, a token is generated for each of the conditional sequence flows that evaluates to true. A token is generated for the default sequence flow only if none of the conditional sequence flows evaluates to true.

These tokens are joined at the merge of the inclusive gateway. When a token reaches the merge gateway, it waits until all of the tokens generated by the split have reached the merge. After all of the tokens have reached the merge of the inclusive gateway, the merge is complete, and the token continues to the next sequence flow after the gateway.

A.7.4 Introduction to the Parallel Gateway

The parallel gateway enables you to split your process into two or more paths when you want your process flow to follow all paths simultaneously. The parallel gateway is useful when your process must perform multiple tasks in parallel.

Figure A–41 shows the default notation for the parallel gateway split.

Figure A–41 The Parallel Gateway (Split)

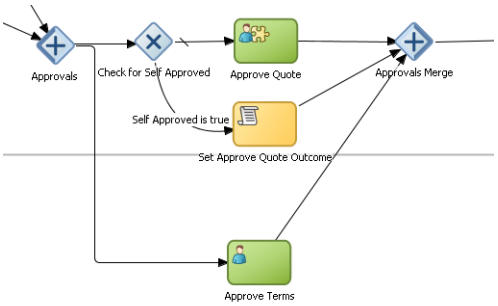
Figure A–42 shows the default notation for the parallel gateway merge.

Figure A–42 The Parallel Gateway (Merge)

A.7.4.1 The Parallel Gateway in Context

The Sales Quote example uses a parallel gateway during the approval stage of the process. Figure A–43 shows how the parallel gateway is used to perform two process paths simultaneously.

Figure A–43 Example of a Parallel Gateway



In this example, two different process paths are executed at the same time.

A.7.4.2 Splitting and Merging Parallel Gateways

When a token reaches a parallel gateway, the parallel gateway creates a token for each outgoing sequence flow. The split of the parallel gateway does not evaluate outgoing sequence flows.

You can also use the parallel gateway to merge process paths split by the parallel gateway. The merge of the parallel gateway waits for a token to arrive from each of the incoming sequence flows. After all tokens arrive, only one token is passed to the outgoing sequence flow.

Note: Design your process so that a token arrives for each incoming sequence flow for the merging parallel gateway. If you do not, your process can freeze if the merge is expecting tokens that do not arrive.

A.7.5 Introduction to the Complex Gateway

The complex gateway splits a process similar to an inclusive gateway. However, it enables you to define an activation condition that determines if the instance can continue even if not all of the tokens have arrived at the complex gateway merge.

For example, you can configure a complex gateway to continue after two or more tokens have arrived. If only two out of the possible conditions in the inclusive gateway evaluate to true the process instance continues to the next activity. However because the inclusive gateway immediately evaluates all the conditional sequence flows, all of the flow objects in these process paths are also run.

Figure A–44 shows the default notation for the complex gateway split.

Figure A–44 The Complex Gateway (Split)

Figure A–45 shows the default notation for the complex gateway merge.

Figure A–45 The Complex Gateway (Merge)

A.7.6 Introduction to the Event-based Gateway

The even-based gateway enables you to branch your process flow based on the possibility that an event may occur. Depending on the context, this may be one of several types of events.

The event-based gateway enables you to anticipate the possibility that several types of events may occur at a specific point in your process. It is similar to the exclusive gateway, but instead of choosing a path based on expressions, the event-based gateway chooses a path based on the occurrence of an event within your process.

For example, in an order processing process, you may reach a point in your process when there is no stock currently available. The process may need to wait until stock is available, but cannot wait indefinitely. By using an event-based gateway, your process can wait for a message saying new stock has been received (using a message catch event) or it can continue if no message is received after a certain amount of time has passed (using a timer event).

Figure A–46 shows the default notation for the event-based gateway.

Figure A–46 The Event-based Gateway

The event-based gateway is different from other gateways in that decisions about process flow are based on an event rather than data-specific conditions.

The event-based gateway is composed of the following:

- The event-based gateway
- Two or more target events. These can be of the following types:
  - Message catch events

    When initiating a process using a message catch event, the process must be invoked using a message throw event.
  - Timer catch events

    Generally only one timer event is used following an event-based gateway.
  - Receive tasks

    You can use the receive task to initiate a process instance following an event-based gateway. However the process must be invoked from a send task within the calling process.

    Note: You cannot mix message events and receive tasks within the same event-based gateway.

The target elements can only have incoming sequence flows from the event-based gateway. They cannot have sequence flows from other parts of the process.

Although the event-based gateway enables you to plan that multiple events may occur in your process, within the process instance, only one event is triggered. When the first event in the event-based gateway is triggered, the path that comes after that event is followed.

By default, when you add an event-based gateway to a process, it is created with a timer and message catch event.

Note: If you delete an event-based gateway, any outgoing sequence flows are also deleted. The associated events are not deleted.

A.7.6.1 Starting a Process with an Event-Based Gateway

You can also use an event-based gateway at the beginning of a process to create a new process instance. This is similar to having multiple start events within a process.

To enable an event-based gateway to create a new process instance, you must ensure the following:

- You have enabled the Initiate property of the event-based gateway.
- There are no incoming sequence flows to the event-based gateway.

Although the event-based gateway can be used to create a new process instance, it does not accept data input from another process. Any data that must be passed to the process instance must be configured using the target events.

A.8 Controlling Process Flow Using Intermediate Events

This section describes intermediate events and describes how to use them to control the flow and behavior of your process.

A.8.1 Introduction to Intermediate Events

Unlike start and stop events, intermediate events occur during the flow of your process.

There are two types of intermediate events:

- Normal flow events

  Normal flow events occur within the typical flow of your process.

- Boundary events

  Boundary events trigger an interruption with your process. Boundary events are associated with flow objects and can be configured to interrupt their usual behavior.

  Boundary events behave similar to sequence flows in that they are used to determine the path a process takes between flow objects.

  Boundary events can be divided into two types: interrupting and non interrupting.

A.8.2 Introduction to the Timer Catch Event

Timer catch events enable you to control the flow of your process using a time condition. Possible uses of the time catch event include:

- Creating a delay before running an activity
- Configuring a deadline for an activity
- Configuring a deadline for a process
- Triggering additional activities after an elapsed time

Figure A–47 shows the default notation for the timer catch event.

Figure A–47 The Timer Catch Event

You can use timer events as boundary events on an activity. Timer events can be defined as either interrupting or non interrupting boundary events.

When an interrupting timer event fires, the token leaves the main process flow to follow the flow the timer event defines. The flow that an interrupting timer event can return directly to the main process flow.

When a non interrupting event fires, a copy of the token is created and passes through the flow the timer event defines. The flow that a non interrupting event defines cannot return to the main process flow.

A.8.3 Introduction to the Error Catch Event

Error catch events are intermediate events used to handle an error that occurs within your process flow.

Note: You can also use inline handlers to handle error conditions that occur within your process.

Error catch events are always used as boundary events and can be attached to the following:

- Service Tasks
- Call Activities
- User Tasks
- Send Tasks
- Receive Tasks
- Script Tasks
- Rules Tasks
- Subprocesses

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

Figure A–48 shows the default notation for the error catch event attached as a boundary event on a service task.

Figure A–48 The Error Catch Event as a Boundary Event on a Service Task

When a service or process fails with an error, the error catch event is triggered. This causes the process flow to follow the path of the outgoing sequence flow of the error catch event.

You can use this flow to define how you handle the error. This is handled in two ways:

- The process flow returns to the main process flow. Any work that must be performed is handled within the error process flow before returning to the main flow.

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

- The process flow continues to an end event. The process is stopped immediately. Process control is returned to the service or process that initiated the process.

A.9 Using Subprocesses in Oracle BPM

You can use subprocesses to organize you business processes. Subprocesses enable you to group functional areas of your process together and also make your processes more readable.

Oracle BPM supports the following types of subprocesses:

- Reusable processes
- Embedded (also called inline) subprocesses
- Event subprocesses (also called inline handlers)

A.9.1 Introduction to Reusable Processes (Reusable Subprocesses)

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

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

Reusable processes have the following characteristics:

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

A.9.2 Introduction to Embedded Subprocesses (Inline Subprocesses)

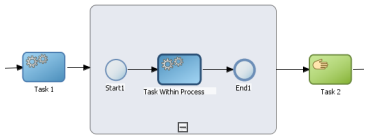
Embedded subprocesses enable you to group BPMN flow objects together to make your process more readable. In Oracle BPM, subprocesses are embedded subprocesses. Subprocesses are contained as part of the parent subprocess. Embedded subprocesses must begin with a start none event and must end with a none end event.

Embedded subprocesses can be expanded or collapsed. Figure A–49 shows how a collapsed subprocess appears within a process.

Figure A–49 Example of a Collapsed Embedded Subprocess

Figure A–50 shows how an expanded embedded subprocess appears within a process. When an embedded subprocess is expanded, you can edit the flow objects within it. You can also click and drag the edge of the embedded subprocess window to make the window larger or smaller.

Figure A–50 Example of an Expanded Embedded Subprocess



Similar to other types of processes, embedded subprocesses have can contain start and end events and contain a separate process flow. An embedded subprocess must begin with a none start event and end with a none end event. Embedded subprocesses cannot contain swimlanes.

Embedded subprocesses also behave similar to activities. They can have incoming and outgoing sequence flows. They also contain data associations that define the data objects used within the embedded subprocesses.

Embedded subprocesses can also contain timer, message, and error boundary events.

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

A.9.2.1 Embedded Subprocesses and Sequence Flows

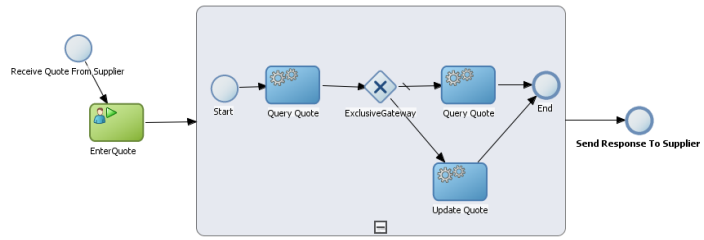
The flow objects within an embedded subprocess cannot have sequence flows that connect to flow objects outside the subprocess.

Similar to other flow objects, embedded subprocesses have incoming and outgoing sequence flows.

A.9.2.2 Embedded Subprocesses in Context

Figure A–51 show an example of a embedded subprocess. In this example, an embedded subprocess is used to group the service task used to process a sales quote.

Figure A–51 Example of an Embedded Subprocess



A.9.2.3 Looping Embedded Subprocesses

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

Note: Looping cannot be configured using Oracle Business Process Composer. You must used Oracle BPM Studio to configure the looping property of an embedded subprocess. See the *Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management* for more information.

A.9.3 Introduction to Event Subprocesses (Event Handlers)

Event subprocesses are a type of subprocesses that enable you to model possible conditions that happen outside of a normal process flow. Event subprocesses are also called event handlers.

Event subprocesses can have the following start events:

- Error start
- Timer start
- Message start

When you add an inline handler to your process, by default it is created with a message start event. You can change the type of start event if necessary.

Note: Inline handlers can only contain one start event. However, you can have multiple inline handlers within your process.

A.10 Changing the Value of Data Objects in Your Process

This section describes how to use the script task to change the values of data objects within your process. See Chapter 12, "Handling Data in Your Business Processes" for general information about data objects.

A.10.1 Introduction to the Script Task

The script task is used to change values of data objects within your process. The script task is used when you want to show this change explicitly within your business process or when you must change the values of data objects outside of another flow object. It is often used to set initial values of data objects at the beginning of a process.

Figure A–52 shows the default notation for the script task.

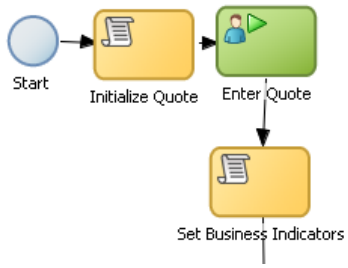
Figure A–52 The Script Task

Script tasks are added to a process by process developers who are responsible for defining the behavior of data objects within a process and process-based application.

A.10.1.1 The Script Task in Context

Figure A–53 shows two examples of the script task used at the beginning of the Sales Quote example. The Sales Quote example uses a script task to set initial values for data objects when the process instance is created and to set values for several business indicators.

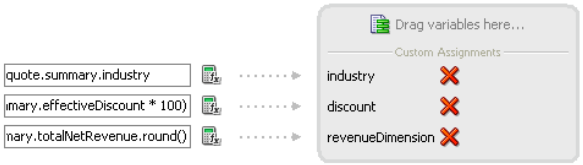
Figure A–53 The Script Task within the Sales Quote Example



Project data objects are data objects that you define in a project. All processes within a project have access to the data object defined, though the value changes according to the process using them. In addition, the engine stores the value of those marked as business indicators to the process analytics databases if the project is configured to use them.

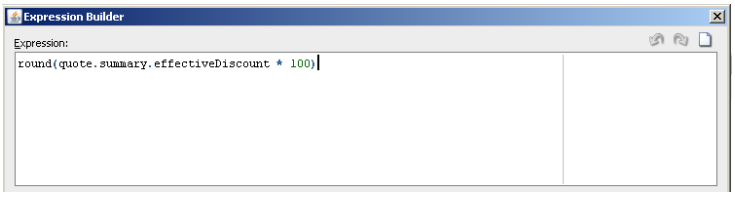
Figure A–54 shows the data associations used to set initial values for the business indicators.

Figure A–54 Data Associations Used by the Set Business Indicators Script Task



As with other flow objects that accept data associations, you can use expressions to change the values of data objects. Figure A–55 shows how an expression is used to change the value of the discount project variable.

Figure A–55 Expression Used to Change the Value of Discount Project Variable



Note: If you implemented the script task using transformations using BPM Studio, they will not be visible when opening the BPM project in Business Process Composer.

B

BPMN Flow Object Property Reference

This appendix provides information for each of the BPMN flow object properties. It contains these topics:

- Section B.1, "Common Properties"
- Section B.2, "Interactive Properties"
- Section B.3, "Activity Properties"
- Section B.4, "Gateway Properties"
- Section B.5, "Event Properties"
- Section B.6, "Measurement Mark Properties"
- Section B.7, "Sequence Flow Properties"

B.1 Common Properties

This section describes common properties shared by multiple BPMN flow objects.

B.1.1 Basic Properties

Table B–1 lists the properties shared by all activities and gateways. These properties appear in the properties popup.

Table B–1 Basic Properties for Activities and Gateways

Property	Description
Name	Defines the name of this flow object. This becomes the name of the flow object within your process.
Description	Provides an optional description of this flow object. Adding a description can make your process more readable.
Type:	Determines the type of flow object. When editing a project you can change the type of flow object by selecting from the drop-down menu.
Icon:	Displays the icon used for this flow object. You can click **Change** to select a different icon.

B.1.2 Implementation Properties

Table B–2 lists the implementation properties that are shared by multiple BPMN flow objects.

Table B–2 Implementation Properties

Property	Description
Is Draft	.When selected, specifies that the flow object is a draft.
Sampling Point	Use to configure sampling points for this flow object. ■ **Inherit Process Default**: Select to use the default sampling configuration defined at the process level. ■ **Generate**: Select to generate sampling point data for this activity. This will override the default configuration defined at the process or project level. ■ **Do Not Generate**: Select to not generate sampling point data. This is primarily used for performance reasons. Sampling points enable you to generate information about the performance of an flow object within in a running process. The data generated according to this configuration is stored in the Process Analytics Database. Sampling point generation specified at the project level is applied to all of the processes within the project. See Section 4.4.12, "How to View and Edit Project Properties" for procedures for setting sampling point for a project. However, you can override project-level settings within a process. Likewise, sampling point generation specified at the process level is applied to all of the flow objects within the process. You can also override process-level settings within each flow object. Overriding sampling point generation at the project or process level is usually done to improve performance. For example, if your project contains a process that contains a great number of activities and you are not interested in obtaining process metrics for this process, you might choose to set its sampling point configuration so that the process does not generate sampling points. Likewise, if you are interested in measuring only one process within your project, you might choose to set the project not to generate sampling point and configure that particular process to generate sampling points. By default, the project sampling configuration is set to generate sampling points only for interactive activities.

B.2 Interactive Properties

This section describes the properties of the interactive activities.

B.2.1 Interactive Activities

The interactive activities represent parts of your process where a process participant is required to perform work. See Section A.3.2, "Introduction to the User Task" for more information. This section lists properties for the following interactive activities:

- User Task
- FYI
- Management
- Group

- Complex
- Initiator

The properties popup of the interactive activities contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

Table B–3 describes the properties that can be edited from the **Implementation** editor.

Table B–3 Interactive Activities - Common Properties

Property	Description
Human Task	Defines the name of the human task assigned to this user task. You can select a list of human tasks in the business catalog.
Pattern	Displays the pattern used for this human task.
Re-initiate	Restarts the approval process from the beginning.

B.2.2 Manual Task

The manual task represents a task within a process that is performed by process participants that is outside of the scope of Oracle BPM.

See Section A.3.3, "Introduction to the Manual Task" for more information.

The properties popup of the manual task contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.3 Activity Properties

The following sections describe the properties of each BPMN activities supported by the Oracle BPM Suite.

B.3.1 Service Task

The service task enables you to communicate with other processes and services.

See Section A.4.1, "Introduction to the Service Task" for more information.

The service task contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.3.1.1 Implementation Properties

Table B–4 describes the properties that can be edited from the **Implementation** properties when you select the Service option.

Table B–4 Service Task Properties (When Process Call is Selected)

Property	Description
Conversation	Determines the type of conversation: Determines the type of conversation: ■ Default: enables you to configure the service conversation using only the process and target node. ■ Advanced: enables you to configure the conversation by defining a specific interface.
Name	Enables you to define the interface for the conversation. (This option is only available when Advanced is selected.)

Table B–4 (Cont.) Service Task Properties (When Process Call is Selected)

Property	Description
Process	Determines the BPMN process called by the service task. The process must be another process within the same BPM project.
Operation	Determines the specific node (flow object) within the BPMN process that is called by this service task.

Table B–5 Service Task Properties (When Service Call is Selected)

Property	Description
Conversation	Determines the type of conversation: ■ Default: enables you to configure the service conversation using only the process and target node. ■ Advanced: enables you to configure the conversation by defining a specific interface.
Name	Enables you to define the interface for the conversation. (This option is only available when Advanced is selected.)
Service	Determines the service called by this service task. This service must be defined in the business catalog of the BPM project.
Operation	Determines the operation called by the service task.

B.3.2 Send Task

The send task sends a message to a system or process outside the current process.

See Section A.4.4, "Introduction to the Send Task" for more information.

The send task contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.3.2.1 Implementation Properties

Table B–6 describes the implementation properties of the send task when **Define Interface** is selected.

Table B–6 Send Task Properties (When Define Interface is Selected)

Property	Description
Default	Defines the interface using only argument definitions that are passed from the service task to the service being called. You can configure the required arguments by clicking the **Add** button. Valid values for argument definitions are: ■ **Name**: defines the name of the argument. ■ **Type**: defines the data type of the argument.
Advanced	Enables you to select the operation name in addition to defining the argument definitions. ■ **Operation Name**:
Asynchronous	Indicates that the interface is called asynchronously.

Table B–6 (Cont.) Send Task Properties (When Define Interface is Selected)

Property	Description
Synchronous	Indicates that the interface is call synchronously. You can also define the following properties: ■ **Reply To**: ■ **Throw Error**: Indicates that the send task uses an error handler when a problem occurs. ■ **Error**: Determines the error called when a problem occurs. This error must be defined in the business catalog.

Table B–7 describes the implementation properties available when **Initiates** is selected.

Table B–7 Send Task Properties (When Use Interface is Selected)

Property	Description
Name	Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Reference	
Operation	Determines the operation invoked by the send task.
Error	Determines the error called when a problem occurs. This error must be defined in the business catalog.

Table B–8 describes the implementation properties when **Continues** is selected.

Table B–8 Send Task Properties (When Process Call is Selected)

Property	Description
Name	Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Process	Defines the BPMN process this send task invokes.
Target Node	Determines the flow object within the BPMN process that is called by the service task.

Table B–9 describes the implementation properties when **Continues** is selected.

Table B–9 Send Task Properties (When Service Call is Selected)

Property	Description
Name	Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Service	Determines the service this send task invokes
Operation	Determines the operation invoked by the send task.

B.3.3 Receive Task

The receive task waits for a message from a system or process outside the current process.

See Section A.4.5, "Introduction to the Receive Task" for more information.

The receive task contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.3.3.1 Implementation Properties

Table B–10 describes the properties of the receive task when Define Interface is selected.

Table B–10 Receive Task (When Define Interface is Selected)

Property	Description
Name	Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Arguments definition	Lists the arguments required to invoke the operation the receive task exposes. These are the arguments passed to the process from the invoking process or service.
Operation Name	.Defines the operation invoked by the received task.

Table B–11 describes the implementation properties when **Use Interface** is selected.

Table B–11 Receive Task Properties (When Use Interface is Selected)

Property	Description
Name	Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Reference	
Operation	Defines the operation invoked by the receive task.

Table B–12 describes the implementation properties when **Service Call** is selected.

Table B–12 Receive Task Properties (When Service Call is Selected)

Property	Description
Name	.Determines the name of the interface used. (This option is only available when **Advanced** is selected.)
Service	Displays the service this receive task invokes.
Operation	Determines the name of the operation this receive task invokes.

B.3.4 Business Rule Task

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

See Section A.5.2, "Introduction to the Business Rule Task" for more information.

The business rule task contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.3.4.1 Implementation Properties

Table B–13 describes the properties that can be edited from the **Implementation** editor.

Table B–13 Business Rule Task Properties

Property	Description
Rule	Determines the business rule assigned to this business rules task.
Operation	Specifies the decision function of the rule specified above.

B.3.5 Script Task

The script task is used to change values of data objects within your process

See Section A.10.1, "Introduction to the Script Task" for more information.

The script task contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of theseS properties.

B.3.6 Call Activity

The call activity allows you to call a reusable process from within the current process.

See Section A.4.3, "Introduction to the Call Activity" for more information.

The call activity contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.3.6.1 Implementation Properties

Table B–14 describes the properties that can be edited from the **Implementation** properties.

Table B–14 Call Activity Properties

Property	Description
Process	Specifies the BPMN process invoked by the call activity.

B.3.7 Subprocesses

Subprocesses allow you to group BPMN flow objects together to make your process more readable.

See Section A.9.2, "Introduction to Embedded Subprocesses (Inline Subprocesses)" for more information.

Subprocesses contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.3.7.1 Implementation Properties

Table B–15 describes the properties that can be edited from the Implementation editor.

Table B–15 Subprocesses Properties

Property	Description
Loop characteristics	Determines how many time the subprocess will repeat. This property is read-only Oracle Business Process Composer.

B.3.8 Inline Handlers

Inline handlers are types of subprocesses that allow you to model conditions that happen outside of a normal process flow.

See Section A.9.3, "Introduction to Event Subprocesses (Event Handlers)" for more information.

Inline handlers contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.4 Gateway Properties

The following sections describe the properties for each BPMN gateway.

B.4.1 Exclusive Gateway

The exclusive gateway enables you to split a process into two or more paths.

See Section A.7.2, "Introduction to the Exclusive Gateway" for more information.

The exclusive gateway contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

Table B–16 describes the properties that can be edited from the **Outflows Order**.

Table B–16 Exclusive Gateway Properties

Property	Description
Order	Enables you to determine the order in which outgoing sequence flows are evaluated. The first condition that evaluates to true determines the path the process follows.

B.4.2 Inclusive Gateway

The inclusive gateway enables you to split your process into two or more paths.

See Section A.7.3, "Introduction to the Inclusive Gateway" for more information.

The inclusive gateway contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.4.3 Parallel Gateway

The parallel gateway enables you to split your process into two or more paths when you want your process flow to follow all paths simultaneously.

See Section A.7.4, "Introduction to the Parallel Gateway" for more information.

The parallel gateway contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.4.4 Complex Gateway

The complex gateway splits a process similar to an inclusive gateway. However, it enables you to define a condition that determines if the instance can continue even if not all of the tokens have arrived at the complex gateway merge.

See Section A.7.5, "Introduction to the Complex Gateway" for more information.

The complex gateway contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

Table B–17 describes the properties that can be edited from the **Implementation** editor.

Table B–17 Complex Gateway Properties

Property	Description
Activation Condition	Enables you to define a condition that specifies when the gateway releases the tokens that arrive to it. Each time a new token arrives to the complex gateway the BPMN Service Engine evaluates this condition. If the condition evaluates to true, then the complex gateway releases all the tokens that arrived until that moment.

B.4.5 Event-Based Gateway

The even based gateway enables you to branch your process flow based on the possibility that an event may occur.

See Section A.7.6, "Introduction to the Event-based Gateway" for more information.

The event-based gateway contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

Table B–18 describes the properties of the event-based gateway.

Table B–18 Event-Based Gateway Properties

Property	Description
Instantiate	Causes the event-based gateway to create a new process instance.

B.5 Event Properties

The following sections describe the properties for each type of events supported by Oracle BPM.

B.5.1 The None Start Event

The none start event is used when no instance trigger is specifically defined. See Section A.2.3, "Introduction to the None Start Event" for more information.

B.5.2 The Message Start Event

The message start event triggers a process instance when a message is received.

See Section A.2.4, "Introduction to the Message Start Event" for more information.

B.5.2.1 Implementation Properties

Table B–19 describes the basic properties that can be edited from the **Implementation** editor.

Table B–19 Message Start Properties

Property	Description
Type	This property is read-only for message start events. Message start events can only initiate a conversation between two processes.

Table B–19 (Cont.) Message Start Properties

Property	Description
Implementation	Enables you to determine how the receive task defines a conversation with the send task that invokes it. ■ **Not Implemented**: No implementation is specified. ■ **Define Interface**: Enables you to define how the process is exposed as a service to other BPMN processes and services. Argument definition: Defines the arguments required by the receive task. These are the arguments passed to the process from the invoking process or services. Type: Defines whether the process is invoke synchronously or asynchronously. Operation Name: Defines the name of the operation for this receive task. Other processes and services that invoke this receive task use this operation name. ■ Use From Catalog: Enables you to select an interface defined in the business catalog. Name: Defines the name of the interface. Operation: Determines the operation within the interface used by the receive task.

B.5.3 The Timer Start Event

The timer start event triggers the creation of a process instance based on a specific time condition. See Section A.2.6, "Introduction to the Timer Start Event" for more information.

B.5.3.1 Implementation Properties

Table B–20 describes the properties that can be edited from the **Implementation** editor.

Table B–20 Timer Start Event Properties

Property	Description
Due Type	.Determines whether the timer start event creates a process instance based on a specific date or after a specific interval has passed. ■ Date: Enables you to specify the date and time when the timer start event creates a new process instance. The time and date are specified according to the following format: day/month/year hour:minute PM/AM ■ Interval: Enables you to specify the interval that the timer even waits to create a new process instance. This is specified in months, days, hours, seconds according to the following format: <*number*>M,<*number*>d,<*number*>h,<*number*>s
Expression Mode	Enables you to define either the date or interval using an expression.

B.5.4 The Signal Start Event

The signal start event is similar to a message start event in that it is based on communication from another process or service.

See Section A.2.5, "Introduction to the Signal Start Event" for more information.

B.5.4.1 Implementation Properties

Table B–21 describes the properties that can be edited from the **Implementation** editor.

Table B–21 SIgnal Start Event Properties

Property	Description
Event	Defines the event used to trigger the signal start event. Events are defined in the business catalog.

B.5.5 The Error Start Event

The error start event is used as the start event of an inline handler.

See Section A.2.7, "Introduction to the Error Start Event" for more information.

B.5.5.1 Implementation Properties

Table B–22 describes the properties that can be edited from the **Implementation** editor.

Table B–22 Error Start Event Properties

Property	Description
Exception	.Defines the error exception implemented by the error start event. This is stored in the business catalog.
Catch all Business Exceptions	Select to allow the catch event to catch any business exception.
Catch all System Exceptions	Select to allow the catch event to catch any system exception

B.5.6 None Catch Event

The none catch event is used as a place holder in your process.

The none catch event contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.5.7 Message Catch Event

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

See Section A.4.8, "Introduction to the Message Catch Event" for more information.

The message catch event contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.5.7.1 Implementation Properties

Table B–23 describes basic implementation properties of the receive task.

Table B–23 Message Catch Properties

Property	Description
Type	Determines how the conversation of the message catch event is implemented. A conversation defines the sequence of a group of message events that communicate with other processes or services. A message event can start a conversation with another process or service, or continue a conversation initiated by a previous message event. Possible values are: ■ Initiates: Invokes another BPMN process or service. ■ Continues: Continues the conversation of a process that was previously invoked.

Table B–24 describes the implementation properties when **Initiates** is selected.

Table B–24 Message Catch Properties (When Initiates is Selected)

Property	Description
Implementation	Enables you to determine how the message catch event defines a conversation with the process or service that invokes it. ■ **Not Implemented**: No implementation is specified. ■ **Define Interface**: Enables you to define how the process is exposed as a service to other BPMN processes and services. Argument definition: Defines the arguments required by the message catch event. These are the arguments passed to the process from the invoking process or services. Type: Defines whether the process is invoked synchronously or asynchronously. Operation Name: Defines the name of the operation for this message catch event. Other processes and services that invoke this message catch use this operation name. ■ Use From Catalog: Enables you to select an interface defined in the business catalog. Name: Defines the name of the interface. Operation: Determines the operation within the interface used by the message catch event.

Table B–25 describes the implementation properties when **Continues** is selected.

Table B–25 Message Catch Properties (When Continues is Selected)

Property	Description
Initiator Node	Allows you to select the message event in the current process that precedes this message catch event.

B.5.8 Timer Catch Event

Timer catch events enable you to control the flow of your process using a time condition.

See Section A.8.2, "Introduction to the Timer Catch Event" for more information.

The the timer catch event contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.5.8.1 Implementation Properties

Table B–26 describes the properties that can be edited from the **Implementation** editor.

Table B–26 Timer Catch Event Properties

Property	Description
Due Type	.Determines whether the timer catch event creates a process instance based on a specific date or after a specific interval has passed. ■ Date: Enables you to specify the date and time when the timer catch event is triggered. The time and date are specified according to the following format: day/month/year hour:minute PM/AM ■ Interval: Enables you to specify the interval that the timer even waits trigger the event. This is specified in months, days, hours, seconds according to the following format: <*number*>M,<*number*>d,<*number*>h,<*number*>s
Expression Mode	Enables you to define either the date or interval using an expression.

B.5.9 Error Catch Event

Error catch events are intermediate events used to handle an error that occurs within your process flow.

The error boundary catch event contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.5.9.1 Implementation Properties

Table B–27 describes the properties that can be edited from the **Implementation** editor.

Table B–27 Error Catch Event Properties

Property	Description
Exception	Defines the error exception implemented by the error catch event. This is stored in the business catalog.
Catch all Business Exceptions	Select to allow the error catch event to catch any business exception.
Catch all System Exceptions	Select to allow the error catch event to catch any system exception

B.5.10 Message Throw Event

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

The message throw event contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.5.10.1 Implementation Properties

Table B–28 describes basic implementation properties of the send task.

Table B–28 Message Throw Event Properties

Property	Description
Type	Defines how the conversation of the message throw event is implemented. A conversation defines the sequence of a group of message events that communicate with other processes or services. A message throw event can start a conversation with another process or service, or continue a conversation initiated by a previous message event. Possible values are: ■ Initiates: Invokes another BPMN process or service. ■ Continues: Continues the conversation of a process that was previously invoked.

Table B–29 describes the implementation properties available when **Initiates** is selected.

Table B–29 Message Throw Event Properties (When Initiates is Selected)

Property	Description
Implementation	The implementation drop-down menu enables you to determine how the send task is implemented. ■ **Not Implemented**: No implementation is specified. ■ **Service Call**: Configures the message throw event to invoke a service contained in the business catalog. Name: Determines the service invoked by the message throw event. Operation: Defines which operation within the service is invoked. ■ **Process Call**: Configures the message throw event to invoke another BPMN process. Process: Determines the BPMN process called by the message throw event. Node: Determines the flow object called by the message throw event.

Table B–30 describes the implementation properties when **Continues** is selected.

Table B–30 Message Throw Event Properties (When Continues is Selected)

Property	Description
Initiator Node	Determines the message event that precedes this send task within the conversation.
Inputs	Defines the arguments required to invoke the operation the message start event exposes.
Type	Displays the process type as defined in the initiator. This property is read-only.
Operation name	Defines the name of the operation for this message catch event. Other processes and services that invoke this message catch use this operation name.

B.5.11 Signal Throw Event

You can use signal events to communicate a message to all the processes that are configured to wait for that message.

The signal throw event contains properties shared by multiple BPMN flow objects. See Section B.1, "Common Properties" for a list of these properties.

B.5.11.1 Implementation Properties

Table B–31 describes the properties that can be edited from the **Implementation** editor.

Table B–31 Signal Throw Event Properties

Property	Description
Event	Defines the event used to trigger the signal start event. Events are defined in the business catalog

B.5.12 None End Event

The none end event is used as a place holder in your process.

See Section A.2.8, "Introduction to the None End Event" for more information on using the none end event.

B.5.13 Message End Event

The message end event is used to send a message to another process or service when the process is completed.

See Section A.2.10, "Introduction to the Message End Event" for more information.

B.5.13.1 Implementation Properties

Table B–32 describes the properties that can be edited from the **Implementation** editor.

Table B–32 Message End Properties

Property	Description
Type	Defines how the conversation is implemented. Since message end events can only be configured to continue a conversation, this property is read-only.
Initiator Node	Determines the message event that precedes this send task within the conversation.
Inputs	For a synchronous process, this property defines the output arguments used to invoke the operation defined by the start or catch message event. For an asynchronous, this property defines the input and output arguments required by the callback operation defined by this end event.
Type	Displays the process type as defined in the initiator. This property is read-only.
Operation Name	For an asynchronous process, this property defines the name of the operation for this message catch event. Other processes and services that invoke this message catch use this operation name. For a synchronous process, this property defines the operation of the event that precedes this end event. It can be a start or a catch event.

B.5.14 Signal End Event

You can use the signal end event to communicate a message to all the processes that are configured to wait for that message. This message communicates to these processes that the current process has finished.

B.5.14.1 Implementation Properties

Table B–33 describes the properties that can be edited from the **Implementation** editor.

Table B–33 SIgnal End Event Properties

Property	Description
Event	Defines the event used to trigger the signal start event. Events are defined in the business catalog.

B.5.15 Error End Event

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

See Section A.2.9, "Introduction to the Error End Event" for more information.

B.5.15.1 Implementation Properties

Table B–34 describes the properties that can be edited from the **Implementation** editor.

Table B–34 Error End Event Properties

Property	Description
Exception	Defines the error exception implemented by the error catch event. This is stored in the business catalog.

B.5.16 Terminate End Event

The terminate end event is used to immediately stop a process. When a terminate end event is reached, the process stops immediately.

See Section A.2.11, "Introduction to the Terminate End Event" for more information.

B.6 Measurement Mark Properties

Measurement marks enable you to measure a business indicator of type measure at a certain point in the process or in a section of the process. The following types of measurement marks are supported:

- Start Measurement Mark
- End Measurement Mark
- Snapshots

Table B–35 Measurement Mark Properties

Property	Description
Name	Defines the name for this measurement mark.
Description	.Provides an optional description for this measurement mark.
Type	Displays the type of measurement mark. This property is read-only.

Table B–35 (Cont.) Measurement Mark Properties

Property	Description
Business Indicators	Defines the business indicators assigned to this measurement mark. See Section 12.9, "Working with Business Indicators and Counter Marks" for more information.

.

B.7 Sequence Flow Properties

Sequence flows define the order or sequence that work is performed within a process. The following sections describe the sequence flow properties you can edit using Oracle Business Process Composer.

B.7.1 Default Sequence Flow

Table B–36 describes the properties of default sequence flows.

Table B–36 Default Sequence Flow Properties

Property	Description
Name	Defines the name of this sequence flow. This name appears next to the sequence flow in your process diagram.
Description	Provides an optional description of this sequence flow. Adding a description can make your process more readable.

B.7.2 Normal Sequence Flow

Table B–37 describes the properties of default sequence flows.

Table B–37 Normal Sequence Flow Properties

Property	Description
Name	Defines the name of this sequence flow. This name appears next to the sequence flow in your process diagram.
Description	Provides an optional description of this sequence flow. Adding a description can make your process more readable.

B.7.3 Conditional Sequence Flow

Table B–38 describes the properties of default sequence flows.

Table B–38 Conditional Sequence Flow Properties

Property	Description
Name	Defines the name of this sequence flow. This name appears next to the sequence flow in your process diagram.
Description	Provides an optional description of this sequence flow. Adding a description can make your process more readable.

Table B–38 (Cont.) Conditional Sequence Flow Properties

Property	Description
Condition	Specifies the expression used to evaluate this conditional sequence flow. You can define an expression by clicking **Edit** to launch the expression editor. See Section B.4.1, "Exclusive Gateway" for more information on configuring the order in which conditional sequence flows are evaluated.

C

Web Form and Web Form Control Property Reference

This appendix provides a description of the properties available for web forms and web form controls.

This appendix contains the following sections:

- Section C.1, "Web Form Properties"
- Section C.2, "Web Form Control Properties"

C.1 Web Form Properties

You can control the behavior and appearance of the web form by changing its properties. Properties are edited using the properties editor which contains tabbed panes that group the web form properties by function.

C.1.1 Settings Tab

Property	Description
Form Name	Defines the name of the form.
Description	Provides a description of the web form. By default all forms use the description "Edit the form to change this description." Change this description to something specific to your form. The description appears as a tool tip when you mouse over the area just to the right of the form's share icon in the forms editor. The description is also visible when you view individual submission documents.
Printable	Displays a print icon at the top of your form. If you do not want users to print your form, disable this checkbox. You can control which form fields are visible in the PDF print view via the printable property in each web form control of your form.
Save	Causes all submissions for this form to be stored in forms' submission repository. This property is enabled by default. If you deselect this checkbox, the form submission will still be logged in the submission repository and you will be able to view the metadata about the submission (time/date submitted, success/failure conditions) but no form field data is saved.

Property	Description
Save PDF	Enables the user to save the file as a PDF. This property can only be enabled if the save property is enabled. When selected a PDF image of the file is also saved in the forms submission repository.

C.1.2 Style Tab

Property	Description
Width	Specifies how wide your form will be. The default "regular" width is 600px, but the drop down also includes thin (400px) and wide (800px). You also may pick the custom option; this will enable the box to the right of the Width drop down and let you specify your own width.
Height	Specifies an initial height for your form as it is loading into the browser. It does not dictate the actual height. In general, you do not have to edit the form's height property since the form can resize dynamically. However if your form is very small it can improve the appearance as your form loads if you set height to the actual height of your form.
Controls	Determines whether the options you define for checkbox and radio controls will be displayed vertically or horizontally.
Theme	Enables you to define a theme to which changes the appearance of your form. Oracle BPM provides two global themes: clear or professional blue. The Clear theme gives your form a clear (white) background and the Professional Blue gives your form a soft blue background. You can create a custom theme instead of using the default themes. After you have added the customized them it appears in the drop down list along with the pre-defined options.
Font Name, Font Size, and Font Color	Defines the font name, size and color properties that are applied to all labels and text on your form. You can override these properties for a particular control by editing the properties for each individual control. When you create a form from scratch these properties are blank; the specific font, size and color of the labels and text in your form depend on the theme you have chosen or the template you used when you created the form.

C.2 Web Form Control Properties

You can control the behavior and appearance of each web form control by changing its properties. Properties are edited using the properties editor which contains tabbed panes grouping the properties by function.

C.2.1 Web Form Control Properties - Settings Tab

The settings tab defines the general properties of a control. The following table provides an alphabetical list of all the control properties that appear on the settings tab.

Note: Not all of the properties on the settings tab are used by every web form control.

Property	Description
Control Type	Defines the type for this web form control. You can change the type of the web form control by selecting the type from the drop-down list. This property applies to the following web form controls: ■ Selection controls (drop down lists, radios, and check boxes). ■ Most input controls (text, text area, email, phone, quantity and number). ■ Date controls (date, time, or date/time). This property is populated automatically when you first add a control to a web form. You can change this property if you want to switch a control in your form to a different type of control. This saves you from having to remove the original control and drag in a new one. There are some limitations: you cannot change a selection control to an input control or vice versa. You can switch a checkbox to a drop down list, but you cannot use this property to change a checkbox to a text control. This property is also useful for verifying what kind of controls are in your form. Since you assign new labels to your controls after you drag them in, you occasionally might forget whether you are looking at a text control or phone control, for example. This property lets you know what kind of controls are in your form no matter what the labels say. If your control was generated from a schema element, it will have a Display As property instead of a Control Type property.
CSS Class	Defines the control's class name that was added to your form's XHTML markup. You can use this CSS class to reference the control in any CSS when customizing themes. One built-in CSS class name is 'f-page-break'. If you want to add a page break to the printed view of the form's PDF, tiff, add f-page-break to the control that you want to start at the top of a new page.
Date Format	Defines the date format used in the form. This property applies only to Date controls and the Date portion of the Date/Time control. It supports European date formatting. By using this property, dates entered into the form get translated according to the chosen format and will be reformatted to match the selected format. ■ A date typed into a form field will be reformatted to match the selected Date Format ■ A date entered into a form field will be translated according to the selected format. For example, if you choose a European format of DD-MM-YYYY and enter 10-05-2009 the date value will be translated as May 10, 2009. If you choose a US format of MM-DD-YYY the date value will be translated as October 5, 2009. ■ If you choose any of MM-DD-YYYY using either "-" "/" or "." as the separator, all will be valid for that format but will be translated to the selected separator ■ Users can still enter dates like Feb 3, 2001. It will be translated into the specified format

Property	Description
Display As	Enables you to change the way your control looks on your form (for example, To change a text control to a drop down, select drop down from the list of allowable controls). This will change the control's appearance but will not affect how the control handles data. If you need to modify the control's validation behavior, you must update the schema. The Display As property is not available for the following schema controls: Message, Upload, Image, Video, Link, Repeat, Date, Time, Date/Time. This property applies only to controls generated from XSD schema elements. (If the control was dragged in from the palette, it has a Control Type property).
Enable if Valid	Enables the user to submit the form even if all of the data has not been validated. By default the form's submit button until all required fields are filled and contain valid data. However, sometimes you may wish to override the form's default behavior and allow the user to submit a form even if it is invalid. To do this, deselect this property. This property applies only to Submit controls.
Enabled	Determines whether the control is enabled or disabled when users first access your form. If you check the DISABLED checkbox, users may not enter a value in the control until the control is enabled. (You can enable the control using a Rule.) For example, say you are creating a wedding invitation form and want to know if the people completing your form are bringing a guest. Your form might include a text control for the guest's name that becomes enabled only after users indicate (in another control) that they are in fact coming to the wedding. You are not permitted to disable grouping controls.
Error Msg	Enables you to display a specific error message if the user does not supply a valid value in the control. If you leave this property blank users will get generic feedback (an "invalid value" message, for example) if they supply a bad value-but if you use this property, you can make the error message more helpful. For example, if you are using a pattern that requires the user to enter an area code of 203 or 860 in a phone control, you can use the Error Message property to let users know this explicitly if they try to enter a different area code.
Help	Enables you to provide for your users more detailed help about a specific control. If you enter text here, a icon will appear next to the control on your form. When the user clicks, the help text you supplied in the Help property will be displayed in a floating box.
Hide Label	Determines whether the control's label will be displayed on your form. Check the checkbox to hide the label on your form; leave it unchecked to show the control's label.
Hint	Enables you to create a tool tip that will display in your form when the user mouses over the control. Simply type the text for your tool tip in the Hint field.

Property	Description
Label	Defines the label of this web form control. This label is displayed in your form beside or above the control. You can hide the label in the form by enabling the Hide Label property When you first add a control to a web form, it is assigned an arbitrary and unique name. After creating a control, you can change this to the text you want to appear to the end-user of your web form. You can a control's label either on the form itself (by clicking the control and selecting the label) or by overlaying the arbitrary name in the Properties Label field. Message controls are the only controls without labels-they are used for static text so you don't need a label, too. You aren't restricted to plain text for labels. You may use arbitrary XHMTL when typing the label name.
Labels	Enables you to change the controls labels, although the XSD values for the control in the schema do not change. In other words, you can change what a user sees in the form, but not the underlying values (i.e., you cannot use the [value=label] syntax you use for palette controls to change the element values). This property applies only to controls generated from schema elements. (If you drag a control in from the palette, it has a Options property; if the control was generated from a schema element, it has a Labels property instead.)
Max Length	Used to limit how many characters users can supply in the control. Simply type a number (positive integer) in the Max Length field. Text area controls do not support this property due to an HTML limitation. However it is very easy to add this functionality to the TextareaMaxLength text area control using a business rule. This property is available for text controls and other input controls
Message	Enables you to include a message in your form. In addition to regular text, you may include in this property field arbitrary XHTML markup that will be formatted and displayed by the browser. This property applies only to message controls and is where you enter the static text that will appear on your form.
Min # and Max #	Enables you to define the minimum and maximum values a user can input in a control. These properties appear as part of a Table Control or when your form has an input control inside a repeat control and apply to the input control. Simply type a number (positive integer) in the two property fields. If you specify a minimum value of 1 and maximum value of 10, users must enter values in at least one input control or they will be unable to submit the form, but they may enter as many as ten. The minimum and maximum properties for the Table Control allow the user to add/delete table rows in a form as needed. Min/Max properties are not editable for controls generated from an uploaded schema, since the schema already specifies this via the minOccurs and maxOccurs attributes.

Property	Description
Name	Defines the name of this control. This name is automatically generated and defaults to the control's label minus any spaces and special characters. Spaces and special characters are removed in order to make the name valid for use in rules and, for XML users, this makes the name valid as a xsd schema element name. Control names will be truncated to 32 characters for all the controls except triggers and panels. If you have two controls with the identical labels and at the same level, the control's name is automatically made unique. If you try to edit the name such that it would no longer be unique, Oracle Web Forms will prevent the edit. In order to use a control in a rule the name must be unique in your form. When a control is dropped inside a section control, it is at a different nesting level then a control dropped outside a section. Also two controls, one inside a section called Car and another in a section called Boat are also at different nesting levels. In both cases the form designer will allow you to name the controls the same. For example both Car and Boat can contain a control named VIN. The Name property is used in several different contexts in Oracle BPM: ■ Determines how you reference the control in within form rules. ■ Used when initializing form fields via the Web_Forms_data URL parameter for controls added from the palette. Note: For controls added from XSD schema, you must use the underlying element name to initialize the control via _data. ■ Determines how you refer to form fields from document URIs. ■ Used in Form Action Display Message and Go to URL templates. ■ Used in Doc Action Email Address Templates. ■ Is the name given to the XML element corresponding to the control you drag into your form from the palette. You can change the Name of controls from schema, although schema controls maintain their underlying XSD element name. For example, suppose you are using controls from two schemas in a form and both contain a control named "FName". You could change the name of one of these controls to "FirstName" to make them unique within the form. This is helpful if you're adding rules to the form, or if you want to use the form as a template. Except for _data, controls from XSD use the same rules (as above) as controls from palette.

Property	Description
Options	Defines how to populate the choices the user sees in these controls. This property appears for drop down, radio and checkbox controls. You may have option labels different from option values. The syntax for the options is <value>=<label>. The <label> is what will be displayed on your form and the <value> is what is saved as the selected value when the user submits the form. When you first drag one of these controls into your form they have generic values: Option 1, Option 2 and Option 3. This matches the generic text you see in the Properties area. To supply your own values, simply overlay the values in the Options property with the values you want and add more (or delete) as necessary. If you do not enter both <value> and <label> using the <value>=<label> syntax, then the value will default to the label. As soon as you tab off the options property, options without values will automatically be converted to the syntax. Here the options are entered without values. In this case Oracle Web Forms will default the value to the label. The order of choices in your control will match the order in the Properties area. If you have choices that need a logical order (you'd want a drop down of US states to be sorted alphabetically, presumably), make sure the order is correct in the Properties area. (You can't sort the text you enter in the Options property field but you can cut and paste.) In addition to the generic Option 1, 2 and 3 choices, a drop down control also includes a blank option that by default will appear first in the list. This blank option will appear no matter what text you supply in the Options property. You cannot remove the blank option but you can make one of the other options the default. See setting defaults. What if you want a '=' character to appear in your options? Since the 1st equal sign is the delimiter between the value and the label, you can put an equal in the label as follows. For example, suppose you want your label to be "good = gold." If you enter this as the option, what you'll see in your drop down is the label "gold." To solve this, use this options string: "a=good = gold". Now the label will be as you wish. The choices cannot be changed if they have been generated from an uploaded schema, since the schema specifies the choices. On controls generated from schema, you won't see the Options property in the Properties area. However, you can change the option labels. See Labels.
Password	Causes a the text entered in this field to be blocked out. This property applies only to text controls and other input controls. If you check the Password checkbox, the text the user enters will appear on the form as asterisks (it will be submitted as normal text, however).

Property	Description
Pattern	Enables you to define additional restrictions on the type of data a user inputs into a control. Most controls automatically ensure that users provide the correct data type, but patterns give you the flexibility to impose additional restrictions on what users enter in a particular control. In the Pattern field in the Properties area, type your pattern using XML schema regular expressions. A simple example is a pattern that restricts a text control to only allow strings formatted as a US zip code: \d{5}|\d{5}-\d{4}. If you type this expression in the Patterns property, your form will permit values entered into this field only if they are five digits or five digits followed by the '-' character, followed by 4 digits. When you define patterns you don't have to restrict what the control handles automatically. It is not necessary to enter a pattern [a-z] for a Number control, since users can't type letters in a number field anyway. Since essentially you would be attempting to expand the allowed data types in the control, Oracle Web Forms would ignore this pattern if you entered it. You have to first save the form before the pattern takes effect. Thus patterns cannot be tested in the form designer, only in use mode
Printable	Determines whether or not this form field will appear in the in the PDF document view of the form. If unchecked, the field will not appear in either view. You can set the property on a section control and it will automatically apply to all controls inside the section control. This is often used in web form rules.
Required	Defines that a control is required when you want to force users to enter a valid value in the control before they submit your form. To do this, check the Required checkbox in the Properties area. If the data is optional, leave the checkbox unchecked. As soon as you mark a control required, red asterisks appear next to it on your form. Until users populate all required controls with valid data, they will not be able to submit the form because the Submit button will be disabled. You cannot mark grouping controls (tabs, sections, panels and repeats) required. Input controls that are direct children (directly inside) of repeat controls also cannot be marked required, nor do they need to be, since the minimum and maximum properties define this. You may, however, make controls required when they are inside sections, tabs and panels. You cannot edit the required property for controls that have been generated from an uploaded schema, since the schema already specifies this via the minOccurs attribute. If a control from schema appears as required and you don't want it to be required, edit the XSD and set minOccurs=0. Re-upload the XSD. Then the control will no longer appear required. This property is very useful when using rules to hide/show sections depending on something else in the form. If you hide the section you may have to set the required property to false.
Sensitive	Determines whether the value saved in the submissions repository is hidden when viewed in the forms in the web submissions user interface. The value is not currently encrypted in the database. Click the checkbox if the data users will enter into this field contains sensitive data that should not be visible when viewed in the user interface.

Property	Description
Time Format	Defines the time format used by the control. This property applies only to the Time control which is created by selecting the Time or Date/Time option from the Date control drop down. It supports standard and military time formats. By using this property, time entries entered into the form get translated according to the chosen format and are reformatted to match the selected format. ■ A time entry typed into a form field will be reformatted to match the selected Time Format. ■ A time entered into a form field will be translated according to the selected format. For example, if you choose a military time format of hh:mm:ss and enter 2:00 PM the time value will be translated as 14:00:00.
Visible	Determines whether the control is visible or hidden when users first access your form. It is almost identical in concept to the Enabled property, but instead of disabling the control you hide it and write a Rule that makes it visible based on what users enter in another control.
# of Rows	Defines the initial size of the text area. This property applies only to text area controls. Scroll bars will appear automatically when the user reaches the # of rows you specify.

C.2.2 Web Form Control Properties - Style Tab

Use the Style tab to define display-specific properties of the text control. The following table provides an alphabetical list all of the properties available on the **Style** tab.

> **Note:** Not all of the properties listed below are available for every web form control.

Property	Description
BG Color	Enables you to specify the color that will appear behind the control. Type any valid CSS color name or its hexadecimal RGB equivalent. For example, if you want a red background, you can type the word RED or #aa2211.
Bold	Causes the control's label to be bold.
Border Color	Defines the border color of the control.
Border Width	Enable you to specify the thickness, format and color of a border around any control. With the exception of sections and repeats, the border is applied only to the area where the user enters data and does not surround the control's label. For tab controls, click the unlabeled area to the right of your right-most tab to access the border properties. Your border properties will be applied uniformly to each individual tab in the tab group. You may not apply borders to panels but you may create borders around controls you drag inside panels. Remember to make the border color property different from the form's background color or users will not be able to see the borders. Specify the color using the standard CSS convention of typing the name of the color or its hexadecimal equivalent. When specifying the border width property, using pixels (5px, for example) works best.
Border Style	Defines the border style of the control.

Property	Description
Italic	Displays the controls label in italics.
Item Width	Used to change the layout of the options from vertical (one radio/checkbox button below the next) to horizontal. This is useful to save vertical space on long forms. And also useful to improve ease of use for forms with questions that each have the same set of options. This property is used by the radio and checkbox controls
Label Color	Enables you to change the font size and color for any specific control on the form. Specify the color by typing any valid CSS color name or its hexadecimal equivalent. These properties work well when you want your entire label to have the same size and color, but for more sophisticated labels you can type XHMTL in the control's label property field. For instance, use XHMTL if you want to apply two different font colors inside the same label. Typing XMHTL also gives you more font precision, since the label size property lets you pick generic font sizes only--small, medium, and so on. There may be controls for which you want a font size somewhere between the small and medium options in the drop down, for example.
Label Size	Defines the size (in pixels) of the label.
Margin	Enables you to add extra space around the outside of your control, while the padding permits you to add extra space within your control's boundary. Type your margin and padding properties using standard CSS syntax. Type a single value such as 5% to apply a 5% margin or padding uniformly on all four sides of the control. You also can type specify different margin or padding on the various sides of the control: for example, a 5px 10px 15px 20px margin property will give your control a five-pixel top margin, a ten-pixel right margin, a 15-pixel bottom margin and a 20-pixel left margin.
Option Width	Used when option values are longer than a single line. In some cases this may cause the option value to start below the radio or checkbox on certain browsers. You can correct this by selecting a smaller option width such as 80%. This property is used by the Radio and Checkbox controls.
Padding	See the margin property.

Property	Description
Width	Used to specify the width of the control. For input controls, the property specifies the width of the area in which users enter data; for example, you might narrow a control used for entering zip codes or widen a control for a full first, middle, and last name. A common example is setting a text control's width to 90% for street addresses when you use the text control inside a panel as part of a two-column layout. (When a control is inside a panel or other grouping control, the width percentage is relative to the grouping control. When a control is not inside a grouping control, the width percentage is relative to the entire form.) This property is crucial for panel controls because you must adjust the panel widths before your columns will line up side by side. For tab controls, you access the width property by clicking the unlabeled area to the right of the last tab in your tab group; the width you specify will be applied uniformly to each tab in group. (See Panels and Tabs for more details.) There is no Width property for section or repeat controls; however, you can edit the Width property of controls you drag inside these controls. Also, if you drag a section or repeat inside other group controls, Oracle Web Forms automatically adjusts the width to a sensible size. When setting a control's width property you may use standard CSS-relative values; for example, 5%, 5em, 5ex or 5px.

D

Web Form Rules Examples

This appendix provides multiple examples and use cases for using web form rules within a web form. For general information about form rules, see Chapter 10, "Working with Web Form Rules."

D.1 Calculate a Total

The following example assumes you have a form with three controls and you have assigned them Names N1, N2 and T respectively. When a user enters a value in either N1 or N2 you want to set the value of T to the sum of N1 and N2.

The rule would be written as

```
if (N1.value > 0 || N2.value > 0) {
    T.value = N1.value + N2.value;
}
```

This rule will automatically run whenever the user types something in N1 or N2 and will set the value of T appropriately. You can use any legal JavaScript operators in the expression such as subtraction or multiplication. However, it is important to ensure that the calculated value is valid in the context of type of T. For example, if T was of type integer and the computed value of the expression was decimal (such as 1.5), then the rule would be attempting to set an invalid value in T. This will generate an error. The rule will set the value as requested, but will mark the field as invalid and take appropriate action such as disabling the submit button, displaying the control with a red background etc.

Also, if controls are added to the form from the palette, it is important to ensure they have the correct type. For example, for a numeric calculation as described above, the controls should be of type Numeric.

D.2 Show/Hide a Billing Address

The following example assumes you have a form with two controls and you have assigned them names B and S respectively. B is a checkbox with a single option whose value is Yes. If checked the user wishes to enter a different billing address in S and you want to display S.

The form rule would be written as

```
if (B[0].value == 'Yes') {
    S.visible = true;
} else {
    S.visible = false;
```

```
}
```

This form rule will automatically run whenever the user checks or unchecked B and will show/hide the billing address section S. You can use any valid JavaScript expression to compute the visible property of S as long as it evaluates to a boolean true or false value. In this example, you would typically set the checkbox B to be initially unchecked and the section S to be initially hidden.

D.3 Show/Hide Message

In the following example, the form has a radio control named Facility and makes a message visible depending on the selected options.

```
if (Facility.value == 'Boston' ||
    CompanyFacility.value == 'New York')
{
    Msg.visible = true;
} else {
    Msg.visible = false;
}
```

D.4 Enable/disable a question

This example assumes you have a form with two controls and you have assigned them Names B and Q respectively. B is a checkbox with a single option - Yes. If checked the user is a smoker and you wish to ask an additional question in Q. The rule would be written as

```
if (B[0].value == 'Yes') {
    Q.enabled = true;
} else {
 Q.enabled = false;
}
```

This rule will automatically run whenever the user selects or deselects B and will enable/disable the question in Q. Again, you could use any legal JavaScript expression to compute the enabled property of Q as long as it evaluates to a boolean true or false value.

In this example, you would typically set the checkbox B to be initially unchecked and the control Q to be initially disabled.

D.5 Compute Subtotals for Repeating Items

The following form rule is an example of working with repeating items. For example, if you have a form with a repeating section representing an Item that the user may purchase. Each section has a Price (with Name P), a Quantity (Name Q) and a Subtotal (Name S). There are multiple items on the page and the number of items on any given page is unknown. The price field is filled in automatically. When the user enters a value in the quantity field for any item, you wish to compute the subtotal.

The rule is written as follows:

```
for (var i = 0; i < S.value.length; i++) {
  if (Q[i].value > 0) {
    S[i].value = Q[i].value * P[i].value;
  } else {
```

```
    S[i].value = 0;
  }
}
```

This rule will automatically run whenever the user enters a value in the quantity field for any item. It will compute the subtotal for each item, for which the quantity is greater than 0 and fill in the subtotal field for that item with the computed value. If a particular item does not have a quantity, the subtotal is not computed.

D.6 Compute an Invoice Total

The following form rule is an example of working with repeating items. It assumes you have a control named Total with Name T. You want to set the value of Total to be the total invoice price, which is the sum of all the computed subtotals above. This rule would be written as:

```
var tot = 0;
for (var i = 0; i < S.value.length; i++) {
  tot = tot + S[i].value;
}
T.value = tot;
```

This rule will runs whenever a subtotal is updated, for example, when it is updated via the rule above. It will add the values of all the subtotals to arrive at an invoice total. You must use a temporary variable to compute the total.

If you write the rule as:

```
T.value = 0;
for (var i = 0; i < S.value.length; i++) {
  T.value = T.value + S[i].value;
}
```

it will not work correctly. This is due to internal limitations in the way rules are evaluated. Note that this rule is working with controls inside a repeat control. To handle the case of a item being deleted from the repeat you need the following addition assuming that the repeat control is named ExpenseRepeat. Table controls are repeats with a different layout. Thus the same applies to the table controls. If your table is named Expense then the repeat is automatically named ExpenseRepeat.

```
if (ExpenseRepeat.itemRemoved) {;}
```

D.7 Textarea Max Length

This example form has a textarea control named 'Desc' where the user can enter up to a 500 character description. In HTML there is no way to set a maxLength on a textarea control. Due to this limitation, the textarea control does not have a maxlength property like the text control does.

It is possible to do this via a form rule. On this control we also set the ErrorMsg property to the string 'You must limit your description to 500 characters'. This message is automatically displayed when the description control is set to invalid by the following business rule.

```
if (Desc.value.length > 500) {
  Desc.valid = false;
} else {
  Desc.valid = true;
```

```
}
```

You can customize the error message by adding the following line to your rule.

```
Desc.status = 'Invalid. Max 20 chars allowed and you have ' + Desc.value.length;
```

The error message will tell the user how many characters they are over the maximum allowed

D.8 Textarea Newline and Break

Users typically enter multi-line text into textarea controls. If you want to display that text in an HTML context, for example on a web page or in an html formatted email or in your form's form action display message you will need to replace newline characters with HTML breaks. This is due to the fact that line breaks entered into a web form textarea are represented by a single newline character \n while line breaks in an html context are represented by the html break characters.

The following example has a textarea control named Description and a hidden control named DF. The user types into the visible control named Description and a business rules converts the newline characters \n into html breaks.

```
var x = Description.value;
x = x.replace(/\\n/g,"<br/>");
DF.value = x;
```

D.9 Dropdown Options

The following example automatically sets the option selected in one dropdown based on the option selected in another. This is often useful when you have a form with choices that were dynamically populated. For example, imagine product choices which are descriptive text. When the user selects a product, your form needs to perform an action based on a product ID rather than the descriptive product text. A nice way to do this is to have the rule that dynamically populates the product choices dropdown also populate a product ID dropdown which remains an invisible control in the form. The product choices dropdown control was named P and the product ID dropdown control was named ID

The 1st rule "Load Products" populates both the visible and hidden dropdowns with options from a database.

```
Load Products:
--------------
if (form.load) {
eval('x=' + http.get('http://localhost:8082/database/products'));

    var opts1 = [];
    var opts2 = [];

    for (var i=0; i < x.resultSet.length; i++) {
        if (x.resultSet[i]) {
            opts1[i] = x.resultSet[i].description;
            opts2[i] = x.resultSet[i].productId;
    }
}


Products.options = opts1;
```

```
PID.options = opts2;
Products.value = opts1[0]; // default to 1st product option
PID.value = opts2[0];
}
```

D.10 Finding a Selected Options Index

The 2nd rule *Select Product ID* keeps the hidden PID dropdown synchronized with the visible Product description dropdown

```
Select Product ID Rule:
-----------------------
if (Products.value.length > 0)
{
    var i;
    for (x in Products.options) {
        if (Products.value == Products.options[x])
            i = Products.options.indexOf(Products.options[x]);
}

PID.value = PID.options[i] + '';
}
```

In v4 rules using hidden dropdowns to keep descriptive option labels visible to the user while keeping cryptic database values hidden are often no longer necessary. Dropdown options have values distinct from the human visible option labels. The above can now be achieved with a single simpler rule:

```
for (var i=0; i < x.resultSet.length; i++) {
    if (x.resultSet[i]) {
            opts1[i] = x.resultSet[i].productId+ '=' + x.resultSet[i].description;
    }
}
Rdocnum.options = opts1;
```

Here is another rule that dynamically populates both the product choices and product ID dropdowns. This rule calls a REST Service which returns an object rather than the result set returned by the database connector as shown above. See the section on dynamic content for more details.

```
if (S.value.length > 0) {
    eval('x=' + http.get('http://localhost:8182/products/?category=' + S.value));
    P.options = x.products;
    ID.options = x.ids;
}
```

D.11 Synchronized Selects

The Product Search example above is often used in conjunction with a hidden select control. Imagine that your database table contains a list of products. Each product has product description also a unique product ID. The user needs to select a product from a dropdown on your form. You want to populate the dropdown with the product descriptions. The users do not need to see or know the product IDs but you need to use the ID as the key into the database for other selects. To do this add another hidden dropdown to the form and populate it with the IDs. This example has a visible dropdown name Products and an invisible dropdown named PID. See the rule above that populates these dropdowns dynamically from the database.

This rule below keeps the PID selected option in sync with the selected Product.

```
var i;
for (x in Products.options) {
// Determine the index of the selected product in the Products dropdown options
if (Products.value == Products.options[x])
    i = Products.options.indexOf(Products.options[x]);
}


// Changed the selected PID to match the selected Product
PID.value = PID.options[i] + '';
```

D.12 Clearing Dropdown Options

This sample resets a dropdown option to the automatically added blank option. For dropdowns added from palette controls and from schema, Oracle Web Forms automatically adds a blank option so the dropdown initially shows no choice by default. To reset the dropdown, set the dropdown control's value to null not the empty string. The empty string will not work since the empty string is not a valid option. This form resets the dropdown named size whenever the value of the product option changes.

```
if (product.value.length > 0) {
    size.value = null;
}
```

D.13 Default Option

When your options are set dynamically as shown below in a business rule, you cannot set a default in on the form designer. You need to set the default in the rule. If your options have <value>=<label> where value is different from label, make sure you set the <control>.value to <value> not <label> and not <value>=<label>

```
if (form.load) {
    var cc = ['R=Red', 'B=Blue', 'G=Green'];
    Colors.options = cc;
    Colors.value = 'B';
}
```

D.14 Checkbox Options - Assigning Color to Checkbox Choices

Checkbox controls are different from all other Oracle Web Forms palette controls in that they are multi-select. Therefore the way to write rules with checkbox controls are in many ways similar to rules with repeat controls. This rule has a checkbox controls with name colorPalette with the options: purple, green, blue, yellow, orange. The form also contains a text control with name colorChoice. This rule assigns colorChoice the choices selected from colorPalette.

```
var choices = '';
for (var i = 0; i < colorPalette.value.length; i++)
{
    choices = choices + colorPalette[i].value;
}
colorChoice.value = choices;
```

Notice that similar to repeat controls, due to an internal evaluation limitation, you must collect the choices in a variable inside the for loop. And then assign that control Name.value to that variable outside the for loop.

This rule is another example showing how checkbox controls are array types.

```
if (colorPalette.value.length > 0)
{
    colorChoice.value = 'Thank you for choosing colors';
}
else
{
    colorChoice.value = 'Please choose colors...';
}
```

D.15 Checkbox Options - Making a Control Visible/Invisible Based on Checkbox Choices

This rule makes visible/invisible a control based on which checkbox options a user selects. This form contains a multi select checkbox named Structures. If the user selects the option "Detached Garage" or "House", we want to make visible a text field named Details.

Again since a checkbox is multi select, it is handled as an array. The array will contain all selected (checked) options.

It is important to note that when a checkbox is added to the form from the palette and its options are multiple words containing spaces, the option array has converted each space character to the '_' character. We must make the comparison as shown below. Checkbox controls from schema do not have space replaced with '_'.

```
var found = false;
for (var i = 0; i < Structures.value.length; i++)
{
    if (Structures[i].value == 'Detached_Garage' ||
        Structures[i].value == 'House') {
      found = true;
      break;
    }
}
if (found == true) {
    Details.visible = true;
} else {
    Details.visible = false;
    Details.value = null;
}
```

Note that when we hide Details we also clear its value. This is because the user may have selected one of the Structures checkboxes that made Details visible AND entered a value into Details. And then they may have changed their minds and deselect the option that caused Details to become visible. If you don't want the value entered into Details to be in your form submission, clear the value when hiding it.

D.16 Checkbox Initialization

Since checkbox options are multi-select, in order to select multiple options via a rule you must use this syntax. In this example CB is the name of a checkbox controls with the following options: red, green, blue. This rule selects all of the options.

```
CB.value = ['red', 'green', 'blue'];
```

To clear all checked option in the control named CB:

```
CB.value = [];
```

D.17 Displaying Selected Checkbox Labels

In this example, the rule displays the labels of the checkboxes the user selects.

```
var selectedcolors = '';

for (var i = 0; i < RGB.value.length; i++)
{
    var v = RGB[i].value;
    for (x in RGB.options) {

        var opt = RGB.options[x];
        var val= opt.split('=')[0];
        var lab= opt.split('=')[1];
        if (v == val) {
            selectedcolors = selectedcolors + ' ' + lab;
        }
    }
}
```

D.18 Repeating Checkboxes

Checkboxes inside repeat controls must be treated as an array (each checkbox control's values) of checkbox option values which is inside another array (the repeating checkbox control itself). This form example has a repeating section containing two controls -- Message which is a text control and AreYouAttending which is a checkbox control with a single option 'yes'. To access the selected options the syntax is:

AreYouAttending[i].value[0] == 'yes'

```
for (var i = 0; i < AreYouAttending.value.length; i++)
{
    if (AreYouAttending[i].value[0] == 'yes') {
        Message[i].value = Name.value +
               ' is attending event #' + i;
    }
}
```

D.19 Display a Message Control Inside a Repeat Control

If you put a message control inside a repeat control, you need to add a rule to the message control to have the message repeat when a user clicks to add a new repeat item. In the example below, the messages appear correctly in the first item, but a rule is necessary to have them appear when the user adds a second, third, or more items.

In the example below, the rule repeats both dynamic text (*Contest Nomination For: [Name]*) in the first message control and static text (*By signing this consent form...*) in the second message control.

```
if (ConsentRepeat.itemAdded)
{
    var index = ConsentRepeat.itemIndex;
```

```
    ConsentNominationForMsg[index].value = 'Contest Nomination For: ' +
ClubName.value + ' by ' + NominatorSName.value;

    ConsentMsg[index].value = 'By signing this consent form you hereby agree that
    it is ok to use a photo or video with your image. Blah, Blah, Blah By signing
    this consent form you hereby agree that it is ok to use a photo or video
    with you in it. By signing this consent form you hereby agree that it is ok to
    use a photo or video with you in it.';
}
```

D.20 String Concatenation

Message controls can be used in business rules to create summary information on your form from values entered into earlier form fields. This rules uses javascript variables to concatenate form field values, text strings and html to format a nice summary page:

```
var totalAssets = TotalAutoValue.value + TotalBankValue.value +
TotalRealEstateValue.value;

BasicSummaryMsg.value =
"<b>Name:</b> " + FirstName.value + " " + LastName.value + "<br/>" +
"<b>Phone:</b> " + Phone.value + "<br/>" +
"<b>Email:</b> " + EmailAddress.value;

if (MilitaryOrCivilian.value == 'Military') {
//Military
DetailedSummaryMsg.value =
"<b>Military Info:</b><br/>" +  "Military ID: " + MilitaryID.value + "<br/>" +
"Rank: " + Rank.value + "<br/>" +
"CurrentTitle: " + CurrentTitle.value + "<br/>" +
"Years of Service: " + YearsOfService.value + "<br/>";
} else if (MilitaryOrCivilian.value == 'Civilian') {
//Civilian
DetailedSummaryMsg.value =
"<b>Civilian Info:</b><br/>" +
"SSN: " + SSN.value + "<br/>" +
"Current Employer: " + CurrentEmployer.value + "<br/>" +
"Current Title: " + CurrentTitle2.value + "<br/>" +
"Start Date: " + StartDate.value + "<br/>";
}

FinancialSummaryMsg.value =
"<b>Total Assets:</b> $" + totalAssets +
"<br/>" + "Total Bank Assets: $" + TotalBankValue.value + "<br/>" +
"Total Real Estate Value: $" + TotalRealEstateValue.value + "<br/>" +
"Total Auto Value: $" + TotalAutoValue.value + "<br/>";
```

Note when using field values from repeat controls you must use a javascript var and assign the concatenation to the var and then the var to the Oracle Web Forms message control value. For example imagine you have a message control named Summary and a repeat control named Account:

```
var acctSummary;
for (var i = 0; i < Account.value.length; i++) {
    if (Q[i].value > 0) {
        acctSummary = acctSummary + 'Account #' + i + ': ' + Account[i].value +
'<br/>';
    }
}
```

```
Summary.value = acctSummary;
```

D.21 Dynamic Labels, Help, Hints

You can set the value of control labels, help and hint dynamically in a rule. For example imagine you do not know the label, help or hint at design time but would rather set it dynamically when a user opens your form.

```
if (form.load) { Text99.label = 'New Label'; Text99.hint = 'New Hint'; Text99.help
= 'New Help'; }
```

In the above example the label, help and hint is still hard-coded. It's just being set from the rule rather than in the form designer controls' properties. To make this more useful you can initialize these properties from _data parameters:

```
if (form.load) { Text99.label = _data.getParameter('label'); Text99.hint = _
data.getParameter('hint'); Text99.help = _data.getParameter('help'); }
```

Since _data.getParameter enables access to values passed to the form that are not bound to actual controls this is often a very useful pattern.

D.22 Visible/Invisible

This rule makes the message control nickNameThankYou visible when the user enters a value into the nickName input text control. And then hides the message control if the user deletes the value in nickName.

```
if (nickName.value.length > 0 )
{
    nickNameThankYou.visible = true;
}
else
{
    nickNameThankYou.visible = false;
}
```

D.23 Visible/Invisible Section

Often section controls contain many inner controls. For example imagine a form that contains a person's medical history. One of the questions on the form asks if the patient uses a hearing aid. If they answer yes, then you want to collect more details on their hearing aid usage such as left ear, right ear, bilateral; hearing aid brand; etc. If they answer no then you want to hide all the questions specific to hearing aids. Also when they answer is yes you want to require them to answer all the hearing aid detailed questions.

Avoid using message control inside of a section that contains other controls that you may want to make invisible. Since a message control always contains a value, it can cause a section, or other controls in a section, to become required, and this can disable the form's Submit button. If you must include a message control, place it outside the section. Another alternative is to write rules for the individual controls within a section to set them to visible/invisible or required/not required

Imagine this example form has a section named HearingAid. By default HearingAid visible is set to false in the form designer.

When they answer yes, you must set HearingAid.visible=true AND also each required field inside the section to field.required = true. If they then change the answer to no then another rule makes the HearingAid.visible=false also need to set all field.required=true again. If the HearingAid section contains many child controls this rule becomes very long and tedious to write

We can simplify this by using the required property for sections. In the designer default all controls that must be answered inside HearingAid to required. Default the HearingAid section to not required and not visible. Your rule can be much simpler. By setting HearingAid.required=false all the inner controls recursively also become required=false.

```
if (useAid.value == 'no') {
    // Hide
    HearingAid.visible = false;
    HearingAid.required = false;
} else {
    // Show
    HearingAid.visible = true;
    HearingAid.required = true;
}
```

D.24 Select Tab

This rule makes a specific tab the selected tab based on the choice of a radio control. The radio is named SelectTab and has three options: person, auto, home. The tabs are named personTab, autoTab and homeTab. Tabs also can be select based on trigger controls or other input controls using the same method.

```
if (SelectTab.value.length > 0)
{
    autoTab.selected = false;
    homeTab.selected = false;
    personTab.selected = false;

    if (SelectTab.value == 'Auto')
        autoTab.selected = true;
    else if (SelectTab.value == 'Home')
    homeTab.selected = true; else personTab.selected = true;
}
```

D.25 Next Tab

This form contains a trigger control at the bottom of each tab labeled "Next". When "Next" is clicked the trigger rule executes and makes the next tab the selected tab. This assists the user in navigating through the form. The Tabs are named T1, T2, T3, T4. The trigger controls are named C1, C2, C3

```
// Navigate Tabs
if (C1.clicked) {
    T2.selected = true;
} else if (C2.clicked) {
    T3.selected = true;
} else if (C3.clicked) {
    T4.selected = true;
}
```

D.26 Expand/Collapse Section

This form has three sections. The first section is expanded and the 2nd and 3rd are collapsed. When the user files in the 1st section they click a "Next" trigger control which causes that section to collapse and the next section to expand. The trigger controls are named next1 and next2. And the sections are named: step1, step2, step3. Try this membership form to see the rule in action.

```
if(next1.clicked)
{
    step1.expanded = false;
    step2.expanded = true;
}

if(next2.clicked)
{
    step2.expanded = false;
    step3.expanded = true;
}
```

D.27 Security Subject Information

You can use a form.load rule to pre-populate fields in your form with information about the currently logged in user. For example, if you have controls in your form named Id, FirstName, LastName, Email and Roles, the following rule will prefill those fields as indicated below.

```
if (form.load) {
    // User Information
    Id.value = _data.getParameter('subject.id'); // Username
    FirstName.value = _data.getParameter('subject.first.name');
    LastName.value = _data.getParameter('subject.last.name');
    Email.value = _data.getParameter('subject.email');

    var roles = _data.getParameter ("subject.roles");
    if (roles) {
        eval ('x=' + roles);
        Roles.options = x;
    }
}
```

This rule is useful in a workflow where you want to make a the tab named Review visible only for the workflow activity named Manager Review.

```
if (form.load) {
    if (_data.getParameter('flow.activity.name') == 'Manager Review')  {
        Review.visible = true;
    }
}
```

D.28 Multiple Choice

This rule makes the appropriate input text controls visible depending on the choice a user makes in a radio option controls searchChoice.

```
if (searchChoice.value == 'Organizations')
{
    orgname.visible = true;
```

```
    firstname.visible = false;
    lastname.visible = false;
    clientId.visible = false;
}
else if (searchChoice.value == 'Individuals')
{
    orgname.visible = false;
    firstname.visible = true;
    lastname.visible = true;
    clientId.visible = false;
} else if (searchChoice.value == 'Client ID')
{
    orgname.visible = false;
    firstname.visible = false;
    lastname.visible = false;
    clientId.visible = true;
}
```

D.29 Dynamic Options

Select control options (radios, checkboxes, dropdowns, T/F) can be set dynamically via rules rather than statically via the control's options property. However if the control comes from an XSD schema data source rather than one of the standard palette controls, then the designer must take care to not set the options to something outside of what is valid for that schema element. For example if your XSD has a string enumeration and list valid options as 'red', 'green', and 'blue', then you should not use a rule to dynamically set the options to 'small', 'medium', 'large'. If you do then your form will not work correctly in use mode. If a user selects the option 'small' they will get a validation error on the form. This is because 'small' is not one of the options allowed by your underlying XSD schema.

D.30 Triggers and Dynamic Options

This rule is executed when the user clicks the trigger controls with Name "search". It then dynamically sets options on a dropdown list control with Name *coffeeShopList*.

```
if (search.clicked)
{
    coffeeShopList.options = ['Koffee', 'Starbucks', 'Willoughbys', 'Dunkin
Donuts']; }
```

Now replace the hard coded list of coffee shops with a rule that invokes an http.get. This must return an X-JSON header which containing a JSON object. The object is evaluated and assigned to the variable x. In this case the JSON object contains an options field of type array.

```
if (search.clicked)
{
    eval('x=' + http.get('http://<webhost>/getCoffeeShopList'));
    coffeeShopList.options = x.options;
}
```

Note: Triggers do not work in repeating items.

D.31 Value Change and Dynamic Options

This rule dynamically sets the options in a dropdown list based on the value selected in another form field. This form contains three fields named Products, Series and Model. The series options are set dynamically based on the product selection. Also when a new product is selected we enable the series dropdown and both clear and disable the model dropdown. This form contains other rules which set the models based on the selected series. Look for the Oracle Web Forms Gallery form named HP Support - 2 under the dynamic forms keyword.

```
if (product.value == 'Laserjet Printers')
{
    series.options = [' ', 'Laserjet5 series', 'Laserjet6 series'];
    series.enabled = true;
    model.options = [];
    model.enabled = false;
}
```

D.32 Dynamic Control Initialization

This rule handles the case of setting multiple control values based on the selection of a single dropdown control. It handles this case better than using a long if/else construct. First add options to the dropdown named SalesRep in the format <value>=<label> where <value> will be used as an index key into an array of details about each person.

```
Megan=Megan Smith
Jim=Jim Brown
Nancy=Nancy Jones
Brian=Brian Jones
```

Next write a rule that first sets up a javascript array with the contact information for each person. The rules then uses the dropdown value to index into the contantInfo array to set the details for the selected person into four other form controls.

```
var contactInfo = {
    "" : {
        name : "",
        email : "",
        phone : "",
        cell : ""
    },
    "Megan" : {
         name : "Megan Smith",
         email : MSmith@mycompany.com,
         phone : "(203) 694-2439 Ext. 516",
         cell : "(203) 337-3242"
    },
    "Jim" : {
        name : "Jim Brown",
        email : jim@comcast.net,
        phone : "203-208-2999",
        cell : ""
    },
    "Nancy" : {
        name : "Nancy Jones",
        email : nancy@snet.net,
        phone : "203-208-2991",
        cell : ""
    },
```

```
    "Brian" : {
        name : "Brian Jones",
        email : BJones@mycompany.com,
        phone : "203-748-6502",
        cell : ""
    }
};

var repId = SalesRep.value;
SalesRepName.value = contactInfo[repId].name;
SalesRepEmail.value = contactInfo[repId].email;
SalesRepPhone.value = contactInfo[repId].phone;
SalesRepCell.value = contactInfo[repId].cell;
```

D.33 Verify User

This rule executes when the user enters a value into the Username text field. It invokes a URL that returns a JSON object that has a boolean field name "exists" that is set to true if the user already exists. If the user already exists this rule then sets the value of a message control, makes that message control visible on the form and sets the Username valid property to false so that Username field displays as invalid to guide the user to make a correction.

```
if (U.value.length > 0) {
    eval('x=' + http.get('http://<webhost>/uniqueUser?username=' + U.value));
    if (x.exists) {
        M.value = 'User: ' + U.value + ' already exists';
        M.visible = true;
        U.valid = false;
    } else {
        M.visible = false;
}
}
```

D.34 Calculate Net Worth

This form contains two rules. One is adding values entered into a column of assets and a column of liabilities and calculating netWorth. The 2nd rule is checking the value of netWorth and displaying an error message and marking netWorth invalid if liabilities exceed assets since the form designer does not want the form to be submittable in that state.

```
if (netWorth.value < 0)
{
    assetsExceedLiabilitiesMsg.visible = true;
    netWorth.valid = false;
}
else
{
    assetsExceedLiabilitiesMsg.visible = false;
    netWorth.valid = true;
}
```

When a rule sets <control>.invalid the control background turns red and the submit button greys out just as if the user had entered an invalid value into a phone control. Oracle Web Forms treats it exactly the same way. This is a good way to dynamically control your form's valid state.

D.35 Dates and Times

Working with dates and times is very common in most forms. The samples below show you how to create the most common business logic with dates and times.

Oracle Web Forms dates can be set in the user's local timezone by using the built-in date, time and date/time methods such as frevvo.currentDateTime(form). Some of the samples below use the javascript Date() object. Since business rules run on the form server these dates will be in the timezone where the form server was installed. There are techniques below to convert to different timezones as you need.

The Date/Time control must be initialized using array syntax from a business rule. For example to initialize a Date/Time control named DtTm to January 1st, 2012 at 4:20 am.

```
DtTm.value = ['5/15/2012', '4:20']
```

Rules initializing dates and time will not work in a form.load rule unless you specify a timezone on the form's Url via the _formTz URL parameter. This is because the form server needs to know the timezone in which to return the date and time. If you do not specify a _formTz the methods will return null and the control values will remain blank. For example, to specify Eastern time: &*_formTz=America/New*York.

D.35.1 Duration

This form initializes the hospital discharge date using a rule, and when the user enters the admission date a 2nd rule calculates the number of days the patient stayed in the hospital.

```
// Calculate Hospital Stay Duration
if (A.value != '' && D.value != '') {
    var da = A.value.split('-');
    var Ams = new Date(da[2],da[0]-1,da[1]);
    da = D.value.split('-');
    var Dms = new Date(da[2],da[0]-1,da[1]);

    if (Ams > Dms) {
        Days.value = 'Discharge date must be after Admission Date';
    } else {
        Days.value = (Dms - Ams) / (1000*60*60*24) + ' days';
    }
}
```

D.35.2 Today's Date and Time

Use Oracle Web Forms's built-in date and time methods to set your date, time, and date/time controls to the current date and time in the user's local timezone.

```
if (form.load) {
  Tm.value = frevvo.currentTime(form);
  Dt.value = frevvo.currentDate(form);
  DtTm.value = frevvo.currentDateTime(form);
}
```

The currentTime(), currentDate() and currentDateTime() will not work in a form.load rule unless you specify a timezone on the form's Url via the _formTz Url parameter. This is because the form server needs to know the timezone in which to return the date and time. If you do not specify a formTz the methods will return null and the control values will remain blank. For example &formTz=America/New_York will set the control values to the current date and time in the eastern timezone.

D.35.3 Date/Time Stamp

This rule sets a control named Signature to the value of a control named Name plus a date/time stamp. Note that it is better to use the Oracle Web Forms built-in date and time methods if you want to set the date to the user's local timezone. The following method will set the date to the form server's timezone.

```
var today=new Date()
var m = today.getMonth() + 1;
var d = today.getDate();
var y = today.getFullYear();
var h = today.getHours(); var min = today.getMinutes(); var todayStr = m + '-' + d
+ '-' + y + ' ' + h + ':' + min; Signature.value = 'Signed by ' + Name.value + '
on ' + todayStr;
```

D.35.4 Invalid if Before Today

This rule makes the date control invalid if the date entered isn't before today's date.

```
var today = new Date();
var bd = DOB.value.split('-');
var bd_date = new Date(bd[0],bd[1]-1,bd[2]);

if (bd_date.getTime() > today.getTime()) {
    MyMsg.value = 'Birth Date must be earlier than today!!!!';
    DOB.valid = false;
} else {
    MyMsg.value = 'This is a good Birth Date: ' + DOB.value;
DOB.valid = true;
}
```

D.35.5 Date no more then 14 days from Today

This rule checks that the date entered into a control named AppointmentDate is no more than 14 days greater than today's date.

```
var date1 = DateUtil.today();
var date2 = Appointment.value;
date1 = date1.split("-");
date2 = date2.split("-");
var sDate = new Date(date1[0]+"/"+date1[1]+"/"+date1[2]);
var eDate = new Date(date2[0]+"/"+date2[1]+"/"+date2[2]);
var DaysApart = Math.round((eDate-sDate)/86400000);

if (DaysApart > 14) {
    Appointment.status = "Date should be within 14 days from today's date.";
} else if (DaysApart < 0) {
    Appointment.status = "Date should not be earlier to today's date.";
}
```

D.35.6 Date no more then 30 days ago

This rule checks that the date entered into a control named EventStartDate is not more then 30 days ago.

```
if (EventStartDate.value != "") {
    var date1 = DateUtil.today();
    var date2 = EventStartDate.value;
```

```
    date1 = date1.split("-");
    date2 = date2.split("-");
    var sDate = new Date(date1[0]+"/"+date1[1]+"/"+date1[2]);
    var eDate = new Date(date2[0]+"/"+date2[1]+"/"+date2[2]);
    var days = Math.round((eDate-sDate)/86400000);

if (!eval(parseInt(days) > parseInt(-30))) {
    EventStartDate.valid = false;
    EventStartDate.status = "The date entered can only go back a maximum of 30
days from the current date. Please try again.";
} else {
    EventStartDate.valid = true;
}
```

D.35.7 Central Timezone adjusted for Daylight Savings

This rule adjust today's date in UTC timezone to Central timezone and adjust for daylight savings time. This additional conversion is most commonly needed for Online users as the javascript Date() and Oracle Web Forms' DateUtil.today() both return today's date in UTC timezone.

```
// Converts UTC to either CST or CDT
if (form.load)
{
    var today = new Date();
    var DST = 1; // If today falls outside DST period, 1 extra hr offset
    var Central = 5; // Minimum 5 hr offset from UTC
    // Is it Daylight Savings Time?
    //
    var yr = today.getFullYear();
    // 2nd Sunday in March can't occur after the 14th
    var dst_start = new Date("March 14, "+yr+" 02:00:00");
    // 1st Sunday in November can't occur after the 7th
    var dst_end = new Date("November 07, "+yr+" 02:00:00");
    var day = dst_start.getDay(); // day of week of 14th
    // Calculate 2nd Sunday in March of this year
    dst_start.setDate(14-day); day = dst_end.getDay(); // day of the week of 7th
    // Calculate first Sunday in November of this year dst_end.setDate(7-day);
    // Does today fall inside of DST period?
    if (today >= dst_start && today < dst_end) { DST = 0;
}

    // Adjust Date for Central Timezone
    today.set Hours(today.getHours() - Central - DST);

    var m = today.getMonth() + 1;
    var d = today.getDate();
    var da = today.getDay();
    var y = today.getFullYear();
    var h = today.getHours();
    var min = today.getMinutes();
    if (min < 10) { min = '0' + min;}
    var timezone = ['CDT', 'CST'];
    var dom = ['Sunday', 'Monday', 'Tuesday', 'Wednesday',
               'Thursday', 'Friday', 'Saturday'];
    var todayStr = dom[da] + ' ' + m + '-' + d + '-' + y + ' ' + h + ':' + min + '
' + timezone[DST];

    DateTime.value = todayStr;
```

```
}
```

D.35.8 Hours >= 4 and <= 6 Apart

This rule makes sure the end time is at least 4 hours great then the start time but no more then 6 hours later then the start time. Also start time must be on or after 1:00. The times must be entered in military units. TS is the name of the Time Start control and TE is the name of the Time End control.

- Use Text controls for the start and end times
- Use this pattern in the control to ensure valid military times 1:00 or greater: ([1-9]|1[0-9]|2[0-4]):([0-5][0-9])

```
if (TS.value.length > 0 && TE.value.length > 0) {
    var sTime = TS.value.split(':');
    var sHour = sTime[0];
    var sMin = sTime[1];
    var sMins = sHour * 60 + parseInt(sMin);

    var eTime = TE.value.split(':');
    var eHour = eTime [0];
    var eMin = eTime [1];
    var eMins = eHour * 60 + parseInt(eMin);

    if ((eMins - sMins) < 4*60 || (eMins - sMins) > 6*60)
    {
        TE.valid = false;
    } else {
    TE.valid = true;
    }
}
```

D.35.9 Times

The date control can be set to either just a date, just a time, or a combined date/time. Here are several examples of initializing a time control name Tm;

```
// Both set the control name Tm to the same time
Tm.value = '8:30 pm'; // uses am/pm notation
Tm.vallue = ' 20:00'; // uses military time

// Initialize a date/time requires an array syntax
DtTm.value = ["Aug 1, 2001", "10.00"];

// Copying values
Tm2.value = Tm.value;
Dt2.value = Dt.value;
DtTm2.value = DtTm.value;
```

D.36 Tenants, Roles, Users

Oracle Web Forms has several built-in methods that enable you to access information about your form server such as the list of tenants; users in a tenant; and roles in a tenant. Some of these methods return a boolean true/false value. Others return a JSON string. Here is a sample of how to use these methods.

```
if(form.load)
{
  eval('x=' + frevvo.listTenants());
  Tenants.options = x.tenants; // Init a dropdown list of all tenants on this form
server
}


if(tenant.value.length > 0)


 // Check if a tenant already exists
 if (frevvo.isUniqueRoleId(role.value, tenant.value) == true) {
    ErrMsg.value = 'The role ' + role.value + ' already exists in tenant ' +
tenant.value;
  }
  // Init dropdown with all tehant users
  eval('x=' + frevvo.listUsers(tenant.value));
  Users.options = x.users;
  // Init dropdown with all tenant roles
  eval('x=' + frevvo.listRoles(tenant.value));
  Roles.options = x.roles;
  // Init dropdown will all tenant roles except admin roles
  eval('x=' + frevvo.listRoles(tenant.value, false));
  RolesNA.options = x.roles;
}

// Verify that a role already exists in the tenant
if(role.value.length > 0)
{
  t.value = frevvo.isUniqueRoleId(role.value, tenant.value);
  if (frevvo.isUniqueRoleId(role.value, tenant.value) == false) {
    ErrMsg.value = 'The role ' + role.value + ' already exists in tenant ' +
tenant.value;
  }
}
```

D.37 Repeat Item Added

This rule executes when a new item is added to a repeat. Imagine your form contains a repeating section named Employee with name Erepeat. NOTE: that the name Erepeat is set on the Repeat control and not on the Section control. The Employee section control contain many controls such as Name, Phone, etc. and a dropdown control labeled Manager with named M. It also contains a radio control labeled Employee Shift named ES whose options have been set to 'Day' and 'Evening'.

Oracle Web Forms will execute this rule each time a user clicks "+" on the repeat to add a new item. Here we want to default the Employee Shift ES to the value 'Day', and populate the Manager dropdown dynamically with values from the Oracle Web Forms Database Connector.

Usually your form will have a form.load rule to initialize dropdown options for the 1st repeat item visible on your form by default.

```
if (form.load)
{
    var baseURL = 'http://www.myhost.com:8080/database/';

    // Manager Dropdown
```

```
    eval('x=' + http.get(baseURL + 'Manager'));
    var opts = ['']; // default 1st option to blank
    for (var i=0; i < x.resultSet.length; i++) {
        if (x.resultSet[i]) {
            opts[i+1] = x.resultSet[i].lastname+ ", " + x.resultSet[i].firstname;
    }
}

// Repeat item index 0 is on the form by default
M[0].options = opts;
ES[0].value = 'Day'; // default the employee shift to day
}
```

Now when a new item gets added when a user clicks the "+" icon we can save the overhead of going back to the database to retrieve dynamic options.

```
if (Erepeat.itemAdded)
{
var index = Erepeat.itemIndex; // which item is this in the list

    // No need to go back to the database for options.
    // We already retrieved them in form.load rule for repeat item index 0
M[index].options = M[0].options; ES[index].value = 'Day'; // default the employee
shift to day }
```

Tables are repeats. So the same rule can be written for a table control. The name of a table's repeat is always <TableName>Repeat. For example if you name your table Children. Then the repeat is named ChildrenRepeat.

D.38 Repeat Item Added - Collapse Other Items

This rule executes when a new item is added to a repeat. This form contains a repeating section with a templatized label. It is nice to collapse the other items when adding a new item to keep the repeat list small and grid-like. Medrepeat is the name of the repeat control. Medication is the name of the section control inside the repeat.

```
if (Medrepeat.itemAdded)
{
    var index = Medrepeat.itemIndex;
    for (var i = 0; i < Medication.value.length; i++) {
        if (i != index)
            Medication[i].expanded = false;
        else
            Medication[i].expanded = true;
    }
}
```

D.39 Tables

Tables are identical to repeat controls when referenced in business rules. Tables are a grid layout of repeating items. All the rule examples in this chapter that discuss repeats apply also to tables. The one important note is that you cannot explicitly name the repeat control inside your table. The repeat control inside a table is automatically named as <TableName>Repeat. For example a table named Expense automatically has a repeat named ExpenseRepeat. The rule ExpenseRepeat.itemAdded and ExpenseRepeat.itemIndex references an item added to your table and that item's index respectively.

D.40 form.load

Rules can be used to initialize field values. This is a very useful feature and is often used to dynamically populate dropdown options from a database. Rules using form.load are triggered when a form first loads and when a workflow is loaded from a task list.

Rules using itemAdded only execute for repeat items added when the user clicks +, and for those added from an initial instance document (See Document URIs). It does '"not"' execute for those items that you have added to your form in the Form Designer. You can either added defaults directly via the form designer or add a 2nd rule to your form as follows.

These two rules together initialize the dropdown fields inside a repeat that are already in the form via the Form Designer, as well as those added each time a user clicks "+" on the repeat to add a new item. These controls are initialized based on a value set in another field.

```
'''1st Rule - Default Items'''
if (form.load)
{
    // The form contains two repeat items by default.

    if (department.value == 'Marketing') {
        Managers[0].options = ['Joe', 'Mary', 'Stan', 'Cindy'];
        Managers[1].options = ['Joe', 'Mary', 'Stan', 'Cindy'];
}
    if (department.value == 'Sales') {
        Managers[0].options = ['Lou', 'George', 'Wendy'];
        Managers[1].options = ['Lou', 'George', 'Wendy'];
    }
}

'''2nd Rule - Items Added'''
if (Erepeat.itemAdded)
{
    var index = Erepeat.itemIndex; // which item is this in the list
    ES[index].value = 'Day'; // default the employee shift to day

    // Use options already selected in form.load rule
    Managers[index].options = Manager[0].options;
}
```

This rule is useful in a workflow where you want to make a the tab named Review visible only for the workflow activity named Manager Review.

```
if (form.load) {
    if (_data.getParameter('flow.activity.name') == 'Manager Review') {
Review.visible = true;
    }
}
```

D.41 form.unload

Rules can be used to finalize field values after the users clicks the form's submit button but before the Form and Doc Action execution. Rules using form.unload are triggered when the form user clicks the submit button and for workflows when the user clicks continue to go to the next activity or finish to complete the flow.

One common example use is for an order form order number. You may only want to assign a unique sequential order number to each order submission. You could initialize the form's order number when the form loads using form.load. However if someone start filling in the order form but never submit it you do not want to consume the next order number in sequence if it will never be used. Using form.unload you can assign the number after the submit button click but before the form data is submitted.

Here OrderNum is the name of a invisible control.

```
if (form.unload)
{
    eval('x=' + http.get('http://<webhost>/json/getNextOrdernum'));
    OrderNum.value = x.num;
}
```

D.42 Unique ID

Forms such as invoices, bills of lading, etc. often need to be stamped with a unique ID. The Sequential Number example is one approach, however it has some limitations. One is that you must guarantee that only one person at a time is filling out your form.

Here is a simpler method if your unique IDs do not need to be sequential. The data named form.id is guaranteed to be unique for every new form instance. You can just use it as follows:

```
if (form.load)
{
    InvoiceNum.value = _data.getParameter('form.id');
}
```

D.43 Repeat Item Initialization

The rule above was one example of initializing a newly added repeating control with a default list of options. This same concept is useful if you want to initialize a repeating control's value. When you add a repeat to your form in the Form Designer you can set a default value in any of those repeating controls visible in the designer. However when the user clicks "+" while using the form to add an additional item the default entered in the Form Designer is not automatically used in this new item. In order to accomplish this you can add a simple rule as follows:

This rule initializes the value of one of the fields in the repeat to a default of '0'. RepeatTrack is the name of the repeat control contain the input control named albumOnly. This rule will execute each time a new RepeatTrack item is added when the user clicks "+".

> **Note:** ItemAdded and itemIndex are properties of the Repeat control.

```
if (RepeatTrack.itemAdded)
{
    var index = RepeatTrack.itemIndex;
    albumOnly[index].value = 0;
}
```

Again, to initialize repeat items already on your form in the Form Designer by default, either enter your initial default values directly into the in the Form Designer or add this rule.

```
if (form.load)
{
    for (var i=0; i < albumOnly.value.length; i++) {
        albumOnly[i].value = 0;
    }
}
```

This rule takes into account a repeat where min > 1. If min is 0 or 1, you can simplify this further by removing the from loop and simply have albumOnly[0].value = 0 inside the if (form.load).

D.44 Repeat ItemAdded by Init Doc

ItemAdded also executes when Oracle Web Forms adds items found in an init doc. You may want to only initialize items added when the user clicks "+" on the form. And not those added from an initial document. This form contains a Mailing Label that repeats. Each label needs a unique ID assigned. However once the form is submitted the assigned IDs are saved in the database via the form's Doc URI. When the form loads it adds items automatically from rows in the database. They already have assigned Ids. We only need to assign new Ids in the sequence when the user manually adds a new Mailing Label by clicking the "+" button on the form. MLrepeat is the name of the Mailing Label repeat. MLmid is the name of the ID field in the repeat.

```
if (MLrepeat.itemAdded)
{
  var index = MLrepeat.itemIndex;
  // This rule is fired both when the user clicks "+"
  // and when frevvo adds items found in the init doc.
  // Need to assign new mid only when user clicks "+"

  // New items added via "+" will have a zero length value.
  if (MLmid[index].value.length == 0) {
    // Assign unique ID to label so it can be referenced
    // in RI Mailing Labels field
    // Count the number of existing mailing labels on the form
    var maxId = 0;
    for (var i=0; i < MLmid.value.length; i++)
    {
      if (MLmid[i].value > maxId) {
        maxId = MLmid[i].value;
      }
    }
    var next = parseInt(maxId) + 1;
    MLmid[index].value = next.toFixed(0);
  }
}
```

D.45 Search Popup

Oracle Web Forms forms can initialize dynamically from backend systems. A common form design pattern is the master/detail. An example of master/detail is a form that contains an order number or employee Id. When the user enters an existing Id into the form field you want all other form fields to populate dynamically from values in a

database. This is done using a form created from XSD data source generated by the database connector. This part of the master/detail form design pattern needs no business rule. It happens automatically using the Doc Action manually set document URIs.

This form was created from employee.xsd generated by the Oracle Web Forms database connector. The master Id is a control from that XSD named racfid. The key to this design pattern is in using the master Id in the form's doc action document URI. In this example the form's doc action document URI for the employee document is set to:http://localhost:8082/database/myqset/employee?racfid={racfid}.

Often the master Id is a cryptic number not easily remembered by people using your form. It is a common and useful form design pattern to populate the master Id using a lookup search.

Here is an example form that allows user to view existing employee records and to update their information.

If you know the RACF ID for an employee you can simply type it into that field. If the RACF ID is found in the database the form will automatically populate all other fields from the database (because of the doc action document uri setting). However if you do not know the employee's RACF ID you can click the Find Employee trigger.

This trigger control is named FindEmployee and fires the following rule that makes the search condition section control visible.

```
//Show Employee Search Criteria
if (FindEmployee.clicked)
{
    EmployeeSearch_s1.visible = true;
    EmployeeSearch_s1.expanded = true;
}
```

The section control contains controls that are the search criteria for finding a RACF ID based on other pieces of information you may know about an employee such as name, type, email address. The values entered into the search criteria can be partial values. For instance entering a name "Smith" will find all employees whose name contains the letters "Smith". If you also select email, it will find all employees whose name contains "Smith" and have an email address containing the string "frevvo".

This is the rule that fires when you click the trigger control named Search.

```
//Search Employee database using criteria
if (Search.clicked) {
    eval('x=' + http.get('database/myqset/findEmployee?name={name_s1}&type={type_
s1}
                                                &racfid={id_s1}&email={email_
s1}'));

    var opts= [];
    for (var i=0; i < x.resultSet.length; i++) {
        if (x.resultSet[i]) {
            opts[i] = x.resultSet[i].racfid + "=" +
                      x.resultSet[i].name + ", " +
                      x.resultSet[i].type;
        }
}
    SearchMsg.value = x.resultSet.length +
    ' Matches Found. Change your criteria and click "search" to try again.';
    SearchMsg.visible = true;
    SearchResults.options = opts;
```

```
    SearchResults.visible = true;
}
```

The Search returns one or more matches and dynamically populates a dropdown control named SearchResults. You can change the search criteria to narrow or expand you search. When you select one of the matches from the SearchResults dropdown this 3rd rule executes to copy the selection into the RACF ID control.

```
//Select from Search results
if (SearchResults.value.length > 0)
{
    racfid.value = SearchResults.value;

    // Hide Employee Search and clear values
    EmployeeSearch.visible = false;
    SearchMsg.visible = false;
    SearchResults.visible = false;
    SearchResults.options = [];
}
```

Once the value is copied into RACF ID the form automatically loads all other values in the form from the database. This is due to the form action document Uri setting.

The values in the employee loaded into the form can now be edited. When the users clicks the form's submit button the values will automatically be updated in the database record. This is due to the form action document Uri setting.

E

Preparing Processes for Import into BPMN

This appendix provides information on how to modify your Microsoft Visio and XPDL processes so that they are more easily converted to BPMN.

This appendix contains the following sections:

- Section E.1, "Preparing a Visio File to Import as a BPMN Process"
- Section E.2, "How to Customize XPDL Import Using XSL Doc"
- Section E.3, "Preparing an XPDL File to be Imported as a BPMN Process"

E.1 Preparing a Visio File to Import as a BPMN Process

You can import Visio files into Business Process Composer. Business Process Composer maps Visio elements to BPMN flow objects using a map file named *VisioMasterMap.xml*.

This map file defines how each Visio element is mapped to a BPMN flow object. It is located in the following directory in the Oracle JDeveloper home directory:

`.../jdeveloper/jdev/extensions/tutor/xml.`

> **Note:** Do not edit this file. You can override the mappings in this file by creating and editing a file called VisioUserMap.xml.

The user map is used to extend and/or override the master map. After parsing VisioMasterMap.xml Business Process Composer checks to see if VisioUserMap.xml exists. It then modifies its rules accordingly.

The Visio import module is designed for extensibility. You can add an XML file to support alternate Visio stencils and specify the mapping of additional Visio master shapes to BPMN objects. This file must be named “VisioUserMap.xml”, and it must be placed in the same directory as the “VisioMasterMap.xml” file.

This file should have the same root element as the default defined in VisioMasterMap.xml file, including the reference to the VisioMasterMap.xsd. VisioUserMap.xml must have the same format as the master map, which may be used as a reference.

Entries added to the user map either add new mappings or overwrite existing entries in the main map. VisioUserMap.xml should contain only the extended or overridden entries and should not repeat all the original entries.

Example E–1 Example Entry for VisioUserMap.xml

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<Masters xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oracle.com/oracle/tutor/visio/masterMapElements
VisioMasterMap.xsd"
 xmlns="http://www.oracle.com/oracle/tutor/visio/masterMapElements">
<!– custom <Master/> elements go here -- >
</Masters>
```

E.1.1 How to Update VisioUserMap.xml

This section describes a possible scenario for importing a Visio file into a BPMN process.

To update VisioUserMap.xml:

In this example, the original Visio file contains a shape named Report, that you want to map to a data object. By default, VisioMasterMap.xml maps this shape to a send task. If you to use it strictly as an input /output object, it is more accurate to map it to a Data Object.

1. In the VisioMasterMap.xml file, locate the master definition for DataObject which appears as:

   ```
   <Master Name="data object">
       <BPMNObject>DataObject</BPMNObject>
   </Master>
   ```

   This definition is followed by a series of additional master elements that use the *Like* attribute to clone the first definition and map additional Visio objects:

   ```
   <Master Name="sequential data" Like="data object"/>
   <Master Name="data" Like="data object"/>
   ```

2. To add or modify a mapping for a Visio **Report** element, add the following code to the VisioUserMap.xml file:

   ```
   <Master Name=”Report” Like=”data object”/>
   ```

Example E–2 shows how an object can be mapped to an exiting BPMN object with additional or different attributes by using the **Extends** attribute.

Example E–2 Mapping a Visio Object to a BPMN Object Using the Extends Attribute

```
<Master Name="flow">
    <BPMNObject>SequenceFlow</BPMNObject>
</Master>

<Master Name="conditional flow" Extends="flow">
    <Attributes>
      <Attribute Name="ConditionType" Value="Expression"/>
    </Attributes>
</Master>
```

In the above example, the Visio object *conditional flow* is mapped to the SequenceFlow BPMN flow object, but has added the attribute “ConditionType” to the value “Expression.”

E.1.2 Valid BPMN Element Values

The following are the valid values for the <BPMNObject> element:

- Task
- Subprocess
- Event
- Gateway
- DataObject
- Group
- Annotation
- Lane
- Pool
- MessageFlow
- SequenceFlow
- Association
- null

> **Note:** When you assign the null value to a Visio element, no BPMN flow object is created.

E.1.3 BPMN Element Attributes

The following tables show valid values for BPMN attributes based on the basic BPMN types of BPMN flow objects:

Task attributes and values

Table E–1 show the valid attributes and values for BPMN tasks.

Table E–1 Task Attributes and Values

Attribute	Values
TaskType	None, Script, Reference, Service, User, Manual, Send, Receive
LoopType	Standard, MultiInstance
isForCompensation	true, false

Subprocess attributes and values

Table E–2 show the valid attributes and values for BPMN subprocesses

Table E–2 Subprocess Attributes and Values

Attribute	Values
isExpanded	true, false
isATransaction	true, false
LoopType	Standard, MultIinstance
isForCompensation	true, false
AdHoc	true, false

Event attributes and values

Table E–3 show the valid attributes and values for BPMN events.

Table E–3 Event Attributes and Values

Attribute	Values
EventType	Start, Intermediate, End
Trigger	TriggerNone, Message, Timer, Conditional, Signal, Multiple, Error, Cancel, Compensation, Link, Terminate

Gateway attributes and values

Table E–4 show the valid attributes and values for BPMN gateways.

Table E–4 Gateway Attributes and Values

Attribute	Values
GatewayType	Parallel, Inclusive, Exclusive, Complex
ExclusiveType	Event, Data
MarkerVisible	true, false

Sequence flow attributes and values

Table E–5 show the valid attributes and values for BPMN sequence flows.

Table E–5 Sequence Flow Attributes and Values

Attribute	Values
ConditionType	None, Expression, Default

Association attributes and values

Table E–6 show the valid attributes and values for BPMN data objects.

Table E–6 Association Attributes and Values

Attribute	Value
Direction	None, One

Pool attributes and values

Table E–7 show the valid attributes and values for BPMN pools.

Table E–7 Pool Attributes and Values

Attribute	Values
BoundaryVisible	true, false

E.2 How to Customize XPDL Import Using XSL Doc

The conversion of models from source to target often follows general transformation rules and has guidelines and constraints. If an XPDL document is imported and does not produce desirable results, there may be a mismatch between the format Business Process Composer is expecting and the format of the XPDL document. When this occurs, an XSLT transform may be used to create a better match between the original XPDL diagram and converted BPMN process.

The following section describes the recommended procedures for incorporating an XSL style sheet to modify the output of an Oracle BPM conversion. Oracle BPM ships with several XSLT files that may serve as examples to XSLT developers.

To customize XPDL Import Using XSL Doc:

Business Process Composer uses XSLT transformations to parse XPDL files and generate a consistent format before the data is passed to BPMN. When performing the conversion, Business Process Composer searches for a transformation file associated with an XPDL document.

Typically, an XPDL file contains a <*Vendor*/> element that provides the vendor name. By adding the Vendor's name and the path to an XSL file to the XSLFilePaths.xml, the vendor name is searched to locate the associated transformation file's path.

The file path is retrieved from index file, and Business Process Composer applies the XSLT transformation on the XPDL file.

1. Create a custom XSLT file.

   Using an XML editor, create a new XSLT file. You should place the XSLT file in the XML directory that contains XSLFilePaths.xml.

2. Open the XPDL you are importing, locate the <Vendor/> element under the <PackageHeader/> element, and note its value.

   For example, if the XPDL looks like this:

   ```
   <Package xmlns="http://www.wfmc.org/2004/XPDL2.0alpha" Id="6" Name="Untitled
   Document 6">
       <PackageHeader>
   <Vendor>Global 360</Vendor>
   …
   </PackageHeader>
   …
   </Package>
   ```

   The Vendor element's value is "Global 360".

3. Locate the XSLFilePaths.xml file.

   This file is located in a folder named XML which is nested beneath your JDeveloper install directory as follows:

   ```
   <JDeveloper_Home>\jdeveloper\jdev\extensions\tutor\xml
   ```

4. Add a new <XSLFilePath/> element with a Vendor attribute containing the name of the vendor, and provide the relative path to the XSLT file as shown in the following example:

   ```
   <XSLFilePaths xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="filesPaths PatchFilesPaths.xsd"
   xmlns="filesPaths">
   …
   <XSLFilePath Vendor="Global 360">Global360Patch.xsl</XSLFilePath>
   </XSLFilePaths>
   ```

   Note: In this example the file name Global360Patch.xsl has no additional path information. Business Process Composer will search for this file in the same folder that contains the XSLFilePaths.xml file. If additional path information is present, Business Process Composer will search for the file relative to the \xml folder.

5. Import the XPDL file in Oracle BPM. Your file will be transformed and processed according to the rules specified in the custom XSLT file.

E.3 Preparing an XPDL File to be Imported as a BPMN Process

When importing XDPL files that do not conform to XPDL 2.1 or might be missing elements that Oracle BPM expects, you can customize the conversion using XSLT. See Section E.1, "Preparing a Visio File to Import as a BPMN Process" for instructions on integrating custom XSLT to produce the best results.

This section provides several sample XSLT templates to handle common situations that you may encounter when importing XPDL files.

The files mentioned in this appendix are located in a directory named XML in the Oracle Tutor install directory. These are located in the JDeveloper install directory at:

<*Jdeveloper_Home*>/jdeveloper/jdev/extensions/tutor/xml.

The following are requirements apply when importing business processes into Oracle BPM:

- Oracle BPM supports importing XPDL 2.1 files. The following namespace must be present in the root element of the XPDL source document as the default namespace:

  http://www.wfmc.org/2008/XPDL2.1

- If the XPDLsource file targeted for import does not conform to XPDL 2.1, then you must perform an XSLT transform to generate the correct XPDL format for import into Oracle BPM. The following sections provide information on creating these transforms.

E.3.1 Handling Namespaces

You should be familiar with namespaces handling in XSLT when importing XPDL. The default namespace used by Oracle BPM is:

```
http://www.wfmc.org/2008/XPDL2.1
```

In some cases a source file may be XPDL 2.1 compliant, but still lack elements required by Oracle BPM. In such cases, XSLT can be used to add those elements.

If your XPDL document contains a different default namespace, the elements and attributes are treated as conforming to XPLD 2.1 during import. This will probably produce poor results. XPDL source documents that do not conform to XPDL 2.1 must be transformed via XSL into valid XPDL 2.1 format.

The following possible scenarios are possible:

- The source XPDL elements user the default namespace and conform to XPDL 2.1.
- The source XPDL elements are in default namespace but they don't conform to XPDL 2.1.
- The source XPDL elements are not in default namespace and they don't conform to XPDL 2.1.
- The source XPDL elements are not in default namespace but they conform to XPDL 2.1.

Example E–3 shows an XSL that copies elements written using a local namespace in the source document to the default namespace in the result document.

Note: The local namespace is added in the root element of the stylesheet. When matching elements in the source XPDL document, the local namespace prefix is required for a match to be successful. This example uses ANY_LOCAL_NAMESPACE.

Example E–3 Copying Elements From Local Namespace to the Default Namespace

```
<xsl:stylesheet version = "1.0"
                                xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
                                xmlns:ANY_LOCAL_NAMESPACE = "http://www.someURI" >
<xsl:output method = "xml" indent = "yes"/>
<xsl:template match="@*|node()">
<xsl:copy>
<xsl:apply-templates select="@*|node()"/>
</xsl:copy>
</xsl:template>
<xsl:template match = "ANY_LOCAL_NAMESPACE:Activity">
……
</xsl:template>
</xsl:stylesheet>
```

E.3.2 Handling Relative Coordinates

XPDL documents may have object coordinates defined as relative or absolute. An XPDL document using absolute coordinates will import all coordinates as absolute coordinates.

When importing XPDL documents into Oracle BPM using relative coordinates you should be aware of the following conditions:

- Pool coordinates are absolute
- Lane coordinates are relative to their parent pool
- Node coordinates not in a subprocess are absolute
- Node coordinates in an embedded subprocess (expanded or collapsed) are relative to the upper-left corner of the subprocess
- Sequence Flow path coordinates connecting nodes in a subprocess are relative to the upper-left corner of the subprocess
- All other flows are absolute

An XPDL document with relative coordinates must be converted to meet the above conditions. The following information may be helpful in converting coordinates from relative to absolute or vice versa when necessary:

- Pool coordinates

  Pool coordinates must be absolute. If the coordinates provided for pools are relative, then Business Process Composer renders all pools overlapping each other. Business Process Composer assumes pools contain absolute coordinates. To override this, convert pool coordinates to absolute.

- Lane coordinates

  Lane coordinates must be relative to their parent pool. In some instances, lanes have absolute coordinates but the model has relative coordinates. In this case lane coordinates can be converted as follows.

Lane X-Coordinate can be taken as the difference between lane X-coordinate and its parent pool X-Coordinate. The same logic applies to the Y-coordinate.

The following XSL example shows how to perform this conversion:

```
<xsl:template match =
"xpdl2:Lane/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coordinates">

        <xsl:copy>
            <xsl:copy-of select = "@*[name() != 'XCoordinate' and name() !=
'YCoordinate']"/>
            <xsl:attribute name = "XCoordinate">
                <xsl:value-of select =
"ancestor::xpdl2:Pool/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coor
dinates/@XCoordinate - @XCoordinate"/>
            </xsl:attribute>
            <xsl:attribute name = "YCoordinate">
                <xsl:value-of select =
"ancestor::xpdl2:Pool/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coor
dinates/@YCoordinate - @YCoordinate"/>
            </xsl:attribute>
            <xsl:apply-templates/>
        </xsl:copy>
    </xsl:template>
```

- Node coordinates not in a subprocess

  Node coordinates not in a subprocess must be absolute. If node coordinates are relative to their parent lane or pool, convert them to absolute.

  Coordinates for activities relative to the parent pool can be converted to absolute by adding their pool coordinates to the activity coordinates.

  Coordinates for activities relative to parent lanes, which are relative to the parent Pool, can be converted to absolute by assigning the sum of the activity, parent lane and parent pool coordinates to activity coordinate.

  The following example shows how to calculate activity coordinates if they are relative to the parent lane and the lane coordinates are relative to the parent pool where the pool coordinates are absolute.

```
<xsl:template
match="xpdl2:Activity/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coor
dinates">
        <xsl:copy>
            <xsl:copy-of select="@*[name()!='YCoordinate' and
name()!='XCoordinate']"/>
            <xsl:variable name="LaneId">
                <xsl:value-of
select="ancestor::xpdl2:NodeGraphicsInfo/@LaneId"/>
            </xsl:variable>
            <xsl:variable name="PoolId">
                <xsl:value-of
select="//xpdl2:Lane[@Id=$LaneId]/ancestor::xpdl2:Pool/@Id"/>
            </xsl:variable>
            <xsl:attribute name="YCoordinate">
                <xsl:value-of
select="//xpdl2:Pool[@Id=$PoolId]/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInf
o/xpdl2:Coordinates/@YCoordinate
                +
//xpdl2:Lane[@Id=$LaneId]//xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2
:Coordinates/@YCoordinate
```

```
  +
@YCoordinate"/>
            </xsl:attribute>
            <xsl:attribute name="XCoordinate">
                <xsl:value-of
select="//xpdl2:Pool[@Id=$PoolId]/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInf
o/xpdl2:Coordinates/@XCoordinate
                +
//xpdl2:Lane[@Id=$LaneId]//xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2
:Coordinates/@XCoordinate
  +
@XCoordinate"/>
            </xsl:attribute>
        </xsl:copy>
    </xsl:template>
```

- Node coordinates in an embedded subprocess

  Node coordinates in an embedded subprocess (expanded or collapsed) must be relative to the upper-left corner of the subprocess

  For child nodes of a subprocess, coordinates are relative to the parent subprocess.

  If the coordinates of child nodes of a subprocess are not relative to the parent subprocess, it is necessary to determine whether the coordinates are relative to the parent lane or parent pool.

  To make node coordinates relative to the subprocess, subtract the appropriate subprocess coordinates from the node coordinates and assign the resulting values to the node coordinates.

  A sample style sheet template is given below that calculates the coordinates of the child nodes of a subprocess. This template will work if the child node coordinates are not relative to a subprocess.

```
<xsl:template match =
"xpdl2:ActivitySet/xpdl2:Activities/xpdl2:Activity/xpdl2:NodeGraphicsInfos/xpdl
2:NodeGraphicsInfo/xpdl2:Coordinates">
        <xsl:variable name = "ActivitySetId">
            <xsl:value-of select = "ancestor::xpdl2:ActivitySet/@Id"/>
        </xsl:variable>
        <xsl:copy>
            <xsl:copy-of select = "@*[name() != 'XCoordinate' and name() !=
'YCoordinate']"/>
            <xsl:attribute name = "XCoordinate">
                <xsl:value-of select = "@XCoordinate  -
 //xpdl2:Activity[@Id =
$ActivitySetId]/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coordinate
s/@XCoordinate"/>
            </xsl:attribute>
            <xsl:attribute name = "YCoordinate">
                <xsl:value-of select = "@YCoordinate -
//xpdl2:Activity[@Id =
$ActivitySetId]/xpdl2:NodeGraphicsInfos/xpdl2:NodeGraphicsInfo/xpdl2:Coordinate
s/@YCoordinate"/>
            </xsl:attribute>
        </xsl:copy>
    </xsl:template>
```

- Sequence Flow path coordinates

Sequence Flow path coordinates connecting nodes in a subprocess must be relative to the upper-left corner of the subprocess.

In your XSL transform, you should determine whether sequence flow path coordinates are relative to Subprocess or not. If not, convert them to relative to upper-left corner of the Subprocess.

Simple program logic can be used to convert coordinates to relative. Such logic would subtract the subprocess coordinates from the sequence flow path coordinates. All other flows must be absolute.

Also, in your XSL code you should determine whether coordinates are relative or absolute. If coordinates are not absolute, convert them. To convert, include templates in a style sheet that contains logic to add the parent pool coordinates to the flow coordinates and then assigns these values to the flow coordinates.

It is important to note the following if coordinates are relative to the parent pool or the parent lane:

- If sequence flows are relative to the parent pool, add the pool coordinates to the flow coordinates.
- If sequence flows are relative to the parent lane, add the parent pool coordinates and the parent lane coordinates to the flow coordinates.

E.3.3 Handling Extended Attributes

Two extended attributes are used by Business Process Composer to parse XPDL files. These extended attributes are optional but can be very useful when working with XPDL documents that employ relative coordinates for flows. These attributes are:

- redrawConnections
- isRelativeObjectCoordinates

These extended attributes are not standard XPDL elements but can be used to mitigate the amount of work required to convert Flow coordinates from relative to absolute or from absolute to relative. Although the <ExtendedAttributes> element is found in many elements of the XPDL document, Oracle BPM will search for it under <Package/> element. Place the <ExtendedAttributes> element under <Package/> element to ensure it is located by Oracle BPM.

E.3.4 Handling redrawConnections

Setting this attribute to true will let Oracle BPM redraw flow connections instead of using the original coordinates. This is useful when coordinates of flows need to be converted to relative or absolute. If it is hard to convert coordinates of flows to relative or absolute, then setting this attribute to true will let Oracle BPM recreate the coordinates for flows.

When this attribute is set to true, there is no need to convert coordinates to absolute or relative. Oracle BPM will not consider the original coordinates of flows but will create the most suitable coordinates to flows.

Oracle BPM will use the original coordinates when the ExtendedAttribute is not included in XPDL file or is set to false.

Therefore, use of this attribute should be considered when retaining the layout of the original model is of lesser importance than providing a shorter development time.

Example E–4 shows how to set this attribute.

Example E–4 Setting the ExtendedAttribute when Handling redrawConnections

```
<xsl:template match="xpdl:Package/xpdl:ExtendedAttributes">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xpdl:ExtendedAttribute Name="redrawConnections" Value="true"/>
                <xsl:apply-templates/>
            </xsl:copy>
        </xsl:template>
        <xsl:template match="xpdl:Package">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xsl:if test="not(child::xpdl:ExtendedAttributes)">
                    <xpdl:ExtendedAttributes>
                        <xpdl:ExtendedAttribute Name="redrawConnections"
Value="true"/>
                    </xpdl:ExtendedAttributes>
                </xsl:if>
                <xsl:apply-templates/>
            </xsl:copy>
        </xsl:template>
```

The above templates add the 'redrawConnections' ExtendedAttribute to the <ExtendedAttributes/> element as a child of < Package/>.

E.3.5 Handling isRelativeObjectCoordinates

The isRelativeObjectCoordinates extended attribute is used to notify Oracle BPM that object coordinates in the XPDL file are relative or absolute.

If the source XPDL file presents ALL object coordinates in absolute form then every coordinate is measured from the 0,0 point at the top left corner of the diagram. This attribute should be used and its value set to false.

Setting this extended attribute to true informs Oracle BPM that all coordinates in the source XPDL file are relative in conformance with the relative coordinates rules of Oracle BPM as specified above.

Oracle BPM by default assumes all coordinates of input document comply with the relative coordinates rules of Oracle BPM as outlined above. When this attribute is set to true or is omitted, make sure that all coordinates meet the relative coordinates rules. Otherwise, you should include templates in your style sheet to make them conform to the relative coordinates rules.

Example E–5 shows how to include the <ExtendedAttribute/> element in the <ExtendedAttributes/> element, which is a child of the <Package/> element, is given below.

Example E–5 Including the ExtendedAttribute in the ExtendedAttributes Element

```
<xsl:template match="xpdl21:Package/xpdl21:ExtendedAttributes">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xpdl21:ExtendedAttribute Name="isRelativeObjectCoordinates"
Value="false"/>
                <xsl:apply-templates/>
            </xsl:copy>
        </xsl:template>
        <xsl:template match="xpdl21:Package">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xsl:if test="not(child::xpdl21:ExtendedAttributes)">
```

```
                    <xpdl21:ExtendedAttributes>
                        <xpdl21:ExtendedAttribute
Name="isRelativeObjectCoordinates" Value="false"/>
                    </xpdl21:ExtendedAttributes>
                </xsl:if>
                <xsl:apply-templates/>
            </xsl:copy>
        </xsl:template>
```

E.3.6 Removing Invisible Elements

Oracle BPM considers all graphical elements of the source XPDL file to be visible elements, even if their visibility is set to false. Thus you may find some differences as formerly invisible elements will be visible in the converted BPMN process.

To resolve this issue, you must remove these elements from the XPDL with the help of an XSL style sheet templates as shown in Example E–6.

Example E–6 Removing Invisible Elements

```
<xpdl:Activity Name="ProcessGroup" Id="ProcessGroup">
  <xpdl:NodeGraphicsInfos>
                        <xpdl:NodeGraphicsInfo ToolId="XYZ" LaneId="PMCoE"
IsVisible="false">
                            <xpdl:Coordinates XCoordinate="7740.0"
YCoordinate="80.0"/>
                        </xpdl:NodeGraphicsInfo>
                    </xpdl:NodeGraphicsInfos>
   ……..
</xpdl:Activity>
```

In this example the activity's visibility is set to false, but when this model is imported into Oracle BPM, you will still see this activity. To eliminate invisible elements include a style sheet template to remove them.

Example E–7 shows how to remove invisible activities.

Example E–7 Removing Invisible Activities

```
<xsl:template match="xpdl:Activity">
        <xsl:variable name="isVisible">
            <xsl:choose>
                <xsl:when
test="xpdl:NodeGraphicsInfos/xpdl:NodeGraphicsInfo/@IsVisible = 'false'">
                    <xsl:text>false</xsl:text>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:text>true</xsl:text>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:if test="$isVisible = 'true'">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xsl:apply-templates/>
            </xsl:copy>
        </xsl:if>
    </xsl:template>
```

E.3.7 Handling the Orientation Attribute

Orientation is an attribute found in many XPDL documents to specify the orientation of the model. This attribute must have HORIZONTAL or VERTICAL as its value. XPDL documents generated by some tools may have coordinates that identify the model as horizontal when the model is actually in vertical orientation and vice versa.

Make sure the model has correct coordinates with respect to the model orientation. If model has mismatching coordinates, include style sheet templates to swap the coordinates.

The orientation attribute of model is also important to Oracle BPM for calculating dimensions of pools and lanes (which will be discussed later in this document). The orientation attribute can be found in pool elements. If this attribute is not found in the XPDL document then the model is assumed to be in horizontal orientation by Oracle BPM.

If the orientation attribute is in a tool specific namespace it must be made available to Oracle BPM by creating an orientation attribute on the pool elements and setting their values to the one found in tool specific namespace.

Example E–8 shows how to set orientation of pools. This template tries to find the orientation of pools. If it is not found then this template will set the orientation attribute to HORIZONTAL.

Example E–8 Setting the Orientation of Pools

```
<xsl:template match="xpdl:Pool">
        <xsl:variable name="orientation">
            <xsl:choose>
                <xsl:when test="not(@Orientation)">
                    <xsl:text>HORIZONTAL</xsl:text>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="@Orientation"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:copy>
            <xsl:copy-of select="@*"/>
            <xsl:attribute name="Orientation">
                <xsl:value-of select="$orientation"/>
            </xsl:attribute>
            <xsl:apply-templates/>
        </xsl:copy>
    </xsl:template>
```

E.3.8 Specifying the View Type for Subprocesses

The view attribute of <BlockActivity/> and <Subflow/> elements specify whether the subprocess is an expanded or a collapsed subprocess. Oracle BPM assumes the default view type of subprocess elements is COLLAPSED. For this reason, if a model has an expanded subprocess element but its' view' type is not given in XPDL file, Oracle BPM will render this element as a collapsed subprocess element.

Note: Subflow elements of an XPDL file cannot be rendered with their child elements into Oracle BPM, only the subflow elements themselves will be found in the imported model. It is possible to identify the subprocess according to its view type. If the subflow is expanded Oracle BPM will render this element as an expanded subflow element.

The following example shows <BlockActivity/> and <Subflow/> elements that do not have a view type specified. Their corresponding interpretation by Oracle BPM will result in a collapsed subprocess.

```
<xpdl2:BlockActivity ActivitySetId="_gJ5DQeE3Ed6tmt0cZVxmlA"/>
<xpdl2:SubFlow Id="_swKUEGyzEd6oxIP3ZfQL-g" PackageRef="ProcessPackage"/>
```

If the view attribute is not found in <BlockActivity> element, it may be present in a tool specific namespace. If so, include templates in XSL to create view attribute on <BlockActivity> or <Subflow> elements.

Example E–9 shows a style sheet used to add the view attribute to <Subflow> elements.

Example E–9 Adding the View Attribute to a Subflow Element

```
<xsl:template match="xpdl21:SubFlow">
            <xsl:copy>
                <xsl:copy-of select="@*"/>
                <xsl:attribute name="View">
                    <xsl:choose>
                        <xsl:when
test="ancestor::xpdl21:Activity/xpdl21:NodeGraphicsInfos/xpdl21:NodeGraphicsInfo/@
Expanded='false'">
                            <xsl:text>COLLAPSED</xsl:text>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:text>EXPANDED</xsl:text>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:attribute>
            </xsl:copy>
        </xsl:template>
```

Example E–10 shows a sample style sheet template that adds the View attribute to <BlockActivity> elements. This template will work XPDL files generated by Oracle Business Process Management Studio. This template also shows how to create the view attribute on <BlockActivity> elements by accessing the view attribute value from a tool specific namespace.

Example E–10 Adding the View Attribute to BlockActivity Elements

```
<xsl:template match="xpdl:Activity/xpdl:BlockActivity">
  <xsl:copy>
   <xsl:copy-of select="@*"/>
   <xsl:choose>
    <xsl:when
test="ancestor::xpdl:Activity/xpdl:Extensions/albpm:ALBPMExtensions/albpm:FeatureS
et/albpm:BooleanFeature[@ name='collapsed']/@value='true'">
     <xsl:attribute name="View">
      <xsl:text>COLLAPSED</xsl:text>
```

```
     </xsl:attribute>
    </xsl:when>
    <xsl:otherwise>
        <xsl:attribute name="View">
  <xsl:text>EXPANDED</xsl:text>
</xsl:attribute>
    </xsl:otherwise>
   </xsl:choose>
   <xsl:apply-templates/>
  </xsl:copy>
 </xsl:template>
```

E.3.9 Handling the Object Pin

In some tools the coordinates provided for activities are given for the upper-left corner of the activity object, but for others the coordinates are based on the center of the activity object. These coordinates serve as the object pin.

When an XPDL document containing an object pin at center of activities is imported into Oracle BPM, the activities may occur at different positions than expected. This is because the object pin for activities is located in the center but Oracle BPM expects the object pin in the upper left corner. To resolve this discrepancy the activity coordinates must be recalculated.

You can use simple logic to calculate these coordinates, such as subtracting half of the activity's width from its XCoordinate and subtracting half of the activity's height from its YCoordinate.

Example E–11 shows a style sheet template that does this recalculation. Note that this template will work only if width, height, x, and y-coordinates are provided for activities.

Example E–11 Recalculating the Location of an Object Pin

```
<xsl:template match =
"xpdl:Activity/xpdl:NodeGraphicsInfos/xpdl:NodeGraphicsInfo/xpdl:Coordinates">
<xsl:copy>
            <xsl:copy-of select = "@*[name() != 'XCoordinate' and name() !=
'YCoordinate']"/>
            <xsl:attribute name = "XCoordinate">
                <xsl:value-of select = "@XCoordinate -
ancestor::xpdl:NodeGraphicsInfo/@Width div 2 "/>
            </xsl:attribute>
            <xsl:attribute name="YCoordinate">
                <xsl:value-of select = "@YCoordinate -
ancestor::xpdl:NodeGraphicsInfo/@Height div 2"/>
            </xsl:attribute>
            <xsl:apply-templates/>
        </xsl:copy>
    </xsl:template>
```

E.3.10 Modifying the Height and Width of Activities

Some tools do not provide height and width for activities in their XPDL files. But coordinates and dimensions of activities are needed to import an XPDL file into Oracle BPM. So include templates in an XSL style sheet that will set height and width for activities and lanes. If the XPDL file does not contain height and width for activities, set some default dimensions for them. For instance, set 80x40 width and height for tasks, collapsed subprocesses, and set 30x30 width and height to events and gateways.

It can be especially difficult to set the height and width for expanded <BlockActivity> and <Subflow> elements, as an expanded <BlockActivity> element may also contain another expanded <BlockActivity> elements. Here the innermost BlockActivity height and width should be calculated first and then its parent BlockActivity. This recursion must bubble up to the topmost <BlockActivity>. It can be difficult to code this recursion process in XSL. For this reason Oracle BPM provides a feature which calculates height and width of expanded Subprocesses. To use this feature set the height and width of <BlockActivities> to 0,0 using XSL templates.

Example E–12 is an event element which does not have width and height defined.

Example E–12 Event Element that Does Not Contain Height and Width Defined

```
<xpdl:Activity Name="Begin" Id="Begin">
                    <xpdl:Event>
<xpdl:StartEvent Trigger="None"/>
                    </xpdl:Event>
   ……..
      <xpdl:NodeGraphicsInfos>
                        <xpdl:NodeGraphicsInfo LaneId="Accounting"
IsVisible="true">
                            <xpdl:Coordinates XCoordinate="36.0"
YCoordinate="110.0"/>
                        </xpdl:NodeGraphicsInfo>
                    </xpdl:NodeGraphicsInfos>
 </xpdl:Activity>
```

Example E–13 shows a sample style sheet template used to set height and width of activities. This template can be used if height and width of activities are not given in the XPDL document. This example sets 80 x 40 dimensions (width, height) for task elements and collapsed BlockActivities, it will set 30 x 30 dimensions to route and gateways, and it will set 0x0 dimensions to expanded BlockActivities and thus lets Oracle BPM calculate the dimensions for these expanded BlockActivity elements. This template will work for Oracle BPM Studio XPDL files.

Example E–13 Setting the Height and Width of Activities

```
<xsl:template match =
"xpdl:Activity/xpdl:NodeGraphicsInfos/xpdl:NodeGraphicsInfo">
        <xsl:variable name = "activityType">
            <xsl:choose>
<xsl:when test = "ancestor::xpdl:Activity/xpdl:Implementation/xpdl:SubFlow">
                    <xsl:text>SubFlow</xsl:text>
                </xsl:when>
                <xsl:when test = "ancestor::xpdl:Activity/xpdl:Implementation">
                    <xsl:text>Task</xsl:text>
                </xsl:when>
                <xsl:when test = "ancestor::xpdl:Activity/xpdl:Event">
                    <xsl:text>Event</xsl:text>
                </xsl:when>
                <xsl:when test = "ancestor::xpdl:Activity/xpdl:Route">
                    <xsl:text>Route</xsl:text>
                </xsl:when>
                <xsl:when test = "ancestor::xpdl:Activity/xpdl:BlockActivity">
                    <xsl:choose>
                        <xsl:when test =
"ancestor::xpdl:Activity/xpdl:Extensions/albpm:ALBPMExtensions/albpm:FeatureSet/al
bpm:BooleanFeature[@ name = 'collapsed']/@value != 'true'">
                            <xsl:text>ExpandedBlockActivity</xsl:text>
                        </xsl:when>
```

```
                        <xsl:otherwise>
                            <xsl:text>CollapsedBlockActivity</xsl:text>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:when>
            </xsl:choose>
        </xsl:variable>

        <xsl:copy>
          <xsl:copy-of select = "@*"/>
              <xsl:attribute name = "Width">
                  <xsl:choose>
                      <xsl:when test = "$activityType = 'Task' or $activityType =
'CollapsedBlockActivity' or $activityType = 'SubFlow'">
                          <xsl:text>80</xsl:text>
                      </xsl:when>
                      <xsl:when test = "$activityType = 'Route' or $activityType =
'Event'">
                          <xsl:text>30</xsl:text>
                      </xsl:when>
                      <xsl:when test = "$activityType = 'ExpandedBlockActivity'">
                          <xsl:text>0</xsl:text>
                      </xsl:when>
                  </xsl:choose>
              </xsl:attribute>
              <xsl:attribute name = "Height">
                  <xsl:choose>
                      <xsl:when test = "$activityType = 'Task' or $activityType =
'CollapsedBlockActivity' or $activityType = 'SubFlow'">
                          <xsl:text>40</xsl:text>
                      </xsl:when>
                      <xsl:when test = "$activityType = 'Route' or $activityType =
'Event'">
                          <xsl:text>30</xsl:text>
                      </xsl:when>
                      <xsl:when test = "$activityType = 'ExpandedBlockActivity'">
                          <xsl:text>0</xsl:text>
                      </xsl:when>
                  </xsl:choose>
              </xsl:attribute>
          <xsl:apply-templates/>
        </xsl:copy>
    </xsl:template>
```

E.3.11 Modifying the Height and Width of Lanes

In many XPDL documents, width or height dimensions for lanes is provided depending on orientation of the parent pool. For instance, if the orientation of the parent pool is horizontal then the width of the lane might be found in the XPDL document but not its height. As mentioned before, the height and width of lanes and activities are needed for Oracle BPM to determine the sizes of graphical elements. If the lane height or width is not found in the XPDL document, set these attributes to a value which is sufficient to hold all its containing elements.

If the height of lanes is provided but not their width, simple logic can be used to set the width of lanes. Find the largest sum of x-coordinate plus the widths of Activities, and set this value to the widths of all lanes. If the lane width is merely sufficient to hold all elements, you can find the lane right border running from the right border of activity that has the largest sum of x-coordinate and width. A small padding value can

also be added to the largest sum to extend the lanes enough to allow them to appear as the containers of the other objects.

The above logic will work if all activities contain width and height values. But there are cases where activities might not contain height and width values. In such cases it is difficult to calculate lane widths using XSLT with the above logic as each activity height and width should be calculated before calculating lane height or width. To resolve this problem, Oracle BPM provides a feature that will set the lane widths or heights.

To use this feature set the missing dimension of lane to zero using XSLT. This feature assumes one dimension was provided for the lane.

E.3.12 Modifying the Height and Width of Pools

If height and widths are provided for pools, Oracle BPM will use those dimensions for Pools. If these values are absent, Oracle BPM will try to calculate them. Oracle BPM will calculate both dimensions even if one of dimensions is provided. The source XPDL document must contain both dimensions for Pools to circumvent this feature

E.3.13 Location of Activities

When coordinates and dimensions are provided in the XPDL file, Oracle BPM will use those values without manipulating them. But if some dimensions are missing, such as dimensions for Lanes dimensions or Subprocesses, Oracle BPM will try to calculate these dimensions. In the process of calculating dimensions, Oracle BPM will add some padding to the dimensions for the model to be friendlier but this model will look good only if the activities have some space around them. Otherwise the imported model may have borders of activities or lanes running on top of other activities or lanes. To avoid this problem position activities with some space around them.

Problems may occur when importing models containing duplicate ids for more than one lane or activity into Oracle BPM. Oracle BPM cannot create more than one lane or activity with the same name. Only one lane or activity from the source will be created, and the results will not accurately reflect the original model.

To avoid this problem, create model elements with unique ids.

E.3.14 Including Missing Elements

If the XPDL source document is missing elements or attributes required for Oracle BPM to properly perform its conversion, then add those elements and attributes using XSLT.

For instance, there are 8 types of tasks in XPDL. For Oracle BPM to recognize these task types, a <Task> element should contain another child element which specifies whether the Task is a service task, a receive task, and so on. If these child elements are not found under a source <Task> element, that <Task> element will be converted to a default <Task> element.

For example, the <Task> element shown below is a user task but does not have a child <TaskUser> element. Hence it is assumed to be a default Task element by Oracle BPM

```
<Implementation>
            <Task />
 </Implementation>
```

In order for the activity to be identified by Oracle BPM as user task, a <TaskUser> element must be added under Task element as follows:

```
<Implementation>
<Task>
        <TaskUser>
....
                    </TaskUser>
</Task>
</Implementation>
```

As mentioned before many attributes may be given under tool specific namespaces. If task type is not found under the XPDL namespace, try finding it in the tool specific namespace and include templates in the style sheet to include these elements under the Task element.

Example E–14 demonstrates how to include a <TaskService> element as a child of <Task> element whenever it finds an "Automatic" Task element. This template will work for XPDL source files generated by Oracle BPM Studio.

Example E–14 Including a TaskService Element as a Child of the Task Element

```
<xsl:template match="xpdl:Activity/xpdl:Implementation/xpdl:Task">
  <xsl:copy>
   <xsl:copy-of select="@*"/>
       <xsl:if
test="ancestor::xpdl:Activity/xpdl:Extensions/albpm:ALBPMExtensions/albpm:FeatureS
et/albpm:StringFeature[@name='type']/@value='AUTOMATIC'
and not(child::xpdl:TaskService)">
     <xpdl:TaskService/>
    </xsl:if>
     <xsl:apply-templates/>
  </xsl:copy>
 </xsl:template>
```

E.3.15 Checking the Correctness of Activities

Be sure that the source XPDL contains the correct elements to represent activities when they are exported as XPDL from the source tool. For example, if a model contains an event activity and while exporting that model into XPDL the tool creates <Route> or <Task> element for that Event Activity, this element must be replaced with an element that accurately represents an Event Activity.

Consider the XPDL element shown below. The activity is a start event, but instead of an event element under the <Activity> element, you find a <Route> element. When this model is imported into Oracle BPM, this activity is converted into a route activity rather than as an event activity.

Example E–15 Checking the Correctness of Activities

```
<xpdl:Activity Name="Group$Begin" Id="Group$Begin">
                            <xpdl:Route GatewayType="XOR" MarkerVisible="true"/>
……….
</xpdl:Activitiy>
The correct notation for the above element should be:
<xpdl:Activity Name="Group$Begin" Id="Group$Begin">
<xpdl:Event>
                        <xpdl:StartEvent Trigger="None"/>
        ……..
                </xpdl:Event>
    ……..
</xpdl:Activity>
```

If this problem is found in your source XPDL, include templates in a style sheet to replace the incorrect elements with correct elements.

F

The Sales Quote Example Process

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

See Appendix A, "BPMN Flow Object Reference" for more detailed information about Oracle's implementation of BPMN 2.0.

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

This chapter includes the following sections:

- Section F.1, "Introduction to Business Process Management Notation (BPMN)"
- Section F.2, "Introduction to the Sales Quote Example"

F.1 Introduction to Business Process Management Notation (BPMN)

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

F.1.1 What is Business Process Management Notation (BPMN)

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

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

F.1.2 Business Processes

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

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

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

F.1.2.1 Process Instances

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

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

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

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

F.1.2.2 Process Tokens

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

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

F.1.3 Flow Objects

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

- Tasks: represent the work performed by a process.
- Events: define something that happens during a process.
- Gateways: determine the flow of your process.
- Sequence flows: connect flow objects and define the logical flow of a process.

F.1.4 Data Objects

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

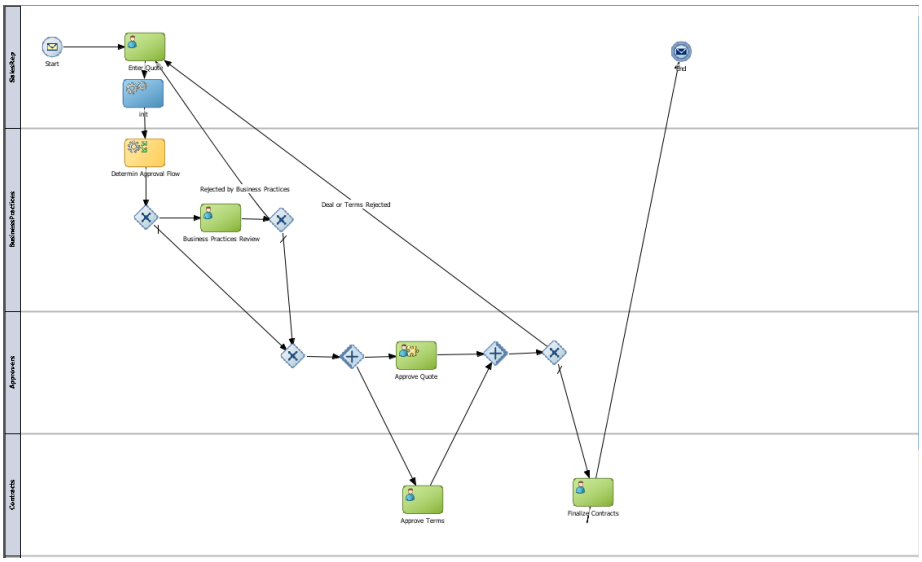
For example, the Sales Quote example defines several data objects used to store information about the sales quote. At runtime, the process instance generates and stores specific values for these variables.

F.2 Introduction to the Sales Quote Example

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

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

Figure F–1 The Sales Quote Example



****

F.2.1 Breakdown of the Sales Quote Example

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

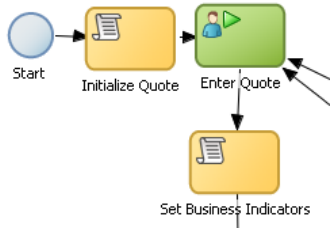
- Initiate Sales Quote
- Determine Business Practice Review
- Approve Quote
- Approvals Outcome

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

F.2.1.1 Initiate Sales Quote

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

Figure F–2 Initiate Sales Quote



The initiate sales quote portion performs the following:

1. Defines the start point of the process (none start event)

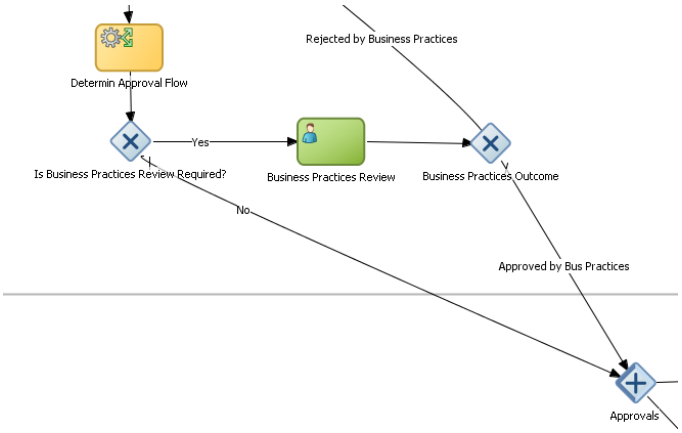
2. Initializes data objects (script task)
3. Initiates the process instance (user task with initiator pattern)
4. Set values for the business indicator data objects (script task)

F.2.1.2 Determine Business Practice Review

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

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

Figure F–3 Determine Business Practice Review



****

A process instances passes through the business practice review as follows:

1. Determine Approval Flow (business rules task).

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

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

F.2.1.3 Approve Quote

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

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

- Approve the sales quote.

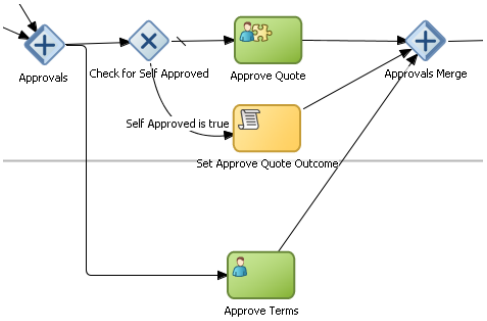
This process path is also split into two paths. In this example, it is possible for the quote to be self approved, which means that the quote is approved based on certain criteria, or it may require explicit approval from a process participant.

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

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

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

Figure F–4 Approve Quote



A process instances passes through the approve quote section of the process as follows:

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

      Approve Quote

      Set Approve Quote Outcome

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

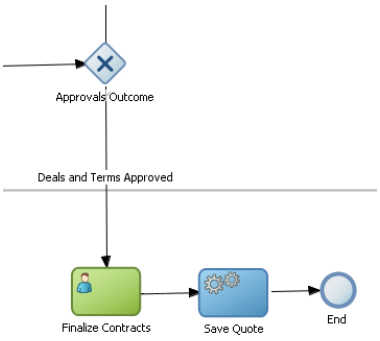
F.2.1.4 Approvals Outcome

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

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

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

Figure F–5 Approval Outcome



A process instances passes through the approvals outcome section of the process as follows:

1. Approvals outcome (exclusive gateway)

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

   a. Approved

      This is implemented with a default sequence flow.

      Finalize Quote

      Save Quote

      End Event

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

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